Institiúid Teicneolaíochta Cheatharlach

INSTITUTE of TECHNOLOGY CARLOW

At the Heart of South Leinster

4th Year Final Project – Technical Manual

BSc (Honours) Software Development

Name: Alex Matthews
Student ID: C00208942
Supervisor: Dr. Oisin Cawley
Date: 12/04/2019

Contents

Introduction

This document will first go through the requirements of the project and how to install the project locally. The code from the project is also listed in this document.

Prerequisites

This is a list of software that needs to be installed on the system.

- Python 3.x (https://www.python.org/downloads/)
- Pip (https://www.liquidweb.com/kb/install-pip-windows/)
- React JS 16.8.3 Starter Kit (https://react-cn.github.io/react/downloads.html)
- Flask 1.0.2 (https://pypi.org/project/Flask/)
- Flask-CORS (https://flask-cors.readthedocs.io/en/latest/)

Running the Application

Frontend

Before running the application there are three third party React plugins that need to be installed. In order to do this, navigate to the React project folder and type the following commands:

- **`npm install react-infinite-calendar –save`**
- **`npm install --save react-toastify`**
- **`npm i react-loading`**

Once these commands are executed and the necessary libraries are installed, execute the following command to run start the React front-end: **`npm start`**

Once **`npm start`** is executed, the application will be available on localhost, port 3000 (127.0.0.1:3000).

Backend

As long as Python, flask, and flask-CORS is installed on the system, just navigate to the API directory on the command line. Change directory into the `py` folder. From there, run **`python app.py.`** This will start the API on localhost, port 5000.

In order for the API to function there must be a live database running. I used the built in SQL server in XAMPP but any SQL server could be used. The SQL setup file which creates the database and all of the tables can be found here: https://github.com/itcOnlineGaming/ErasmusAPI/tree/master/Docker/MySQL. Simply import the file into your SQL server and the database will be automatically created.

React Source Code

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';
import './tabs.css';

ReactDOM.render(<App />, document.getElementById('root'));
```

App.js

```
import React, {Component} from 'react';
import Tabs from './Tabs';
import FoodDiary from './FoodDiary';
import Recipes from './Recipes';

class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      activeSection: 'Food Diary',
      addEntryClicked: false,
      createRecipeClicked: false,
    };
  }


  handleSectionTabChange(tab) {
    this.setState({activeSection: tab});
  }

  handleAddEntryClick() {
    this.setState({addEntryClicked: !this.state.addEntryClicked});
  }

  handleCreateRecipeClick() {
    this.setState({createRecipeClicked: !this.state.createRecipeClicked});
  }

  handleAddEntryBackClick() {
    if (window.confirm("Are you sure you want to exit? All changes will be lost!")) {
      this.setState({addEntryClicked: !this.state.addEntryClicked});
    }
  }

  handleCreateRecipeBackClick() {
    this.setState({createRecipeClicked: !this.state.createRecipeClicked});

  }
```

```
  handleSubmitButtonClick() {
    this.setState({addEntryClicked: !this.state.addEntryClicked});
  }

  handleRecipeSubmitButtonClick() {
    this.setState({createRecipeClicked: !this.state.createRecipeClicked});
  }

  render() {
    return (
      <>
      <Tabs classVariant="section-" onClickTab={(tab) =>
this.handleSectionTabChange(tab)}>
        <div label="Food Diary">
          <FoodDiary
            title="Food Diary"
            addEntryClick={() => this.handleAddEntryClick()}
            showAddEntry={this.state.addEntryClicked}
            addEntryBackClicked={() => this.handleAddEntryBackClick()}
            handleSubmitButtonClick={() => this.handleSubmitButtonClick()}
          />
        </div>
        <div label="Recipes">
          <Recipes
            title="Recipes"
            createRecipeClick={() => this.handleCreateRecipeClick()}
            showCreateRecipe={this.state.createRecipeClicked}
            createRecipeBackClicked={() => this.handleCreateRecipeBackClick()}
            handleSubmitButtonClick={() => this.handleRecipeSubmitButtonClick()}
          />
        </div>
      </Tabs>
      </>
    );
  }
}

export default App;
```

FoodDiary.js

```
import React, {Component} from 'react';
import Banner from './Banner';
import PickDateBanner from './PickDateBanner';
import DisplayDateBanner from './DisplayDateBanner';
import MealTabs from './MealTabs';
import Calendar from './Calendar'
import EntryList from './EntryList';
import AddEntryScreen from './AddEntryScreen';
import EditEntryScreen from './EditEntryScreen';
import ReactLoading from 'react-loading';
import EmptyEntryList from './EmptyEntryList';
```

```
import ViewEntry from './ViewEntry';
import FoodDiaryContextMenu from './FoodDiaryContextMenu';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

class FoodDiary extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      showCalendar: false,
      selectedDate: new Date(),
      entryClicked: false,
      activeMeal: 'Breakfast',
      entries: [],
      isLoaded: false,
      error: null,
      entrySubmitted: false,
      showEditEntry: false,
      editEntryData: null,
      showViewEntry: false,
      showContextMenu: false,
      contextMenuData: null,
      contextMenuID: null,
    };

    this.handleEditEntry = this.handleEditEntry.bind(this);
    this.handleViewEntry = this.handleViewEntry.bind(this);
    this.editEntryBackButtonClicked = this.editEntryBackButtonClicked.bind(this);
    this.viewEntryBackButtonClicked = this.viewEntryBackButtonClicked.bind(this);
    this.handleSubmitButtonClick = this.handleSubmitButtonClick.bind(this);
    this.handleSubmitButtonClick2 = this.handleSubmitButtonClick2.bind(this);
    this.handleContextClick = this.handleContextClick.bind(this);
    this.handleDateClick = this.handleDateClick.bind(this);
  }

  handleContextClick(entry_id, data) {
    this.setState({showContextMenu: !this.state.showContextMenu, contextMenuID: entry_id,
contextMenuData: data});
  }

  componentDidMount() {
    var date = this.state.selectedDate.getFullYear() + "-" +
(this.state.selectedDate.getMonth() + 1) + "-" + this.state.selectedDate.getDate();
    fetch("http://10.40.6.204:5000/diary/get_diary_entries/1?meal=" +
this.state.activeMeal + "&date=" + date)
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isLoaded: true,
            entries: result
          });
```

```
        },
        (error) => {
          this.setState({
            isLoaded: true,
            error
          });
        }
      )
  }

  componentDidUpdate(prevProps, prevState) {
    if (this.state.activeMeal !== prevState.activeMeal || this.state.selectedDate !==
prevState.selectedDate || this.state.entrySubmitted !== prevState.entrySubmitted) {
      var date = this.state.selectedDate.getFullYear() + "-" +
(this.state.selectedDate.getMonth() + 1) + "-" + this.state.selectedDate.getDate()
      fetch("http://10.40.6.204:5000/diary/get_diary_entries/1?meal=" +
this.state.activeMeal + "&date=" + date)
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isLoaded: true,
            entries: result
          });
        },
        (error) => {
          this.setState({
            isLoaded: true,
            error
          });
        }
      )
    }
  }

  handleDateButtonClick() {
    this.setState({showCalendar: !this.state.showCalendar});
  }

  handleAddEntryClick() {
    this.props.addEntryClick();
  }

  handleDateChange(date) {
    this.setState({selectedDate: date});
    this.setState({showCalendar: !this.state.showCalendar});
  }

  handleEntryClicked() {
    this.setState({entryClicked: !this.state.entryClicked});
  }
```

```
handleMealTabChange(tab) {
  this.setState({activeMeal: tab});
}

handleSubmitButtonClick(meal) {
  this.props.handleSubmitButtonClick();
  this.setState({activeMeal: meal});
  this.setState({entrySubmitted: !this.state.entrySubmitted});
  toast("Entry successfully added. Keep it up!", {className: "toast"});
}

handleSubmitButtonClick2(meal) {
  this.setState({showEditEntry: !this.state.showEditEntry});
  this.setState({activeMeal: meal});
  this.setState({entrySubmitted: !this.state.entrySubmitted});
  toast("Entry successfully updated. Keep it up!", {className: "toast"});
}

handleDeleteEntry() {
  this.setState({entrySubmitted: !this.state.entrySubmitted});
  toast("Entry successfully deleted!", {className: "toast"});
}

handleEditEntry(entry_data){
  this.setState({showEditEntry: !this.state.showEditEntry});
  this.setState({editEntryData: entry_data})
}

handleViewEntry(entry_data){
  this.setState({showViewEntry: !this.state.showViewEntry});
  this.setState({editEntryData: entry_data})
}

editEntryBackButtonClicked(meal) {
  if (window.confirm("Are you sure you want to exit? All changes will be lost!")) {
    this.setState({showEditEntry: !this.state.showEditEntry});
    this.setState({activeMeal: meal});
  }
}

viewEntryBackButtonClicked(meal) {
  this.setState({showViewEntry: !this.state.showViewEntry});
  this.setState({activeMeal: meal});
}

divClick() {
  this.setState({showContextMenu: false, showCalendar: false});
}

handleDateClick() {
  this.setState({showCalendar: !this.state.showCalendar});
}
```

```
  render() {
    if (this.state.showEditEntry) {
      return (
        <EditEntryScreen
          entry_data={this.state.editEntryData}
          onBackButton={this.editEntryBackButtonClicked}
          handleSubmitButtonClick={this.handleSubmitButtonClick2}
        />
      )
    }
    else if (this.state.showViewEntry) {
      return (
        <ViewEntry
          entry_data={this.state.editEntryData}
          onBackButton={this.viewEntryBackButtonClicked}
        />
      )
    }
    return (
      <>
        { this.props.showAddEntry ?
          <AddEntryScreen
            onBackButton={() => this.props.addEntryBackClicked()}
            selectedDate={this.state.selectedDate.toISOString()}
            handleSubmitButtonClick={this.handleSubmitButtonClick}
            selectedMeal={this.state.activeMeal}
          />
        :
        <div label="Food Diary">
          <Banner title = {this.props.title}/>
          <PickDateBanner
            onClick={() => this.handleDateButtonClick()}
            onAddClick={() => this.handleAddEntryClick()}
          />
          { this.state.showCalendar ?
            <div
              className="context-div"
              onClick={() => this.divClick()}>
              <Calendar onSelect={(date) => this.handleDateChange(date)}/>
            </div> : null }
          <DisplayDateBanner title = {this.state.selectedDate.toDateString()}
handleOnClick={this.handleDateClick}/>
          {this.state.showContextMenu ?
            <div
              className="context-div"
              onClick={() => this.divClick()}>
              <FoodDiaryContextMenu
                entry_id={this.state.contextMenuID}
                entry_data={this.state.contextMenuData}
                handleDeleteEntry={() => this.handleDeleteEntry()}
                handleEditEntry={this.handleEditEntry}
```

```
                handleViewEntry={this.handleViewEntry}
              />
            </div>: null}
          <MealTabs classVariant="meals-" onClickTab={(tab) =>
this.handleMealTabChange(tab)} selectedTab={this.state.activeMeal}>
            <div label="Breakfast">
              {

                this.state.isLoaded ?
                  this.state.entries.length != 0 ?
                    <EntryList
                      entries={this.state.entries}
                      handleDeleteEntry={() => this.handleDeleteEntry()}
                      handleEditEntry={this.handleEditEntry}
                      handleViewEntry={this.handleViewEntry}
                      handleContextClick={this.handleContextClick}
                    />
                  :
                !this.state.showCalendar ?
                  <EmptyEntryList onAddClick={() => this.handleAddEntryClick()}/> : null
                :
                <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
              }
            </div>
            <div label="Lunch">
              {
                this.state.isLoaded ?
                  this.state.entries.length != 0 ?
                    <EntryList
                      entries={this.state.entries}
                      handleDeleteEntry={() => this.handleDeleteEntry()}
                      handleEditEntry={this.handleEditEntry}
                      handleViewEntry={this.handleViewEntry}
                      handleContextClick={this.handleContextClick}
                    />
                  :
                !this.state.showCalendar ? <EmptyEntryList onAddClick={() =>
this.handleAddEntryClick()}/> : null
                :
                <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
              }
            </div>
            <div label="Dinner">
              {
                this.state.isLoaded ?
                  this.state.entries.length != 0 ?
                    <EntryList
                      entries={this.state.entries}
                      handleDeleteEntry={() => this.handleDeleteEntry()}
                      handleEditEntry={this.handleEditEntry}
```

```
                    handleViewEntry={this.handleViewEntry}
                    handleContextClick={this.handleContextClick}
                  />
                :
                !this.state.showCalendar ? <EmptyEntryList onAddClick={() =>
this.handleAddEntryClick()}/> : null
                :
                <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
              }
            </div>
            <div label="Snacks">
              {
                this.state.isLoaded ?
                  this.state.entries.length != 0 ?
                    <EntryList
                      entries={this.state.entries}
                      handleDeleteEntry={() => this.handleDeleteEntry()}
                      handleEditEntry={this.handleEditEntry}
                      handleViewEntry={this.handleViewEntry}
                      handleContextClick={this.handleContextClick}
                    />
                  :
                !this.state.showCalendar ? <EmptyEntryList onAddClick={() =>
this.handleAddEntryClick()}/> : null
                :
                <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
              }
            </div>
            <div label="Drinks">
              {
                this.state.isLoaded ?
                  this.state.entries.length != 0 ?
                    <EntryList
                      entries={this.state.entries}
                      handleDeleteEntry={() => this.handleDeleteEntry()}
                      handleEditEntry={this.handleEditEntry}
                      handleViewEntry={this.handleViewEntry}
                      handleContextClick={this.handleContextClick}
                    />
                  :
                !this.state.showCalendar ? <EmptyEntryList onAddClick={() =>
this.handleAddEntryClick()}/> : null
                :
                <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
              }
            </div>
          </MealTabs>
        </div>
        }
```

```
        <ToastContainer />
      </>
    );
  }
}

export default FoodDiary;
```

EntryList.js

```
import React, {Component} from 'react';
import Entry from './Entry';

class EntryList extends React.Component {
  render() {
    return (
      <div className="entries">
        <ol className="entry-list">
        {this.props.entries.map((entry, i) => {
          var time = entry['time'].substring(0, entry['time'].length-3)
          return (
            <Entry
              key={i}
              entry_id={entry['entry_id']}
              label={entry['title']}
              image={"http://10.40.6.204:5000/uploads/" + entry['image']}
              time={time}
              handleDeleteEntry={() => this.props.handleDeleteEntry()}
              handleEditEntry={this.props.handleEditEntry}
              handleViewEntry={this.props.handleViewEntry}
              entry_data={entry}
              handleContextClick={this.props.handleContextClick}
            />
          );
        })}
        </ol>

      </div>
    );
  }
}

export default EntryList;
```

Entry.js

```
import React, {Component} from 'react';

class Entry extends React.Component {
  constructor(props) {
    super(props);
  };
```

```
  handleEntryClick() {
    this.props.handleContextClick(this.props.entry_id, this.props.entry_data);
  }

  render() {
    let className = 'entry-list-item';
    return (
      <li className={className} onClick={() => this.handleEntryClick()}>
        <div className="entry-div">
          <img src={this.props.image} className="entry-image"/>
          <label className="entry-name-header">{this.props.label}</label>
          <label className="entry-time-label">{this.props.time}</label>
        </div>
      </li>
    );
  }
}

export default Entry;
```

AddEntryBanner.js

```
import React, {Component} from 'react';

class AddEntryBanner extends React.Component {
  render() {
    return (
      <div className="AEbannerDiv">
        <button
          className="AEbannerButton"
          type="button"
          onClick={this.props.onBackButton}>
            ←
          </button>
        <h1 className="AEbannerHead">{this.props.title}</h1>
      </div>
    );
  }
}

export default AddEntryBanner;
```

AddEntryScreen.js

```
import React, {Component} from 'react';
import AddEntryBanner from './AddEntryBanner'
import FoodDiaryQuickAdd from './FoodDiaryQuickAdd';

class AddEntryScreen extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      file: '/image_placeholder.png',
```

```
    title: "",
    description: "",
    meal: this.props.selectedMeal,
    datetime: this.props.selectedDate,
    error: null,
    form_submitted: false,
  };

  this.handlePicChange = this.handlePicChange.bind(this);
  this.handleTitleChange = this.handleTitleChange.bind(this);
  this.handleDescChange = this.handleDescChange.bind(this);
  this.handleMealChange = this.handleMealChange.bind(this);
  this.handleTimeChange = this.handleTimeChange.bind(this);
  this.handleSubmit = this.handleSubmit.bind(this);
  this.handleSubmitClick = this.handleSubmitClick.bind(this);
  this.handleQuickAddClick = this.handleQuickAddClick.bind(this);
}

handlePicChange(event) {
  if(event.target.files[0]) {
    this.setState({
      file: URL.createObjectURL(event.target.files[0])
    })
  }

}

handleTitleChange(event) {
  this.setState({title: event.target.value});
}

handleDescChange(event) {
  this.setState({description: event.target.value});
}

handleMealChange(event) {
  this.setState({meal: event.target.value});
}

handleTimeChange(event) {
  this.setState({datetime: event.target.value});
}

handleSubmitClick() {
  if(this.state.file == '/image_placeholder.png') {
    alert("Please select a photo!");
  }
}

handleSubmit(event) {
  event.preventDefault();
  const data = new FormData(event.target);
```

```
    let param = JSON.stringify(this.state).replace(/&/g, "%26");
    fetch("http://10.40.6.204:5000/diary/add_diary_entry/1?title=" +
this.state.title.replace(/&/g, "%26") + "&date=" + this.state.datetime + "&meal=" +
this.state.meal + "&desc=" + param, {
        method: 'post',
        body: data
      })
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            form_submitted: true,
          });
        },
        (error) => {
          this.setState({
            form_submitted: true,
            error
          });
        }
      )
    this.props.handleSubmitButtonClick(this.state.meal);

  }

  handleQuickAddClick(entryData) {
    if (window.confirm("Add Entry?")) {
      let param = JSON.stringify(this.state).replace(/&/g, "%26");
      fetch("http://10.40.6.204:5000/diary/add_diary_entry/1?title=" +
entryData['title'].replace(/&/g, "%26") + "&desc=" + param + "&date=" +
this.state.datetime + "&meal=" + this.state.meal + "&image=" + entryData['image'], {
        method: 'post',
      })
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            form_submitted: true,
          });
        },
        (error) => {
          this.setState({
            form_submitted: true,
            error
          });
        }
      )
    this.props.handleSubmitButtonClick(this.state.meal);
    }

  }
```

```
  render() {
    return (
        <>
        <AddEntryBanner onBackButton={this.props.onBackButton} title="Add Entry"/>
        <FoodDiaryQuickAdd selectedMeal={this.state.meal}
handleClick={this.handleQuickAddClick}/>
        <img src={this.state.file} id="add-entry-img"/>
        <form className="add-entry-form" name="add-entry-form"
onSubmit={this.handleSubmit} encType="multipart/form-data">
          <input type="file" name="file" id="file" className="add-entry-pic-input"
accept="image/*" onChange={this.handlePicChange} required />
          <label htmlFor="file">Choose a Picture</label>
          <br/><br/>
          <label htmlFor="title" id="add-entry-title-input-label" className="add-entry-
labels">Title</label>
          <br/>
          <input
            className="add-entry-title-input input-style"
            name="title" maxLength="40" type="text"
            value={this.state.title}
            onChange={this.handleTitleChange}
            placeholder="Enter title here..."
            required autoComplete="off"
          />
          <br/><br/>
          <label htmlFor="desc" id="add-entry-desc-input-label" className="add-entry-
labels">Description</label>
          <br/>
          <textarea
            className="add-entry-desc-input input-style"
            name="desc" type="text" form="add-entry-form"
            value={this.state.decription}
            onChange={this.handleDescChange}
            placeholder="Enter description here (optional)...">
          </textarea>
          <br/><br/>
          <label htmlFor="desc" id="add-entry-meal-input-label" className="add-entry-
labels">Meal</label>
          <br/>
          <select id="add-entry-meal-input" className="input-style"
value={this.state.meal} onChange={this.handleMealChange} required>
              <option value = "Breakfast">Breakfast</option>
              <option value = "Lunch">Lunch</option>
              <option value = "Dinner">Dinner</option>
              <option value = "Snacks">Snacks</option>
              <option value = "Drinks">Drinks</option>
          </select>
          <br/><br/>
          <label htmlFor="timestamp" id="add-entry-timestamp-input-label" className="add-
entry-labels">Date/Time</label>
```

```
          <br/>
          <input
            value={this.state.datetime.substr(0, 16)}
            type="datetime-local" id="add-entry-timestamp-input"
            className="input-style" name="timestamp"
            onChange={this.handleTimeChange} required >
          </input>
          <br/><br/>
          <input className="add-entry-pic-submit-btn" type="submit" value="Submit"
onClick={this.handleSubmitClick} />
        </form>

        </>
    );
  }
}

export default AddEntryScreen;
```

EditEntryScreen.js

```
import React, {Component} from 'react';
import AddEntryBanner from './AddEntryBanner'

class EditEntryScreen extends React.Component {
  constructor(props) {
    super(props);
    let meal_list = ["Breakfast", "Lunch", "Dinner", "Snacks", "Drinks"]
    this.state = {
      file: "http://10.40.6.204:5000/uploads/" + this.props.entry_data['image'],
      title: this.props.entry_data['title'],
      description: this.props.entry_data['description'],
      meal: meal_list[this.props.entry_data['meal']],
      datetime: this.props.entry_data['iso_datetime'],
      error: null,
      form_submitted: false,
    };

    this.handlePicChange = this.handlePicChange.bind(this);
    this.handleTitleChange = this.handleTitleChange.bind(this);
    this.handleDescChange = this.handleDescChange.bind(this);
    this.handleMealChange = this.handleMealChange.bind(this);
    this.handleTimeChange = this.handleTimeChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
    this.handleBackButton = this.handleBackButton.bind(this);
  }

  handlePicChange(event) {
```

```
    if(event.target.files[0]) {
      this.setState({
        file: URL.createObjectURL(event.target.files[0])
      })
    }

  }

  handleTitleChange(event) {
    this.setState({title: event.target.value});
  }

  handleDescChange(event) {
    this.setState({description: event.target.value});
  }

  handleMealChange(event) {
    this.setState({meal: event.target.value});
  }

  handleTimeChange(event) {
    let date = new Date(event.target.value);
    this.setState({datetime: date.toISOString()});
  }

  handleSubmit(event) {
    event.preventDefault();
    const data = new FormData(event.target);
    fetch("http://10.40.6.204:5000/diary/update_diary_entry/" +
this.props.entry_data['entry_id'] + "?title=" + this.state.title.replace(/&/g, "%26") +
"&desc=" + this.state.description.replace(/&/g, "%26") + "&date=" + this.state.datetime +
"&meal=" + this.state.meal, {
        method: 'post',
        body: data
      })
      .then(res => res.json())
      .then(
        (result) => {

          this.setState({
            form_submitted: true,
          });
        },
        (error) => {
          this.setState({
            form_submitted: true,
            error
          });
        }
      )
    this.props.handleSubmitButtonClick(this.state.meal);
  }
```

```
  handleBackButton() {
    this.props.onBackButton(this.state.meal);
  }

  render() {
    return (
        <>
          <AddEntryBanner onBackButton={this.handleBackButton} title="Edit Entry"/>
          <img src={this.state.file} id="add-recipe-img"/>
          <form className="add-entry-form" name="add-entry-form"
onSubmit={this.handleSubmit} encType="multipart/form-data">
            <input type="file" name="file" id="file" className="add-entry-pic-input"
accept="image/*" onChange={this.handlePicChange}/>
            <label htmlFor="file">Choose a New Picture</label>
            <br/><br/>
            <label htmlFor="title" id="add-entry-title-input-label" className="add-entry-
labels">Title</label>
            <br/>
            <input
              className="add-entry-title-input input-style"
              name="title" maxLength="40" type="text"
              value={this.state.title} onChange={this.handleTitleChange}
              placeholder="Enter title here..." required autoComplete="off"
            />
            <br/><br/>
            <label htmlFor="desc" id="add-entry-desc-input-label" className="add-entry-
labels">Description</label>
            <br/>
            <textarea
              className="add-entry-desc-input input-style"
              name="desc" type="text" form="add-entry-form"
              onChange={this.handleDescChange}
              placeholder="Enter description here (optional)...">
                {this.state.description}
            </textarea>
            <br/><br/>
            <label htmlFor="desc" id="add-entry-meal-input-label" className="add-entry-
labels">Meal</label>
            <br/>
            <select id="add-entry-meal-input" className="input-style"
onChange={this.handleMealChange} value={this.state.meal} required>
              <option value = "Breakfast">Breakfast</option>
              <option value = "Lunch">Lunch</option>
              <option value = "Dinner">Dinner</option>
              <option value = "Snacks">Snack</option>
              <option value = "Drinks">Drink</option>
            </select>
            <br/><br/>
            <label htmlFor="timestamp" id="add-entry-timestamp-input-label"
className="add-entry-labels">Date/Time</label>
            <br/>
```

```
            <input
              value={this.state.datetime.substr(0, 16)}
              type="datetime-local" id="add-entry-timestamp-input"
              className="input-style" name="timestamp"
              onChange={this.handleTimeChange} required >
            </input>
            <br/><br/>
            <input className="add-entry-pic-submit-btn" type="submit" value="Submit" />
          </form>
        </>
    );
  }
}


export default EditEntryScreen;
```

EmptyEntryList.js

```
import React, {Component} from 'react';

class EmptyEntryList extends React.Component {
  render() {
    return (
      <div className="empty-entry-list-div">
        <button className="no-entries-button" onClick={this.props.onAddClick}>+</button>
        <p className="empty-entry-list-p2">Click to add a new entry!</p>
      </div>
    );
  }
}

export default EmptyEntryList;
```

FoodDiaryContextMenu.js

```
import React, {Component} from 'react';

class FoodDiaryContextMenu extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      isDeleted: false,
      error: null,
    };
  }

  handleDeleteButton() {
    if (window.confirm("Are you sure you want to delete this entry?")) {
      fetch("http://10.40.6.204:5000/diary/delete_diary_entry/" + this.props.entry_id,
{method: 'post'})
      .then(res => res.json())
      .then(
```

```
        (result) => {
          this.setState({
            isDeleted: true,
            response: result
          });
        },
        (error) => {
          this.setState({
            isDeleted: false,
            error
          });
        }
      )
      this.props.handleDeleteEntry();
    }
  }

  handleEditButton() {
    this.props.handleEditEntry(this.props.entry_data);
  }

  handleViewButton() {
    this.props.handleViewEntry(this.props.entry_data);
  }

  render() {
    let myDropId = "myDropdown" + this.props.entry_id;
    return (
      <div id={myDropId} className="dropdown-content2">
        <button className="context-button" onClick={() =>
this.handleViewButton()}>View</button>
        <button className="context-button" onClick={() =>
this.handleEditButton()}>Edit</button>
        <button className="context-button" onClick={() =>
this.handleDeleteButton()}>Delete</button>
      </div>
    );
  }
}

export default FoodDiaryContextMenu;
```

FoodDiaryQuickAdd.js

```
import React, {Component} from 'react';
import QuickAddEntry from './QuickAddEntry';

class FoodDiaryQuickAdd extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      entries: [],
```

```
      isLoaded: false,
      meal: this.props.selectedMeal,
    };
  }

  componentDidMount() {
    fetch("http://10.40.6.204:5000/diary/get_quick_add_entries/" +
this.props.selectedMeal)
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isLoaded: true,
            entries: result
          });
        },
        (error) => {
          this.setState({
            isLoaded: true,
            error
          });
        }
      )
  }

  componentDidUpdate(prevProps, prevState) {
    if (this.props.selectedMeal != prevProps.selectedMeal) {
      fetch("http://10.40.6.204:5000/diary/get_quick_add_entries/" +
this.props.selectedMeal)
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isLoaded: true,
            entries: result
          });
        },
        (error) => {
          this.setState({
            isLoaded: true,
            error
          });
        }
      )
    }
  }

  render() {
    return (
      <>
        <label id="quick-add-label">Quick Add</label>
        <div className="quick-add">
```

```
        {
          this.state.entries.length == 0 ? "Nothing to show.." :
          !this.state.isLoaded ? "Please Wait.." :
            this.state.entries.map((entry, i) => {
              return(
                <QuickAddEntry
                  entryData={entry}
                  handleClick={this.props.handleClick}
                />
              );
            })
        }




        </div>
      </>

    );
  }
}

export default FoodDiaryQuickAdd;
```

QuickAddEntry.js

```
import React, {Component} from 'react';

class QuickAddEntry extends React.Component {

  handleClick() {
    this.props.handleClick(this.props.entryData);
  }

  render() {
    return (
      <div className="quick-add-entry" onClick={() => this.handleClick()}>
        <img className="quick-add-img" src={"http://10.40.6.204:5000/uploads/" +
this.props.entryData['image']} /><br/>
        <label className="quick-add-title">{this.props.entryData['title']}</label>
      </div>
    );
  }
}

export default QuickAddEntry;
```

ViewEntry.js

```
import React, {Component} from 'react';
import AddEntryBanner from './AddEntryBanner'
```

```
class ViewEntry extends React.Component {
  constructor(props) {
    super(props);
    let meal_list = ["Breakfast", "Lunch", "Dinner", "Snacks", "Drinks"]
    this.state = {
      meal: meal_list[this.props.entry_data['meal']],
    };

    this.handleBackButton = this.handleBackButton.bind(this);
  }

  handleBackButton() {
    this.props.onBackButton(this.state.meal);
  }

  render() {
    return (
        <>
          <AddEntryBanner onBackButton={this.handleBackButton} title="View Entry"/>
          <div className="view-entry-div">
          <h1 className="view-entry-title">{this.props.entry_data['title']}</h1>
            <img src={"http://10.40.6.204:5000/uploads/" + this.props.entry_data['image']}
id="view-entry-img"/>
            <p className="view-entry-meal">{this.state.meal}</p>
            <p className="view-entry-date">{new
Date(this.props.entry_data['date']).toDateString()}</p>
            <div className="view-entry-desc-div">
              <label>Description:</label>
              <p>{this.props.entry_data['description'] != "" ?
this.props.entry_data['description'] : "No Description Given."}</p>
            </div>
          </div>
        </>
    );
  }
}


export default ViewEntry;
```

Calendar.js

```
import React from 'react';
import InfiniteCalendar from 'react-infinite-calendar';
import 'react-infinite-calendar/styles.css';

class Calendar extends React.Component {
  render() {
    var today = new Date();
    var lastWeek = new Date(today.getFullYear(), today.getMonth(), today.getDate() - 7);
    return (
      <InfiniteCalendar
```

```
        className="infinite-calendar"
        width={250}
        height={300}
        selected={today}
        minDate={lastWeek}
        onSelect={this.props.onSelect}
      />
    );
  }
}

export default Calendar;
```

Banner.js

```
import React, {Component} from 'react';

class Banner extends React.Component {
  render() {
    return (
      <div className="bannerDiv">
        <h1 className="bannerHead">{this.props.title}</h1>
      </div>
    );
  }
}

export default Banner;
```

DisplayDateBanner.js

```
import React, {Component} from 'react';

class DisplayDateBanner extends React.Component {

  handleOnClick() {
    this.props.handleOnClick();
  }

  render() {
    return (
      <div className="DDbannerDiv">
        <h1 className="DDbannerHead" onClick={() =>
this.handleOnClick()}>{this.props.title}</h1>
      </div>
    );
  }
}

export default DisplayDateBanner;
```

PickDateBanner.js

```
import React, {Component} from 'react';
```

```
class PickDateBanner extends React.Component {
  render() {
    return (
      <div className="PDbannerDiv">
        <button className="PDbannerAddButton" type="button"
onClick={this.props.onAddClick}>Add Entry</button>
        <button className="PDbannerButton" type="button" onClick={this.props.onClick}>Pick
Date</button>
      </div>
    );
  }
}

export default PickDateBanner;
```

Recipes.js

```
import React, {Component} from 'react';
import Banner from './Banner';
import MealTabs from './MealTabs';
import RecipesBanner from './RecipesBanner';
import CreateRecipeScreen from './CreateRecipeScreen';
import RecipeList from './RecipeList';
import EmptyRecipeList from './EmptyRecipeList';
import EditRecipeScreen from './EditRecipeScreen';
import ReactLoading from 'react-loading';
import ViewRecipe from './ViewRecipe';
import MyRecipeContextMenu from './MyRecipeContextMenu';
import CommRecipeContextMenu from './CommRecipeContextMenu';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

class Recipes extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      title: "My Recipes",
      activeRecipes: "My Recipes",
      commRecipesActiveMeal: "Breakfast",
      myRecipesActiveMeal: "Breakfast",
      recipeList: [],
      isLoaded: false,
      recipeSubmitted: false,
      showEditRecipe: false,
      editRecipeData: null,
      showViewRecipe: false,
      showContextMenu: false,
      contextMenuData: null,
      contextMenuID: null,
      showContextMenuComm: false,
      contextMenuDataComm: null,
```

```
      contextMenuIDComm: null,
    };

    this.handleCreateRecipeBackButton = this.handleCreateRecipeBackButton.bind(this);
    this.handleEditRecipe = this.handleEditRecipe.bind(this);
    this.handleViewRecipe = this.handleViewRecipe.bind(this);
    this.editRecipeBackButtonClicked = this.editRecipeBackButtonClicked.bind(this);
    this.handleEditRecipeSubmitButton = this.handleEditRecipeSubmitButton.bind(this);
    this.handleCreateRecipeSubmitButtonClick =
this.handleCreateRecipeSubmitButtonClick.bind(this);
    this.handleViewRecipeBackButton = this.handleViewRecipeBackButton.bind(this);
    this.handleContextClickMyRecipe = this.handleContextClickMyRecipe.bind(this);
    this.handleContextClickComm = this.handleContextClickComm.bind(this);
  }

  handleContextClickMyRecipe(recipe_id, data) {
    this.setState({showContextMenu: !this.state.showContextMenu, contextMenuID: recipe_id,
contextMenuData: data});
  }

  handleContextClickComm(recipe_id, data) {
    this.setState({showContextMenuComm: !this.state.showContextMenu, contextMenuIDComm:
recipe_id, contextMenuDataComm: data});
  }

  handleCreateRecipeClick() {
    this.props.createRecipeClick();
  }

  handleRecipesTabChange(tab) {
    this.setState({title: tab});
    this.setState({activeRecipes: tab});
  }

  handleMyRecipesMealTabChange(tab) {
    this.setState({myRecipesActiveMeal: tab});
  }

  handleCommRecipesMealTabChange(tab) {
    this.setState({commRecipesActiveMeal: tab});
  }

  handleDeleteRecipe() {
    this.setState({recipeSubmitted: !this.state.recipeSubmitted});
  }

  handleShareRecipe() {
    this.setState({recipeSubmitted: !this.state.recipeSubmitted});
  }

  handleViewRecipe(recipe_data) {
    this.setState({showViewRecipe: !this.state.showViewRecipe});
```

```
  this.setState({editRecipeData: recipe_data});
}

handleCreateRecipeBackButton(meal, recipeSection) {
  if (window.confirm("Are you sure you want to exit? All changes will be lost!")) {
    this.setState({activeRecipes: recipeSection});
    this.setState({commRecipesActiveMeal: meal});
    this.setState({myRecipesActiveMeal: meal});
    this.props.createRecipeBackClicked();
    this.forceUpdate();
  }
}

handleCreateRecipeSubmitButtonClick(meal, recipeSection) {
  this.props.handleSubmitButtonClick();
  this.setState({activeRecipes: recipeSection});
  this.setState({commRecipesActiveMeal: meal});
  this.setState({myRecipesActiveMeal: meal});
  this.setState({recipeSubmitted: !this.state.recipeSubmitted});
  this.forceUpdate();
  toast("Recipe Successfully Created!", {className: "toast"});
}

handleEditRecipe(recipe_data){
  this.setState({showEditRecipe: !this.state.showEditRecipe});
  this.setState({editRecipeData: recipe_data});
}

editRecipeBackButtonClicked(meal, activeRecipeSection) {
  if (window.confirm("Are you sure you want to exit? All changes will be lost!")) {
    this.setState({showEditRecipe: !this.state.showEditRecipe});
    this.setState({activeRecipes: activeRecipeSection});
    this.setState({myRecipesActiveMeal: meal});
  }
}

handleEditRecipeSubmitButton(meal, recipeSection) {
  this.setState({showEditRecipe: !this.state.showEditRecipe});
  this.setState({myRecipesActiveMeal: meal});
  this.setState({activeRecipes: recipeSection});
  this.setState({recipeSubmitted: !this.state.recipeSubmitted});
  toast("Recipe Successfully Updated!", {className: "toast"});
}

handleViewRecipeBackButton(meal, recipeSection) {
  this.setState({showViewRecipe: !this.state.showViewRecipe});
  this.setState({myRecipesActiveMeal: meal});
  this.setState({activeRecipes: recipeSection});
}

componentDidMount() {
```

```
    fetch("http://10.40.6.204:5000/recipes/get_personal_recipes/1?meal=" +
this.state.myRecipesActiveMeal)
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isLoaded: true,
            recipeList: result
          });
        },
        (error) => {
          this.setState({
            isLoaded: true,
            error
          });
        }
      )
  }

  componentDidUpdate(prevProps, prevState) {
    if (this.state.activeRecipes == "My Recipes") {
      if (this.state.activeRecipes !== prevState.activeRecipes ||
this.state.myRecipesActiveMeal !== prevState.myRecipesActiveMeal ||
this.state.recipeSubmitted !== prevState.recipeSubmitted) {
        fetch("http://10.40.6.204:5000/recipes/get_personal_recipes/1?meal=" +
this.state.myRecipesActiveMeal)
        .then(res => res.json())
        .then(
          (result) => {
            this.setState({
              isLoaded: true,
              recipeList: result
            });
          },
          (error) => {
            this.setState({
              isLoaded: true,
              error
            });
          }
        )
      }
    }
    else {
      if (this.state.activeRecipes !== prevState.activeRecipes ||
this.state.commRecipesActiveMeal !== prevState.commRecipesActiveMeal ||
this.state.recipeSubmitted !== prevState.recipeSubmitted) {
        fetch("http://10.40.6.204:5000/recipes/get_community_recipes/" +
this.state.commRecipesActiveMeal)
        .then(res => res.json())
        .then(
          (result) => {
```

```
            this.setState({
              isLoaded: true,
              recipeList: result
            });
          },
          (error) => {
            this.setState({
              isLoaded: true,
              error
            });
          }
        )
      }
    }

  }

  divClick() {
    this.setState({showContextMenu: false, showContextMenuComm: false});
  }

  searchFunction() {
    var input, filter, ol, li, a, i, txtValue;
    input = document.getElementById("search-input");
    filter = input.value.toUpperCase();
    ol = document.getElementById("entry-list");
    li = ol.getElementsByTagName("li");
    for (i = 0; i < li.length; i++) {
        a = li[i].getElementsByTagName("label")[0];
        txtValue = a.textContent || a.innerText;
        if (txtValue.toUpperCase().indexOf(filter) > -1) {
            li[i].style.display = "";
        } else {
            li[i].style.display = "none";
        }
    }
  }

  render() {
    if (this.state.showEditRecipe) {
      return (
        <EditRecipeScreen
          recipe_data={this.state.editRecipeData}
          onBackButton={this.editRecipeBackButtonClicked}
          onSubmitButton={this.handleEditRecipeSubmitButton}
          activeRecipeSection={this.state.activeRecipes}
          activeMeal={this.state.activeRecipes == "My Recipes" ?
this.state.myRecipesActiveMeal : this.state.commRecipesActiveMeal }
        />
      )
    }
    else if (this.state.showViewRecipe) {
```

```
      return (
        <ViewRecipe
          recipeData={this.state.editRecipeData}
          onBackButton={this.handleViewRecipeBackButton}
          activeSection={this.state.activeRecipes}
        />
      )
    }

    return (
      <>
        { this.props.showCreateRecipe ?
            <CreateRecipeScreen
              onBackButton={this.handleCreateRecipeBackButton}
              activeRecipeSection={this.state.activeRecipes}
              activeMeal={this.state.activeRecipes == "My Recipes" ?
this.state.myRecipesActiveMeal : this.state.commRecipesActiveMeal}
              onSubmitButton={this.handleCreateRecipeSubmitButtonClick}
            />
          :
            <div label="Recipes">
              <Banner title = {this.state.title}/>
              <RecipesBanner onClick={() => this.handleCreateRecipeClick()}
searchFunction={() => this.searchFunction()}/>
              <MealTabs classVariant="recipes-" onClickTab={(tab) =>
this.handleRecipesTabChange(tab)} selectedTab={this.state.activeRecipes}>
                <div label="My Recipes">
                  {this.state.showContextMenu ?
                    <div
                      className="context-div"
                      onClick={() => this.divClick()}>
                        <MyRecipeContextMenu
                          recipe_id={this.state.contextMenuID}
                          recipeData={this.state.contextMenuData}
                          activeRecipesTab={this.state.activeRecipes}
                          handleDeleteRecipe={() => this.handleDeleteRecipe()}
                          handleEditRecipe={this.handleEditRecipe}
                          handleShareRecipe={() => this.handleShareRecipe()}
                          handleViewRecipe={this.handleViewRecipe}
                        />
                    </div> : null}
                  <MealTabs
                    classVariant="recipes-meals-"
                    onClickTab={(tab) => this.handleMyRecipesMealTabChange(tab)}
                    selectedTab={this.state.myRecipesActiveMeal}
                  >
                    <div label="Breakfast">
                      {
                        this.state.isLoaded ?
                          this.state.recipeList.length != 0 ?
                            <RecipeList
                              recipeList={this.state.recipeList}
```

```
                            activeRecipesTab={this.state.activeRecipes}
                            handleDeleteRecipe={() => this.handleDeleteRecipe()}
                            handleEditRecipe={this.handleEditRecipe}
                            handleShareRecipe={() => this.handleShareRecipe()}
                            handleViewRecipe={this.handleViewRecipe}
                            onContextClick={this.handleContextClickMyRecipe}
                          />
                        :
                          <EmptyRecipeList onClick={() =>
this.handleCreateRecipeClick()} />
                      :
                        <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                    }
                  </div>
                  <div label="Lunch">
                  {
                    this.state.isLoaded ?
                      this.state.recipeList.length != 0 ?
                        <RecipeList
                          recipeList={this.state.recipeList}
                          activeRecipesTab={this.state.activeRecipes}
                          handleDeleteRecipe={() => this.handleDeleteRecipe()}
                          handleEditRecipe={this.handleEditRecipe}
                          handleShareRecipe={() => this.handleShareRecipe()}
                          handleViewRecipe={this.handleViewRecipe}
                          onContextClick={this.handleContextClickMyRecipe}
                        />
                      :
                        <EmptyRecipeList onClick={() => this.handleCreateRecipeClick()}
/>
                    :
                      <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                  }
                  </div>
                  <div label="Dinner">
                    {
                      this.state.isLoaded ?
                        this.state.recipeList.length != 0 ?
                          <RecipeList
                            recipeList={this.state.recipeList}
                            activeRecipesTab={this.state.activeRecipes}
                            handleDeleteRecipe={() => this.handleDeleteRecipe()}
                            handleEditRecipe={this.handleEditRecipe}
                            handleShareRecipe={() => this.handleShareRecipe()}
                            handleViewRecipe={this.handleViewRecipe}
                            onContextClick={this.handleContextClickMyRecipe}
                          />
                        :
                          <EmptyRecipeList onClick={() =>
this.handleCreateRecipeClick()} />
```

```
                        :
                          <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                        }
                      </div>
                      <div label="Snacks">
                        {
                          this.state.isLoaded ?
                            this.state.recipeList.length != 0 ?
                              <RecipeList
                                recipeList={this.state.recipeList}
                                activeRecipesTab={this.state.activeRecipes}
                                handleDeleteRecipe={() => this.handleDeleteRecipe()}
                                handleEditRecipe={this.handleEditRecipe}
                                handleShareRecipe={() => this.handleShareRecipe()}
                                handleViewRecipe={this.handleViewRecipe}
                                onContextClick={this.handleContextClickMyRecipe}
                              />
                            :
                              <EmptyRecipeList onClick={() =>
this.handleCreateRecipeClick()} />
                          :
                            <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                        }
                      </div>
                      <div label="Drinks">
                        {
                          this.state.isLoaded ?
                            this.state.recipeList.length != 0 ?
                              <RecipeList
                                recipeList={this.state.recipeList}
                                activeRecipesTab={this.state.activeRecipes}
                                handleDeleteRecipe={() => this.handleDeleteRecipe()}
                                handleEditRecipe={this.handleEditRecipe}
                                handleShareRecipe={() => this.handleShareRecipe()}
                                handleViewRecipe={this.handleViewRecipe}
                                onContextClick={this.handleContextClickMyRecipe}
                              />
                            :
                              <EmptyRecipeList onClick={() =>
this.handleCreateRecipeClick()} />
                          :
                            <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                        }
                      </div>
                    </MealTabs>
                  </div>
                  <div label="Community Recipes">
                    {this.state.showContextMenuComm ?
                      <div
```

```
                    className="context-div"
                    onClick={() => this.divClick()}>
                      <CommRecipeContextMenu
                        recipe_id={this.state.contextMenuIDComm}
                        recipeData={this.state.contextMenuDataComm}
                        activeRecipesTab={this.state.activeRecipes}
                        handleEditRecipe={this.handleEditRecipe}
                        handleViewRecipe={this.handleViewRecipe}
                      />
                    </div> : null}
                  <MealTabs
                    classVariant="recipes-meals-"
                    onClickTab={(tab) => this.handleCommRecipesMealTabChange(tab)}
                    selectedTab={this.state.commRecipesActiveMeal}
                  >
                    <div label="Breakfast">
                      {
                        this.state.isLoaded ?
                          this.state.recipeList.length == 0 ?
                            <EmptyRecipeList
                              onClick={() => this.handleCreateRecipeClick()}
                              handleViewRecipe={this.handleViewRecipe}
                            />
                          :
                            <RecipeList recipeList={this.state.recipeList}
activeRecipesTab={this.state.activeRecipes} onContextClick={this.handleContextClickComm}/>
                        :
                          <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                      }
                    </div>
                    <div label="Lunch">
                      {
                        this.state.isLoaded ?
                          this.state.recipeList.length == 0 ?
                            <EmptyRecipeList
                              onClick={() => this.handleCreateRecipeClick()}
                              handleViewRecipe={this.handleViewRecipe}
                            />
                          :
                            <RecipeList recipeList={this.state.recipeList}
activeRecipesTab={this.state.activeRecipes} onContextClick={this.handleContextClickComm}/>
                        :
                          <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                      }
                    </div>
                    <div label="Dinner">
                      {
                        this.state.isLoaded ?
                          this.state.recipeList.length == 0 ?
                            <EmptyRecipeList
```

```
                            onClick={() => this.handleCreateRecipeClick()}
                            handleViewRecipe={this.handleViewRecipe}
                          />
                        :
                          <RecipeList recipeList={this.state.recipeList}
activeRecipesTab={this.state.activeRecipes} onContextClick={this.handleContextClickComm}/>
                      :
                        <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                    }
                  </div>
                  <div label="Snacks">
                    {
                      this.state.isLoaded ?
                        this.state.recipeList.length == 0 ?
                          <EmptyRecipeList
                            onClick={() => this.handleCreateRecipeClick()}
                            handleViewRecipe={this.handleViewRecipe}
                          />
                        :
                          <RecipeList recipeList={this.state.recipeList}
activeRecipesTab={this.state.activeRecipes} onContextClick={this.handleContextClickComm}/>
                      :
                        <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                    }
                  </div>
                  <div label="Drinks">
                    {
                      this.state.isLoaded ?
                        this.state.recipeList.length == 0 ?
                          <EmptyRecipeList
                            onClick={() => this.handleCreateRecipeClick()}
                            handleViewRecipe={this.handleViewRecipe}
                          />
                        :
                          <RecipeList recipeList={this.state.recipeList}
activeRecipesTab={this.state.activeRecipes} onContextClick={this.handleContextClickComm}/>
                      :
                        <ReactLoading type={"cylon"} height={'20%'} width={'20%'}
color={"#3a85ff"} className={"diary-loading"}/>
                    }
                  </div>
                </MealTabs>
              </div>
            </MealTabs>
          </div>
          }
          <ToastContainer />
        </>

    );
```

```
  }
}

export default Recipes;
```

RecipeList.js

```
import React, {Component} from 'react';
import RecipeListItem from './RecipeListItem';

class RecipeList extends React.Component {
  render() {
    return (
      <div className="entries">
        <ol className="entry-list" id="entry-list">
        {this.props.recipeList.map((recipe, i) => {
          return (
            <RecipeListItem
              key={i}
              recipe_id={recipe['recipe_id']}
              label={recipe['name']}
              image={"http://10.40.6.204:5000/uploads/" + recipe['image_path']}
              time={recipe['time']}
              date={recipe['date']}
              cookingTimeHrs={recipe['cooking_time_hrs']}
              cookingTimeMin={recipe['cooking_time_min']}
              activeRecipesTab={this.props.activeRecipesTab}
              handleDeleteRecipe={() => this.props.handleDeleteRecipe()}
              handleShareRecipe={() => this.props.handleShareRecipe()}
              handleEditRecipe={this.props.handleEditRecipe}
              handleViewRecipe={this.props.handleViewRecipe}
              recipeData={recipe}
              onContextClick={this.props.onContextClick}
            />
          );
        })}
        </ol>

      </div>
    );
  }
}

export default RecipeList;
```

RecipeListItem.js

```
import React, {Component} from 'react';

class RecipeListItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
```

```
      popupVisible: false
    };
    this.handleContextClick = this.handleContextClick.bind(this);
  };

  handleContextClick() {
    this.props.onContextClick(this.props.recipe_id, this.props.recipeData);

  }

  render() {
    let className = 'entry-list-item';
    return (
      <li className={className} onClick={() => this.handleContextClick()} >
        <div className="entry-div">
          <img src={this.props.image} className="entry-image"/>
          <div className="recipe-info-div">
            <label className="recipe-name-header">{this.props.label}</label><br/>
            <img src="/calendar_icon.png" className="clock-icon" />
            <label className="recipe-date-
label">{this.props.recipeData['date']}</label><br/>
            <img src="/clock_icon.png" className="clock-icon" />
            <label className="recipe-time-label">{this.props.cookingTimeHrs + "hrs " +
this.props.cookingTimeMin + "min"}</label><br/>
            {
              this.props.recipeData['shared'] == 1 ?
                <>
                  <img
                    src="/share-tick.png"
                    className="share-icon"
                  />
                  <label className="recipe-share-label">Shared with community</label>
                </>
              :
                null
            }
          </div>
        </div>
      </li>
    );
  }
}

export default RecipeListItem;
```

RecipesBanner.js

```
import React, {Component} from 'react';

class RecipesBanner extends React.Component {
  render() {
```

```
    return (
      <div className="PDbannerDiv">
        <input
          type="text" className="search-box"
          onKeyUp={this.props.searchFunction}
          id="search-input"
          placeholder="Search Recipes.."
          autoComplete="off" >
        </input>
        <button className="PDbannerAddButton" type="button"
onClick={this.props.onClick}>Create Recipe</button>
      </div>
    );
  }
}

export default RecipesBanner;
```

MyRecipeContextMenu.js

```
import React, {Component} from 'react';

class MyRecipeContextMenu extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      isDeleted: false,
      error: null,
    };
  }

  handleDeleteButton() {
    if (window.confirm("Are you sure you want to delete this recipe?")) {
      fetch("http://10.40.6.204:5000/recipes/delete_recipe/" + this.props.recipe_id,
{method: 'post'})
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isDeleted: true,
            response: result
          });
        },
        (error) => {
          this.setState({
            isDeleted: false,
            error
          });
        }
      )
      alert("Recipe deleted!");
      this.props.handleDeleteRecipe();
```

```
    }
  }

  handleShareButton() {
    if (window.confirm("Are you sure you want to share your recipe with the community?"))
{
      fetch("http://10.40.6.204:5000/recipes/share_recipe/" + this.props.recipe_id +
"?share=1", {method: 'post'})
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isDeleted: true,
            response: result
          });
        },
        (error) => {
          this.setState({
            isDeleted: false,
            error
          });
        }
      )
      alert("Recipe shared with community!");
    }
    this.props.handleShareRecipe();
  }

  handleStopShareButton() {
    if (window.confirm("Are you sure you want to stop sharing your recipe with the
community?")) {
      fetch("http://10.40.6.204:5000/recipes/share_recipe/" + this.props.recipe_id +
"?share=0", {method: 'post'})
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isDeleted: true,
            response: result
          });
        },
        (error) => {
          this.setState({
            isDeleted: false,
            error
          });
        }
      )
      alert("You have stopped sharing your recipe with the community!");
    }
    this.props.recipeData['shared'] = 0;
    this.props.handleShareRecipe();
```

```
  }

  handleEditButton() {
    this.props.handleEditRecipe(this.props.recipeData);
  }

  handleViewButton() {
    this.props.handleViewRecipe(this.props.recipeData);
  }

  render() {
    let myDropId = "myDropdown" + this.props.recipe_id;
    return (
        <div id={myDropId} className="dropdown-content2">
            <button className="context-button" onClick={() =>
this.handleViewButton()}>View</button><br/>
            <button className="context-button" onClick={() =>
this.handleEditButton()}>Edit</button><br/>
            {
              this.props.recipeData['shared'] == 0 ?
                <button
                  className="context-button"
                  onClick={() => this.handleShareButton()}>
                  Share
                </button>
              :
                <button
                  className="context-button"
                  onClick={() => this.handleStopShareButton()}>
                  Stop Sharing
                </button>}
            <br/>
            <button className="context-button" onClick={() =>
this.handleDeleteButton()}>Delete</button>
        </div>
    );
  }
}

export default MyRecipeContextMenu;
```

CommRecipeContextMenu.js

```
import React, {Component} from 'react';

class CommRecipeContextMenu extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      isDeleted: false,
      error: null,
```

```
    };
  }

  handleViewButton() {
    this.props.handleViewRecipe(this.props.recipeData);
  }

  render() {
    let myDropId = "myDropdown" + this.props.entry_id;
    return (
      <div id={myDropId} className="dropdown-content2">
          <button className="context-button" onClick={() =>
this.handleViewButton()}>View</button>
      </div>
    );
  }
}

export default CommRecipeContextMenu;
```

EmptyRecipeList.js

```
import React, {Component} from 'react';

class EmptyRecipeList extends React.Component {
  render() {
    return (
      <div className="empty-entry-list-div">
        <button className="no-entries-button" onClick={this.props.onClick}>+</button>
        <p className="empty-recipe-list-p2">Click to add a new recipe!</p>
      </div>
    );
  }
}

export default EmptyRecipeList;
```

CreateRecipeScreen.js

```
import React, {Component} from 'react';
import AddEntryBanner from './AddEntryBanner'

class CreateRecipeScreen extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      file: '/image_placeholder.png',
      title: "",
      description: "",
      meal: this.props.activeMeal,
      cook_time_hours: "",
      cook_time_mins: "",
      serves: "",
```

```
      ingredients_inputs: [{"ingredient": "", "quantity": ""}],
      step_inputs: [{"step": ""}],
      share_with_community: this.props.activeRecipeSection == "Community Recipes" ? true :
false,
      datetime: new Date().toISOString(),
      activeRecipeSection: this.props.activeRecipeSection,
      form_submitted: false,

    };

    this.handlePicChange = this.handlePicChange.bind(this);
    this.handleTitleChange = this.handleTitleChange.bind(this);
    this.handleDescChange = this.handleDescChange.bind(this);
    this.handleMealChange = this.handleMealChange.bind(this);
    this.handleHoursChange = this.handleHoursChange.bind(this);
    this.handleMinsChange = this.handleMinsChange.bind(this);
    this.handleServesChange = this.handleServesChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
    this.handleAddIngredient = this.handleAddIngredient.bind(this);
    this.handleMinusIngredient = this.handleMinusIngredient.bind(this);
    this.handleAddStep = this.handleAddStep.bind(this);
    this.handleMinusStep = this.handleMinusStep.bind(this);
    this.handleShareCheckbox = this.handleShareCheckbox.bind(this);
    this.handleBackButton = this.handleBackButton.bind(this);
  }

  handlePicChange(event) {
    if(event.target.files[0]) {
      this.setState({
        file: URL.createObjectURL(event.target.files[0])
      })
    }
  }

  handleTitleChange(event) {
    this.setState({title: event.target.value});
  }

  handleDescChange(event) {
    this.setState({description: event.target.value});
  }

  handleMealChange(event) {
    this.setState({meal: event.target.value});
  }

  handleAddIngredient() {
    let arr = this.state.ingredients_inputs.concat([{"ingredient": "", "quantity": ""}]);
    this.setState({ingredients_inputs: arr});
  }

  handleMinusIngredient() {
```

```
  if(this.state.ingredients_inputs.length != 1) {
    this.state.ingredients_inputs.pop();
    this.forceUpdate();
  }

}

handleAddStep() {
  let arr = this.state.step_inputs.concat([{"step": ""}]);
  this.setState({step_inputs: arr});
}

handleMinusStep() {
  if(this.state.step_inputs.length != 1) {
    this.state.step_inputs.pop();
    this.forceUpdate();
  }

}

handleShareCheckbox() {
  this.setState({share_with_community: !this.state.share_with_community});
}

handleHoursChange(event) {
  this.setState({cook_time_hours: event.target.value});
}

handleMinsChange(event) {
  this.setState({cook_time_mins: event.target.value});
}

handleServesChange(event) {
  this.setState({serves: event.target.value});
}

handleStepInputChange(i, event) {
  this.state.step_inputs[i].step = event.target.value;
  this.forceUpdate();
}

handleIngredientInputChange(i, event) {
  this.state.ingredients_inputs[i].ingredient = event.target.value;
  this.forceUpdate();
}

handleQuantityChange(event, id) {
  this.state.ingredients_inputs[id].quantity = event.target.value;
  this.forceUpdate();
}

handleSubmit(event) {
```

```
    event.preventDefault();
    if(this.state.file == '/image_placeholder.png') {
      alert("Please select a photo!");
    }
    else {
      const data = new FormData(event.target);
      let param = JSON.stringify(this.state).replace(/&/g, "%26");
      fetch("http://10.40.6.204:5000/recipes/add_recipe/1?data=" + param, {
          method: 'post',
          body: data,
        })
        .then(
          (result) => {
            this.setState({
              form_submitted: true,
            });
          },
          (error) => {
            this.setState({
              form_submitted: true,
              error
            });
          }
        )
      this.props.onSubmitButton(this.state.meal, this.state.activeRecipeSection);
    }

  }

  handleBackButton() {
    this.props.onBackButton(this.state.meal, this.state.activeRecipeSection);
  }

  render() {
    return (
        <>
        <AddEntryBanner onBackButton={this.handleBackButton} title="Create Recipe"/>
        <img src={this.state.file} id="add-recipe-img"/>
        <form className="add-entry-form" name="add-entry-form"
onSubmit={this.handleSubmit} encType="multipart/form-data">
          <input type="file" name="file" id="file" className="add-entry-pic-input"
accept="image/*" onChange={this.handlePicChange}/>
          <label htmlFor="file">Choose a Picture</label>
          <br/><br/>
          <label htmlFor="title" id="add-entry-title-input-label" className="add-entry-
labels">Title</label>
          <br/>
          <input className="add-entry-title-input input-style"
                 maxLength="40" name="title" type="text"
                 value={this.state.title} onChange={this.handleTitleChange}
                 placeholder="Enter title here..." required autoComplete="off"
          />
```

```
          <br/><br/>
          <label htmlFor="desc" id="add-entry-desc-input-label" className="add-entry-
labels">Description</label>
          <br/>
          <textarea className="add-entry-desc-input input-style"
                    name="desc" type="text" form="add-entry-form"
                    value={this.state.decription} onChange={this.handleDescChange}
                    placeholder="Enter description here (optional)...">
          </textarea>
          <br/><br/>
          <label htmlFor="desc" id="add-entry-meal-input-label" className="add-entry-
labels">Meal</label>
          <br/>
          <select id="add-entry-meal-input" className="input-style"
value={this.state.meal} onChange={this.handleMealChange} required>
            <option value = "Breakfast">Breakfast</option>
            <option value = "Lunch">Lunch</option>
            <option value = "Dinner">Dinner</option>
            <option value = "Snacks">Snacks</option>
            <option value = "Drinks">Drinks</option>
          </select>
          <br/><br/>
          <label className="create-recipe-cook-time" htmlFor="cook_time">Preparation
Time</label>
          <label className="create-recipe-serves-label" htmlFor="serves">Serves</label>
          <br/>
          <input className="create-recipe-cook-time input-style"
                 type="number" min="0" max="100" placeholder="HH"
                 name="cook_time" onChange={this.handleHoursChange}
          />
          <input className="create-recipe-cook-time input-style"
                 type="number" min="0" max="59" placeholder="MM"
                 name="cook_time" onChange={this.handleMinsChange}
          />
          <input className="create-recipe-cook-time input-style"
                 type="number" min="1" max="20" placeholder="Serves.."
                 name="serves" onChange={this.handleServesChange}
          />
          <br/><br/>
          <label id="create-recipe-ingred-step-label">Ingredients</label>
          <button type="button" id="add-ingred-step-button"
onClick={this.handleAddIngredient}>+</button>
          <button type="button" id="minus-ingred-step-button"
onClick={this.handleMinusIngredient}>-</button><br/>
          {
            this.state.ingredients_inputs.map((input, i) => {
              return (
                <>
                  <input defaultValue={input.ingredient}
                         className="ingred-input input-style"
                         placeholder={"Ingredient " + (i+1) + ".."}
                         onChange={(event) => this.handleIngredientInputChange(i, event)}
```

```
                        required
                    />
                    <input defaultValue={input.quantity}
                        className="ingred-qnty-input input-style"
                        placeholder="Quantity"
                        onChange={(event) => this.handleQuantityChange(event, i)}
                    />
                    <br/><br/>
                  </>

                );

              })
            }
            <label id="create-recipe-ingred-step-label">Recipe Steps</label>
            <button type="button" id="add-ingred-step-button"
onClick={this.handleAddStep}>+</button>
            <button type="button" id="minus-ingred-step-button"
onClick={this.handleMinusStep}>-</button><br/>
            {
              this.state.step_inputs.map((input, i) => {
                return (
                  <>
                    <input defaultValue={input.step}
                        className="step-input input-style"
                        placeholder={"Step " + (i+1) + ".."}
                        onChange={(event) => this.handleStepInputChange(i, event)}
                        required
                    />
                    <br/><br/>
                  </>

                );

              })
            }
            <input id="share-checkbox" type="checkbox"
                name="share-with-community"
                defaultChecked={this.state.share_with_community}
                onChange={() => this.handleShareCheckbox()}
            /> Share my recipe with the community
            <input className="add-entry-pic-submit-btn"
                type="submit" value="Save Recipe"
            />
          </form>

          </>
      );
    }
}

export default CreateRecipeScreen;
```

EditRecipeScreen.js

```
import React, {Component} from 'react';
import AddEntryBanner from './AddEntryBanner'

class EditRecipeScreen extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      file: "http://10.40.6.204:5000/uploads/" + this.props.recipe_data['image_path'],
      title: this.props.recipe_data['name'],
      description: this.props.recipe_data['description'],
      meal: this.props.activeMeal,
      cook_time_hours: parseInt(this.props.recipe_data['cooking_time_hrs']),
      cook_time_mins: parseInt(this.props.recipe_data['cooking_time_min']),
      serves: parseInt(this.props.recipe_data['serves']),
      ingredients_inputs: [{"ingredient": "", "quantity": ""}],
      step_inputs: [{"step": ""}],
      deleted_steps: [],
      deleted_ingredients: [],
      share_with_community: !!+this.props.recipe_data['shared'],
      datetime: new Date().toISOString(),
      activeRecipeSection: this.props.activeRecipeSection,
      isIngredientsLoaded: false,
      isStepsLoaded: false,

    };

    this.handlePicChange = this.handlePicChange.bind(this);
    this.handleTitleChange = this.handleTitleChange.bind(this);
    this.handleDescChange = this.handleDescChange.bind(this);
    this.handleMealChange = this.handleMealChange.bind(this);
    this.handleHoursChange = this.handleHoursChange.bind(this);
    this.handleMinsChange = this.handleMinsChange.bind(this);
    this.handleServesChange = this.handleServesChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
    this.handleAddIngredient = this.handleAddIngredient.bind(this);
    this.handleMinusIngredient = this.handleMinusIngredient.bind(this);
    this.handleAddStep = this.handleAddStep.bind(this);
    this.handleMinusStep = this.handleMinusStep.bind(this);
    this.handleShareCheckbox = this.handleShareCheckbox.bind(this);
    this.handleBackButton = this.handleBackButton.bind(this);
  }

  handlePicChange(event) {
    if(event.target.files[0]) {
      this.setState({
        file: URL.createObjectURL(event.target.files[0])
      })
    }
  }
```

```
  handleTitleChange(event) {
    this.setState({title: event.target.value});
  }

  handleDescChange(event) {
    this.setState({description: event.target.value});
  }

  handleMealChange(event) {
    this.setState({meal: event.target.value});
  }

  handleAddIngredient() {
    let arr = this.state.ingredients_inputs.concat([{"ingredient": "", "quantity": ""}]);
    this.setState({ingredients_inputs: arr});
  }

  handleMinusIngredient() {
    if(this.state.ingredients_inputs.length != 1) {
      let arr =
this.state.deleted_ingredients.concat(this.state.ingredients_inputs.pop());
      this.setState({deleted_ingredients: arr});
      this.forceUpdate();
    }

  }

  handleAddStep() {
    let arr = this.state.step_inputs.concat([{"step": ""}]);
    this.setState({step_inputs: arr});
  }

  handleMinusStep() {
    if(this.state.step_inputs.length != 1) {
      let arr = this.state.deleted_steps.concat(this.state.step_inputs.pop());
      this.setState({deleted_steps: arr});
      this.forceUpdate();
    }

  }

  handleShareCheckbox() {
    this.setState({share_with_community: !this.state.share_with_community});
  }

  handleHoursChange(event) {
    this.setState({cook_time_hours: event.target.value});
  }

  handleMinsChange(event) {
    this.setState({cook_time_mins: event.target.value});
```

```
  }

  handleServesChange(event) {
    this.setState({serves: event.target.value});
  }

  handleStepInputChange(i, event) {
    this.state.step_inputs[i].step = event.target.value;
    this.forceUpdate();
  }

  handleIngredientInputChange(i, event) {
    this.state.ingredients_inputs[i].ingredient = event.target.value;
    this.forceUpdate();
  }

  handleQuantityChange(event, id) {
    this.state.ingredients_inputs[id].quantity = event.target.value;
    this.forceUpdate();
  }

  handleSubmit(event) {
    event.preventDefault();
    const data = new FormData(event.target);
    let param = JSON.stringify(this.state).replace(/&/g, "%26");
    fetch("http://10.40.6.204:5000/recipes/update_recipe/" +
this.props.recipe_data['recipe_id'] + "?data=" + param, {
        method: 'post',
        body: data
      })
      .then(
        (result) => {
          this.setState({
            form_submitted: true,
          });
        },
        (error) => {
          this.setState({
            form_submitted: true,
            error
          });
        }
      )
    this.props.onSubmitButton(this.state.meal, this.state.activeRecipeSection);
  }

  handleBackButton() {
    this.props.onBackButton(this.state.meal, this.state.activeRecipeSection);
  }

  componentDidMount() {
```

```
    fetch("http://10.40.6.204:5000/recipes/get_recipe_steps_ingredients/" +
this.props.recipe_data['recipe_id'])
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isIngredientsLoaded: true,
            ingredients_inputs: result['ingredients'],
            step_inputs: result['steps']
          });
        },
        (error) => {
          this.setState({
            isIngredientsLoaded: true,
            error
          });
        }
      )

  }

  render() {
    return (
        <>
        <AddEntryBanner onBackButton={this.handleBackButton} title="Edit Recipe"/>
        <img src={this.state.file} id="add-recipe-img"/>
        <form className="add-entry-form" name="add-entry-form"
onSubmit={this.handleSubmit} encType="multipart/form-data">
          <input
            type="file" name="file" id="file"
            className="add-entry-pic-input"
            accept="image/*" onChange={this.handlePicChange}
          />
          <label htmlFor="file">Choose a new Picture</label>
          <br/><br/>
          <label htmlFor="title" id="add-entry-title-input-label" className="add-entry-
labels">Title</label>
          <br/>
          <input
            className="add-entry-title-input input-style"
            maxLength="40" name="title" type="text"
            value={this.state.title} onChange={this.handleTitleChange}
            placeholder="Enter title here..." required autoComplete="off"
          />
          <br/><br/>
          <label htmlFor="desc" id="add-entry-desc-input-label" className="add-entry-
labels">Description</label>
          <br/>
          <textarea
            className="add-entry-desc-input input-style"
            name="desc" type="text" form="add-entry-form"
            onChange={this.handleDescChange}
```

```
            placeholder="Enter description here (optional)...">
              {this.props.recipe_data['description']}
          </textarea>
          <br/><br/>
          <label htmlFor="desc" id="add-entry-meal-input-label" className="add-entry-
labels">Meal</label>
          <br/>
          <select id="add-entry-meal-input" className="input-style"
value={this.state.meal} onChange={this.handleMealChange} required>
            <option value="Breakfast">Breakfast</option>
            <option value="Lunch">Lunch</option>
            <option value="Dinner">Dinner</option>
            <option value="Snack">Snack</option>
            <option value="Drink">Drink</option>
          </select>
          <br/><br/>
          <label className="create-recipe-cook-time" htmlFor="cook_time">Preparation
Time</label>
          <label className="create-recipe-serves-label" htmlFor="serves">Serves</label>
          <br/>
          <input
            className="create-recipe-cook-time input-style"
            type="number" min="0" max="100" placeholder="HH"
            name="cook_time" onChange={this.handleHoursChange}
            value={this.state.cook_time_hours} required
          />
          <input
            className="create-recipe-cook-time input-style"
            type="number" placeholder="MM" min="0" max="59"
            name="cook_time" onChange={this.handleMinsChange}
            value={this.state.cook_time_mins} required
          />
          <input
            className="create-recipe-cook-time input-style"
            type="number" min="1" max="20" placeholder="Serves.."
value={this.state.serves}
            name="serves" onChange={this.handleServesChange}
          />
          <br/><br/>
          <label id="create-recipe-ingred-step-label">Ingredients</label>
          <button
            type="button" id="add-ingred-step-button"
            onClick={this.handleAddIngredient}>
            +
          </button>
          <button
            type="button" id="minus-ingred-step-button"
            onClick={this.handleMinusIngredient}>
            -
          </button><br/>
          {
            this.state.ingredients_inputs.map((input, i) => {
```

```
            return (
              <>
                <input
                  defaultValue={input.ingredient}
                  className="ingred-input input-style"
                  placeholder={"Ingredient " + (i+1) + ".."}
                  onChange={(event) => this.handleIngredientInputChange(i, event)}
                  required autoComplete="off"
                />
                <input
                  defaultValue={input.quantity}
                  className="ingred-qnty-input input-style"
                  placeholder="Quantity"
                  onChange={(event) => this.handleQuantityChange(event, i)}
                />
                <br/><br/>
              </>

            );

          })
        }
        <label id="create-recipe-ingred-step-label">Recipe Steps</label>
        <button
          type="button" id="add-ingred-step-button"
          onClick={this.handleAddStep}>
          +
        </button>
        <button
          type="button" id="minus-ingred-step-button"
          onClick={this.handleMinusStep}>
          -
        </button><br/>
        {
          this.state.step_inputs.map((input, i) => {
            return (
              <>
                <input
                  defaultValue={input.step}
                  className="step-input input-style"
                  placeholder={"Step " + (i+1) + ".."}
                  onChange={(event) => this.handleStepInputChange(i, event)}
                  required autoComplete="off"
                />
                <br/><br/>
              </>

            );

          })
        }
        <input
```

```
            id="share-checkbox" type="checkbox"
            name="share-with-community"
            defaultChecked={this.state.share_with_community}
            onChange={() => this.handleShareCheckbox()}
          /> Share my recipe with the community
          <input className="add-entry-pic-submit-btn" type="submit" value="Update
Recipe"/>
        </form>

        </>
    );
  }
}

export default EditRecipeScreen;
```

ViewRecipe.js

```
import React, {Component} from 'react';
import AddEntryBanner from './AddEntryBanner'

class ViewRecipe extends React.Component {
  constructor(props) {
    super(props);
    let meal_list = ["Breakfast", "Lunch", "Dinner", "Snacks", "Drinks"]
    this.state = {
      meal: meal_list[this.props.recipeData['meal']],
      ingredients_steps: [],
      isLoaded: false,
      date: new Date(this.props.recipeData['date']),
    };

    this.handleBackButton = this.handleBackButton.bind(this);
  }

  handleBackButton() {
    this.props.onBackButton(this.state.meal, this.props.activeSection);
  }

  componentDidMount() {
    fetch("http://10.40.6.204:5000/recipes/get_recipe_steps_ingredients/" +
this.props.recipeData['recipe_id'])
      .then(res => res.json())
      .then(
        (result) => {
          console.log(result);
          this.setState({
            isLoaded: true,
            ingredients_steps: result
          });
        },
        (error) => {
```

```
          this.setState({
            isLoaded: true,
            error
          });
        }
      )
  }

  render() {
    return (
        <>
          <AddEntryBanner onBackButton={this.handleBackButton} title="View Recipe"/>
          <div className="view-recipe-div">
          <h1 className="view-entry-title">{this.props.recipeData['name']}</h1>
            <img src={"http://10.40.6.204:5000/uploads/" +
this.props.recipeData['image_path']} id="view-entry-img"/>
            <p className="view-entry-meal">{this.state.meal}</p>
            <p className="view-entry-serves">{"Serves: " +
this.props.recipeData['serves']}</p>
            <p className="view-entry-date">{this.state.date.toDateString()}</p>
            <div className="view-entry-desc-div">
              <label>Description:</label>
              <p>{this.props.recipeData['description'] != "" ?
this.props.recipeData['description'] : "No Description Given."}</p>
            </div>
            <h3>Ingredients</h3>
            <hr className="style"/>
              {
                this.state.isLoaded ?
                this.state.ingredients_steps['ingredients'].map((ingredient, i) => {
                  return (
                    <>
                      <div className="view-recipe-ing-div">
                        <label>Ingredient {i + 1}: </label>{ingredient['ingredient']}
                        <br/>
                        <label>Quantity {i + 1}: </label>{ingredient['quantity']}
                      </div>
                    </>

                  );

              }) : <label>No Ingredients</label>
            }
            <h3>Steps</h3>
            <hr className="style"/>
            {
                this.state.isLoaded ?
                this.state.ingredients_steps['steps'].map((step, i) => {
                  return (
                    <>
                      <div className="view-recipe-ing-div"><label>{i + 1}.
</label>{step['step']}</div>
```

```
                    </>

                  );

              }) : null
            }
          </div>
        </>
    );
  }
}


export default ViewRecipe;
```

Tabs.js

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';

import Tab from './Tab';

class Tabs extends Component {
  static propTypes = {
    children: PropTypes.instanceOf(Array).isRequired,
  }

  constructor(props) {
    super(props);

    this.state = {
      activeTab: this.props.children[0].props.label,
    };
  }

  onClickTabItem = (tab) => {
    this.setState({ activeTab: tab });
    this.props.onClickTab(tab);
  }

  render() {
    const {
      onClickTabItem,
      props: {
        children,
      },
      state: {
        activeTab,
      }
    } = this;

    return (
```

```
      <div className={this.props.classVariant + "tabs"}>
        <ol className={this.props.classVariant +"tab-list"}>
          {children.map((child) => {
            const { label } = child.props;

            return (
              <Tab
                activeTab={activeTab}
                key={label}
                label={label}
                onClick={onClickTabItem}
                classVariant={this.props.classVariant}
              />
            );
          })}
        </ol>
        <div className={this.props.classVariant + "tab-content"}>
          {children.map((child) => {
            if (child.props.label !== activeTab) return undefined;
            return child.props.children;
          })}
        </div>
      </div>
    );
  }
}

export default Tabs;
```

Tab.js

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';

class Tab extends Component {
  static propTypes = {
    activeTab: PropTypes.string.isRequired,
    label: PropTypes.string.isRequired,
    onClick: PropTypes.func.isRequired,
  };

  onClick = () => {
    const { label, onClick } = this.props;
    onClick(label);
  }

  render() {
    const {
      onClick,
      props: {
        activeTab,
        label,
```

```
      },
    } = this;

    let className = this.props.classVariant + 'tab-list-item';

    if (activeTab === label) {
      className = className + " " + this.props.classVariant + 'tab-list-active';
    }

    return (
      <li
        className={className}
        onClick={onClick}
      >
        {label}
      </li>
    );
  }
}

export default Tab;
```

MealTabs.js

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import Tab from './Tab';

class MealTabs extends Component {
  static propTypes = {
    children: PropTypes.instanceOf(Array).isRequired,
  }

  constructor(props) {
    super(props);

    this.state = {
      activeTab: this.props.selectedTab,
    };
  }

  onClickTabItem = (tab) => {
    this.setState({ activeTab: tab });
    this.props.onClickTab(tab);
  }

  render() {
    const {
      onClickTabItem,
      props: {
        children,
      },
```

```
      state: {
        activeTab,
      }
    } = this;

    return (
      <div className={this.props.classVariant + "tabs"}>
        <ol className={this.props.classVariant + "tab-list"}>
          {children.map((child) => {
            const { label } = child.props;

            return (
              <Tab
                activeTab={this.props.selectedTab}
                key={label}
                label={label}
                onClick={onClickTabItem}
                classVariant={this.props.classVariant}
              />
            );
          })}
        </ol>
        <div className={this.props.classVariant + "tab-content"}>
          {children.map((child) => {
            if (child.props.label !== activeTab) return undefined;
            return child.props.children;
          })}
        </div>
      </div>
    );
  }
}

export default MealTabs;
```

CSS Code

Index.css

```
.bannerDiv {
    width: 100%;
    background-color: #3a85ff;
    position: fixed;
    top: 0;
    left: 0;
    z-index: 999;
    height: 8%;
    display: block;
}

.bannerHead {
```

```
    font-size: 25px;
    font-family: Arial, Helvetica, sans-serif;
    color: white;
    padding-left: 10px;
    font-weight: 700;
}

.PDbannerDiv {
    display: block;
    position: fixed;
    top: 8%;
    left: 0;
    z-index: 997;
    height: 8%;
    width: 100%;
    background-color: white;

}

.PDbannerButton {
    float: right;
    top: 22%;
    right: 4%;
    padding: 7px 20px;
    border-radius:3px;
    border:1px solid white;
    position: relative;
    background-color: #3a85ff;
    color: #ffffff;
    font-weight: 700;
}

.PDbannerAddButton {
    float: right;
    top: 21%;
    right: 2%;
    padding: 7px 20px;
    border-radius:3px;
    border:1px solid white;
    position: relative;
    background-color: #3a85ff;
    color: #ffffff;
    font-weight: 700;
}

::-webkit-scrollbar {
    width: 0px;
    background: transparent; /* make scrollbar transparent */
}

.infinite-calendar {
    margin: auto;
```

```
    width: 250px;
    border: 1px solid gray;
    padding: 3px;
    position: relative;
    top: 190px;
    z-index: 999;
}

.DDbannerDiv {
    width: 100%;
    background-color: #3a85ff;
    position: fixed;
    top: 16%;
    left: 0;
    z-index: 999;
    height: 8%;
    display: block;
    z-index: auto;
}

.DDbannerHead {
    font-size: 25px;
    font-family: Arial, Helvetica, sans-serif;
    color: white;
    padding-left: 10px;
    text-align: center;
    font-weight: 600;

}

.entry-list {
    padding-left: 0;
    position: fixed;
    list-style: none;
    top: 30%;
    left: 0;
    z-index: 1;
    height: 64%;
    width: 100%;
    overflow-y: scroll;
}

.entry-list-item {
    border-bottom: 1px solid black;
    padding: 0.2rem;
    margin: 0.5rem;
    height: 110px;
}

.entry-image {
    width: 100px;
    height: 100px;
```

```
    margin-bottom: 8px;
    float: left;
    background-repeat: no-repeat;
    background-position: center center;
    background-size: cover;
    border: solid 3px #fff;
    box-shadow: 0 0 3px 2px #3a85ff;
    border-radius: 50%;
}

.entry-div {
   height: 100%;
}

.entry-name-header {
    vertical-align: 50%;
    padding-left: 0.3rem;
    font-family: Arial, Helvetica, sans-serif;
    color: gray;
    margin-left: 5px;
    white-space: nowrap;
    width: 18em;
    overflow: hidden;
    text-overflow: ellipsis;
}

.recipe-name-header {
    vertical-align: 50%;
    padding-left: 0.3rem;
    font-family: Arial, Helvetica, sans-serif;
    color: gray;
    margin-left: 5px;
    white-space: nowrap;
    width: 29rem;
    text-overflow: ellipsis;
}

.entry-time-label {
    font-family: Arial, Helvetica, sans-serif;
    color: gray;
    vertical-align: 100%;
    margin-left: 5px;
    font-size: 14px;
    font-style: italic;
    font-weight: 400;
}

.AEbannerDiv {
    width: 100%;
    background-color: #3a85ff;
    position: fixed;
    top: 0%;
```

```
    left: 0;
    z-index: 999;
    height: 8%;
    display: block;
    font-size: 25px;
    font-family: Arial, Helvetica, sans-serif;
    padding-left: 10px;
}

.AEbannerButton {
    background:none;
    border:none;
    margin:0;
    padding:0;
    cursor: pointer;
    position: absolute;
    color: white;
    font-size: 35px;
    font-weight: bold;

}

.AEbannerHead {
    font-size: 25px;
    font-family: Arial, Helvetica, sans-serif;
    color: white;
    text-align: center;
    z-index: 999;
}

.input-style {
    border: 1px solid #ccc;
    border-radius: 4px;
    padding: 12px 20px;
    box-sizing: border-box;
}

@media only screen and (min-width: 768px) {
    .AEbannerButton {
        background:none;
        border:none;
        margin:0;
        padding:0;
        cursor: pointer;
        float: left;
        color: white;
        font-size: 50px;
        font-weight: bold;

    }

    .entry-name-header {
```

```
        vertical-align: 100%;
        padding-left: 0.3rem;
        font-family: Arial, Helvetica, sans-serif;
        color: gray;
        margin-left: 5px;

        white-space: unset;
        width: unset;
        overflow: unset;
        text-overflow: unset;
    }

    .PDbannerAddButton {
        top: 23%;
    }

    .entry-image {
        width: 100px;
        height: 100px;
        background-repeat: no-repeat;
        background-position: center center;
        background-size: cover;
        border: solid 3px #fff;
        border-radius: 50%;
    }

    .entry-list {
        top: 28.5%;
        height: 67%;
    }
}

.add-entry-form {
    width: 100%;
    /*background-color: #25ffa4;*/
    position: relative;
    top: 49.5%;
    left: 0;
    z-index: auto;
    height: 100%;
    display: block;
    overflow-y: scroll;
    padding-bottom: 15%;
}

#add-entry-img {
    display: block;
    margin-left: auto;
    margin-right: auto;
    width: 100%;
    margin-top: 2%;
    height: 40vh;
```

```
    border: 1px solid gray;
    border-radius: 4px;
    box-shadow: 0 0 3px 2px grey;
}

#add-recipe-img {
    display: block;
    margin-left: auto;
    margin-right: auto;
    width: 100%;
    margin-top: 14%;
    height: 40vh;
    border: 1px solid gray;
    border-radius: 4px;
    box-shadow: 0 0 3px 2px grey;
}

.add-entry-pic-input {
    width: 0.1px;
    height: 0.1px;
    opacity: 0;
    overflow: hidden;
    position: absolute;
    z-index: -1;
}

.add-entry-pic-input + label {
    font-family: Arial, Helvetica, sans-serif;
    float: right;
    font-size: 1em;
    font-weight: 700;
    color: white;
    background-color: #3a85ff;
    border: white;
    border-style: solid;
    border-width: 1px;
    padding: 5px;
    margin-top: 5px;
    margin-right: 5px;
}

.add-entry-pic-input:focus + label,
.add-entry-pic-input + label:hover {
    border: #3a85ff;
    border-style: solid;
    border-width: 1px;
    background-color: white;
    color: #3a85ff;
}


#add-entry-meal-input {
```

```
    margin-left: 1%;
    width: 98.5%;
    height: 5%;

}

#add-entry-title-input-label {
    font-family: Arial, Helvetica, sans-serif;
    margin-left: 1%;
}

.add-entry-title-input {
    margin-left: 1%;
    width: 97%;
    height: 5%;

}

.add-entry-desc-input {
    margin-left: 1%;
    width: 97%;
    height: 5%;

}

.add-entry-labels {
    font-family: Arial, Helvetica, sans-serif;
    margin-left: 1%;
}

.add-entry-pic-submit-btn {
    float: right;
    margin-right: 1%;
}

#add-entry-timestamp-input {
    margin-left: 1%;
    width: 98.5%;
    height: 5%;
}

.add-entry-pic-submit-btn {
    font-family: Arial, Helvetica, sans-serif;
    float: right;
    font-size: 1em;
    font-weight: 700;
    color: white;
    background-color: #3a85ff;
    border: #3a85ff;
    border-style: solid;
    border-width: 1px;
    padding: 5px;
```

```
    margin-top: 5px;
}

.add-entry-pic-submit-btn:hover {
    border: #3a85ff;
    border-style: solid;
    border-width: 1px;
    background-color: white;
    color: #3a85ff;

}

.create-recipe-form {
    width: 100%;
    /*background-color: #25ffa4;*/
    position: relative;
    top: 59%;
    left: 0;
    height: 100%;
    display: block;
    overflow-y: scroll;
    padding-bottom: 10%;
}

@media only screen and (min-width: 768px) {
    #add-entry-img {
        display: block;
        margin-left: auto;
        margin-right: auto;
        width: 60%;
        margin-top: 1%;
        height: 68vh;
    }

    #add-entry-img {
        display: block;
        margin-left: auto;
        margin-right: auto;
        width: 60%;
        margin-top: 5%;
        height: 68vh;
    }

    #add-recipe-img {
        display: block;
        margin-left: auto;
        margin-right: auto;
        width: 60%;
        margin-top: 5%;
        height: 68vh;
    }
```

```css
.add-entry-form {
    width: 100%;
    position: relative;
    top: 70%;
    left: 0;
    z-index: auto;
    height: 100%;
    display: block;
    overflow-y: scroll;
    padding: 3%;

}

.create-recipe-form {
    width: 100%;
    position: relative;
    top: 70%;
    left: 0;
    z-index: auto;
    height: 100%;
    display: block;
    overflow-y: scroll;
    padding-left: 5%;
    padding-right: 5%;

}

#add-entry-meal-input {
    margin-left: 1%;
    width: 97.2%;
}

#add-entry-timestamp-input {
    margin-left: 1%;
    width: 97.25%;
}

.add-entry-desc-input {
    margin-left: 1%;
    width: 97%;
    height: 25rem;
}

.add-entry-pic-submit-btn {
    float: right;
    margin-right: 1.5%;
}

.add-entry-pic-input + label {
    margin-right: 1.8%;
}
```

```
}

.create-recipe-cook-time {
    margin-left: 1%;
}

.create-recipe-serves-label {
    margin-left: 22.1%;
}

#create-recipe-ingred-step-label {
    margin-left: 1%;
}

#add-ingred-step-button {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 1em;
    font-weight: 700;
    color: white;
    background-color: #3a85ff;
    border: #3a85ff;
    border-style: solid;
    border-width: 2px;
    border-radius: 50%;
    margin-left: 5px;
}

#minus-ingred-step-button {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 1em;
    font-weight: 700;
    color: white;
    background-color: red;
    border: red;
    border-style: solid;
    border-width: 2px;
    border-radius: 50%;
    margin-left: 5px;
}

.step-input {
    margin-left: 1%;
    width: 98%;
}

.ingred-input {
    margin-left: 1%;
    width: 72%;
}

.ingred-qnty-input {
    width: 25%;
```

```
    margin-left: 1%;
}

#share-checkbox {
    margin-left: 1%;
}

.diary-loading {
    margin-top: 100%;
    margin-left: 41%;
}

.empty-entry-list-div {
    margin-top: 80%;
    margin-left: 28%;
    width: 44%;
}
.no-entries-button {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 9em;
    height: 180px;
    width: 180px;
    font-weight: 700;
    color: white;
    background-color: #3a85ff;
    border: #3a85ff;
    border-style: solid;
    /* border-width: 2px; */
    border-radius: 50%;
    /* padding: 50px; */
}

.empty-entry-list-p2 {
    font-family: Arial, Helvetica, sans-serif;
    margin-left: 9%;
    color: gray;
    font-weight: 700;
    margin-top: 5px;
}

.empty-recipe-list-p2 {
    font-family: Arial, Helvetica, sans-serif;
    margin-left: 8%;
    color: gray;
    font-weight: 700;
    margin-top: 5px;
}

.view-entry-div {
    width: 100%;
    overflow-y: scroll;
    height: 685px;
```

```
    padding-bottom: 5px;
}

#view-entry-img {
    margin-top: 14%;
    width: 90%;
    height: 300px;
    display: block;
    margin-left: auto;
    margin-right: auto;
    border: 1px solid gray;
    border-radius: 4px;
    box-shadow: 0 0 3px 2px gray;
    padding: 5px;

}

.view-entry-title {
    position: relative;
    top: 7%;
    font-size: 22px;
    font-weight: 700;
    font-family: Arial, Helvetica, sans-serif;
    margin-left: 5%;
}

.view-entry-meal {
    border: 1px solid gray;
    border-radius: 4px;
    box-shadow: 0 0 3px 2px gray;
    padding: 5px;
    width: 29%;
    text-align: center;
    float: left;
    margin-left: 5%;
    margin-top: 2%;
    color: gray;
    font-weight: 600;
}

.view-entry-serves {
    border: 1px solid gray;
    border-radius: 4px;
    box-shadow: 0 0 3px 2px grey;
    padding: 5px;
    width: 29%;
    text-align: center;
    float: left;
    color: gray;
    font-weight: 600;
    margin-top: 2%;
    margin-left: 1%;
```

```
}

.view-entry-date {
    border: 1px solid gray;
    border-radius: 4px;
    box-shadow: 0 0 3px 2px gray;
    padding: 5px;
    width: 30%;
    text-align: center;
    float: right;
    margin-right: 5%;
    margin-top: 2%;
    color: gray;
    font-weight: 600;
}

.view-entry-desc-div {
    border: 1px solid gray;
    border-radius: 4px;
    padding: 10px;
    box-shadow: 0 0 3px 2px gray;
    width: 90%;
    margin-left: auto;
    margin-right: auto;
    white-space: pre-wrap;
    white-space: -moz-pre-wrap;
    white-space: -pre-wrap;
    white-space: -o-pre-wrap;
    word-wrap: break-word;
    margin-top: 12%;
    color: gray;
    font-weight: 500;
}

@media only screen and (min-width: 768px) {
    .empty-entry-list-div {
        margin-top: 20%;
        margin-left: 41%;
        width: 50%;
    }

    .create-recipe-serves-label {
        margin-left: 8.1%;
    }

    .diary-loading {
        margin-top: 20%;
        margin-left: 41%;
    }

    .no-entries-button {
        font-family: Arial, Helvetica, sans-serif;
```

```
        font-size: 10em;
        height: 300px;
        width: 300px;
        font-weight: 700;
        color: white;
        background-color: #3a85ff;
        border: #3a85ff;
        border-style: solid;
        border-radius: 50%;
        padding: 50px;
    }

    .empty-entry-list-p2 {
        font-family: Arial, Helvetica, sans-serif;
        margin-left: 9%;
        color: gray;
        font-weight: 700;
        margin-top: 5px;
    }

    #view-entry-img {
        margin-top: 5%;
        width: 70%;
        height: 700px;
        display: block;
        margin-left: auto;
        margin-right: auto;
        border: 1px solid gray;
        border-radius: 4px;
        padding: 5px;

    }

    .view-entry-div {
        width: 100%;
        overflow-y: scroll;
        height: 810px;
        padding-bottom: 5px;
    }

    .view-entry-title {
        position: relative;
        top: 70px;
        font-size: 22px;
        font-weight: 700;
        font-family: Arial, Helvetica, sans-serif;
        margin-left: 15%;
    }

    .view-entry-meal {
        border: 1px solid gray;
        border-radius: 4px;
```

```
        box-shadow: 0 0 3px 2px gray;
        padding: 5px;
        width: 9%;
        text-align: center;
        float: left;
        margin-left: 15%;
        margin-top: 1%;
        color: gray;
        font-weight: 600;
    }

    .view-entry-serves {
        border: 1px solid gray;
        border-radius: 4px;
        box-shadow: 0 0 3px 2px grey;
        padding: 5px;
        width: 9%;
        text-align: center;
        float: left;
        color: gray;
        font-weight: 600;
        margin-top: 1%;
        margin-left: 21%;
    }

    .view-entry-date {
        border: 1px solid gray;
        border-radius: 4px;
        box-shadow: 0 0 3px 2px gray;
        padding: 5px;
        width: 9%;
        text-align: center;
        float: right;
        margin-right: 15%;
        margin-top: 1%;
        color: gray;
        font-weight: 600;
    }

    .view-entry-desc-div {
        border: 1px solid gray;
        border-radius: 4px;
        padding: 10px;
        box-shadow: 0 0 3px 2px gray;
        width: 70%;
        margin-left: auto;
        margin-right: auto;
        white-space: pre-wrap;
        white-space: -moz-pre-wrap;
        white-space: -pre-wrap;
        white-space: -o-pre-wrap;
        word-wrap: break-word;
```

```
        margin-top: 4%;
        color: gray;
        font-weight: 500;
    }
}

.clock-icon {
    width: 15px;
    height: 15px;
    vertical-align: 90%;
    margin-left: 9px;
}

.share-icon {
    width: 20px;
    height: 23px;
    vertical-align: 70%;
    margin-left: 7px;
}


.recipe-time-label {
    font-family: Arial, Helvetica, sans-serif;
    color: gray;
    vertical-align: 100%;
    margin-left: 3px;
    font-size: 14px;
    font-style: italic;
    font-weight: 400;
}

.recipe-share-label {
    font-family: Arial, Helvetica, sans-serif;
    color: gray;
    vertical-align: 100%;
    font-size: 14px;
    font-style: italic;
    font-weight: 400;
}

.recipe-date-label {
    font-family: Arial, Helvetica, sans-serif;
    color: gray;
    margin-left: 3px;
    font-size: 14px;
    font-style: italic;
    font-weight: 400;
    vertical-align: 100%;
}

.recipe-info-div {
    width: 70%;
```

```
    line-height: 8pt;
    float: left;
}

.view-recipe-ing-div {
    border: 1px solid gray;
    border-radius: 4px;
    padding: 10px;
    box-shadow: 0 0 3px 2px gray;
    width: 90%;
    margin-left: auto;
    margin-right: auto;
    white-space: pre-wrap;
    white-space: -moz-pre-wrap;
    white-space: -pre-wrap;
    white-space: -o-pre-wrap;
    word-wrap: break-word;
    margin-top: 1%;
    color: gray;


}

h3 {
    margin-left: 5%;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: 700;
}

hr.style {
    border: 0;
    height: 2px;
    background: #333;
    background-image: -webkit-linear-gradient(left, #ccc, #333, #ccc);
    background-image: -moz-linear-gradient(left, #ccc, #333, #ccc);
    background-image: -ms-linear-gradient(left, #ccc, #333, #ccc);
    background-image: -o-linear-gradient(left, #ccc, #333, #ccc);
    margin-left: 5%;
    margin-right: 5%;
}

.view-recipe-div {
    width: 100%;
    overflow-y: scroll;
    height: 690px;
    padding-bottom: 5px;
}

.dropdown-content2 {
    margin: auto;
    width: 50%;
    border: 1px solid gray;
```

```
    padding: 3px;
    position: relative;
    top: 300px;
    z-index: 999;
    color: black;

}

.context-div {
    background-color: black;
    position:fixed;
    background: rgba(0, 0, 0, 0.7);
    width: 100%;
    height: 900px;
    z-index: 999;
}

.context-button {
    width: 100%;
    height: 5rem;
}

.search-box {
    width: 150px;
    box-sizing: border-box;
    border: 2px solid #ccc;
    border-radius: 4px;
    font-size: 16px;
    background-color: white;
    /* background-image: url(http://10.40.6.204:3000/searchicon.png); */
    background-position: 10px 10px;
    background-repeat: no-repeat;
    padding: 7px;
    -webkit-transition: width 0.4s ease-in-out;
    transition: width 0.4s ease-in-out;
    margin-left: 1%;
    margin-top: 2%;
}

  /* When the input field gets focus, change its width to 100% */
.search-box:focus {
    width: 60%;
}

.quick-add {
    display: flex;
    flex-wrap: nowrap;
    overflow-x: auto;

}

.quick-add-entry {
```

```
    flex: 0 0 auto;
}

.quick-add-img {
    width: 10%;
    padding: 4px;
    height: 130px;
    width: 200px;
    overflow: hidden;
}

.quick-add-title {
    width: 200px;
    overflow: hidden;
    color: gray;
    margin-left: 2%;
    text-overflow: ellipsis;
}

#quick-add-label {
    margin-top: 17%;
    margin-left: 0.4%;
}

@media only screen and (min-width: 768px) {
    .recipe-info-div {
        width: 20%;
        line-height: 9pt;
        float: left;
    }

    .view-recipe-div {
        width: 100%;
        overflow-y: scroll;
        height: 810px;
        padding-bottom: 5px;
    }

    .view-recipe-ing-div {
        border: 1px solid gray;
        border-radius: 4px;
        padding: 10px;
        box-shadow: 0 0 3px 2px gray;
        width: 70%;
        margin-left: auto;
        margin-right: auto;
        white-space: pre-wrap;
        white-space: -moz-pre-wrap;
        white-space: -pre-wrap;
        white-space: -o-pre-wrap;
        word-wrap: break-word;
        margin-top: 1%;
```

```
        color: gray;


    }

    h3 {
        margin-left: 15%;
        font-family: Arial, Helvetica, sans-serif;
        font-weight: 700;
    }

    hr.style {
        margin-left: 15%;
        margin-right: 15%;
    }

    .dropdown-content2 {
        margin: auto;
        width: 10%;
        border: 1px solid gray;
        padding: 3px;
        position: relative;
        top: 300px;
        z-index: 999;
        color: black;

    }

    .search-box {
        width: 150px;
        box-sizing: border-box;
        border: 2px solid #ccc;
        border-radius: 4px;
        font-size: 16px;
        background-color: white;
        /*background-image: url('searchicon.png');*/
        background-position: 10px 10px;
        background-repeat: no-repeat;
        padding: 10px;
        -webkit-transition: width 0.4s ease-in-out;
        transition: width 0.4s ease-in-out;
        margin-left: 1%;
        margin-top: 0.5%;
    }

      /* When the input field gets focus, change its width to 100% */
    .search-box:focus {
        width: 88%;
    }

    .quick-add {
        width: 100%;
```

```
        height: 155px;
        overflow-x: scroll;
        overflow: -webkit-paged-x;
    }

    .quick-add-entry {
        width: 10%;
        float: left;
        border: 1px solid gray;
        padding: 2px;
        overflow: hidden;

    }

    .quick-add-img {
        width: 100%;
        padding: 4px;
        height: 105px;
    }

    .quick-add-title {
        width: 200px;
        overflow: hidden;
        color: gray;
    }

    #quick-add-label {
        margin-top: 4.2%;
        margin-left: 0.2%;
    }

}

.toast {
    background-color: #3a85ff;
    color: white;
}
```

Tabs.css

```
.meals-tab-list {
    border-bottom: 1px solid #ccc;
    background-color: #3a85ff;
    display: block;
    position: fixed;
    top: 21.9%;
    left: 0;
    z-index: 997;
    height: 7%;
    width: 100%;
    z-index: 997;
    padding-left: 0;
```

```
}

.meals-tab-list-item {
    float: left;
    list-style: none;
    padding: 1.5rem 0.5rem 0rem 1rem;
    width: 20%;
    text-align: center;
    color: #e9e9e9;
    height: 100%;
    font-family: Arial, Helvetica, sans-serif;
}

.meals-tab-list-active {
    border-bottom: 4px solid #ccc;
    text-decoration: underline;
    color: white;
    font-weight: 700;
    border-radius: 9px;
}

.section-tab-list {
    border-bottom: 1px solid #ccc;
    background-color: #3a85ff;
    padding-left: 0;
    display: block;
    position: fixed;
    top: 94%;
    left: 0;
    z-index: 997;
    height: 7%;
    width: 100%;
}

.section-tab-list-item {
    float: left;
    list-style: none;
    padding: 1rem 0.5rem 0rem 1rem;
    width: 50%;
    text-align: center;
    color: #e9e9e9;
    height: 69%;
    font-family: Arial, Helvetica, sans-serif;
}

.section-tab-list-active {
    background-color: #3a85ff;
    border-top: 4px solid #ccc;
    text-decoration: underline;
    color: white;
    font-weight: 700;
    border-radius: 9px;
```

```
}

.recipes-tab-list {
    border-bottom: 1px solid #ccc;
    background-color: #3a85ff;
    padding-left: 0;
    display: block;
    position: fixed;
    top: 16%;
    left: 0;
    z-index: 997;
    height: 7%;
    width: 100%;
}

.recipes-tab-list-item {
    float: left;
    list-style: none;
    padding: 1.5rem 0.5rem 0rem 1rem;
    width: 50%;
    text-align: center;
    color: #e9e9e9;
    height: 100%;
    font-family: Arial, Helvetica, sans-serif;
}

.recipes-tab-list-active {
    background-color: #3a85ff;
    border-bottom: 4px solid #ccc;
    color: white;
    text-decoration: underline;
    font-weight: 700;
    border-radius: 9px;
}

.recipes-meals-tab-list {
    border-bottom: 1px solid #ccc;
    background-color: #3a85ff;
    padding-left: 0;
    display: block;
    position: fixed;
    top: 23%;
    left: 0;
    z-index: 997;
    height: 7%;
    width: 100%;
}

.recipes-meals-tab-list-item {
    float: left;
    list-style: none;
    padding: 1.5rem 0.5rem 0rem 1rem;
```

```
    width: 20%;
    text-align: center;
    color: #e9e9e9;
    height: 100%;
    font-family: Arial, Helvetica, sans-serif;
}

.recipes-meals-tab-list-active {
    background-color: #3a85ff;
    border-bottom: 4px solid #ccc;
    text-decoration: underline;
    color: white;
    font-weight: 700;
    border-radius: 9px;
}

/* For Desktop */
@media only screen and (min-width: 768px) {
    .meals-tab-list {
        border-bottom: 1px solid #ccc;
        background-color: #3a85ff;
        padding-left: 0;
        display: block;
        position: fixed;
        top: 22.5%;
        left: 0;
        z-index: 995;
        height: 6%;
        width: 100%;
    }

    .meals-tab-list-item {
        float: left;
        list-style: none;
        padding: 1rem 0.5rem 0rem 1rem;
        width: 20%;
        text-align: center;
        color: white;
        height: 100%;
        font-family: Arial, Helvetica, sans-serif;
    }

    .meals-tab-list-active {
        border-bottom: 5px solid #ccc;
        text-decoration: underline;
        color: white;
        font-weight: 700;
        border-radius: 9px;
    }

    .section-tab-list {
        border-bottom: 1px solid #ccc;
```

```
        background-color: #3a85ff;
        padding-left: 0;
        display: block;
        position: fixed;
        top: 94.5%;
        left: 0;
        z-index: 997;
        height: 6%;
        width: 100%;
    }

    .section-tab-list-item {
        float: left;
        list-style: none;
        padding-top: 1em;
        width: 50%;
        text-align: center;
        color: #e9e9e9;
        height: 69%;
        font-family: Arial, Helvetica, sans-serif;
    }

    .section-tab-list-active {
        background-color: #3a85ff;
        border-top: 5px solid #ccc;
        text-decoration: underline;
        color: white;
        font-weight: 700;
        border-radius: 9px;
    }

    .recipes-meals-tab-list {
        border-bottom: 1px solid #ccc;
        background-color: #3a85ff;
        display: block;
        position: fixed;
        top: 21.9%;
        left: 0;
        z-index: 997;
        height: 6.5%;
        width: 100%;
        z-index: 997;
    }

    .recipes-meals-tab-list-item {
        float: left;
        list-style: none;
        padding: 2rem 0.5rem 0rem 0rem;
        width: 20%;
        text-align: center;
        color: #e9e9e9;
        height: 100%;
```

```
        font-family: Arial, Helvetica, sans-serif;
    }

    .recipes-meals-tab-list-active {
        background-color: #3a85ff;
        border-bottom: 5px solid #ccc;
        text-decoration: underline;
        color: white;
    }

    .recipes-tab-list {
        border-bottom: 1px solid #ccc;
        background-color: #3a85ff;
        padding-left: 0;
        display: block;
        position: fixed;
        top: 16%;
        left: 0;
        z-index: 997;
        height: 6%;
        width: 100%;
    }

    .recipes-tab-list-item {
        float: left;
        list-style: none;
        padding: 2rem 0.5rem 0rem 1rem;
        width: 50%;
        text-align: center;
        color: #e9e9e9;
        height: 100%;
        font-family: Arial, Helvetica, sans-serif;

    }

    .recipes-tab-list-active {
        background-color: #3a85ff;
        border-bottom: 5px solid #ccc;
        text-decoration: underline;
        color: white;
    }
}
```

Python/Flask API Source Code

food_diary.py

```
import os
from flask import Flask, jsonify, request, send_from_directory, Blueprint
```

```
from werkzeug import secure_filename
import requests
import json
import mysql.connector
import datetime
import dateutil.parser
import config

food_diary = Blueprint('food_diary', __name__)

meal_list = ["Breakfast", "Lunch", "Dinner", "Snacks", "Drinks"]

#####################################################
#Food Diary#
#####################################################

@food_diary.route('/diary/get_diary_entries/<user_id>', methods=['GET'])
def get_diary_entries(user_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    meal = meal_list.index(request.args["meal"])
    date = request.args['date']
    query = "SELECT * FROM diaryentry WHERE user_id = %s AND meal = %s AND entry_date =
'%s';"
    cursor.execute(query % (user_id, meal, date))
    entries = []
    for (entry_id, title, description, meal, image, entry_date, time, iso_datetime,
user_id) in cursor:
      entries.append({"entry_id": entry_id, "title": title,
                      "description": description, "date": entry_date,
                      "time": str(time), "iso_datetime": iso_datetime, "meal": meal,
"image": image,
                      "user_id": user_id})
    connection.close()
    cursor.close()
    entries.reverse()
    return jsonify(entries)
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


@food_diary.route('/diary/add_diary_entry/<user_id>', methods=['POST'])
def add_diary_entry(user_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    query = """INSERT INTO diaryentry (entry_id, title, description,
               meal, image_path, entry_date, time, iso_datetime, user_id)
               VALUES (NULL, %s, %s, %s, %s, %s, %s, %s, %s)"""
    title = request.args["title"]
```

```
    desc_data = request.args["desc"]
    desc_dict = json.loads(desc_data)
    desc = desc_dict['description']
    date = request.args["date"]
    dateObj = dateutil.parser.parse(date)
    meal = meal_list.index(request.args["meal"])
    user_id = 1
    image = request.args.get("image")
    if image != None:
      cursor.execute(query, (title, desc, meal, request.args['image'],
dateObj.strftime("%Y-%#m-%#d"),
                             dateObj.strftime("%H:%M:%S"), date, user_id))
    else:
      f = request.files['file']
      f.save(os.path.join("uploads", secure_filename(f.filename)))
      #The '#' in the date format removes the leading zero - Should be changed to '-' on
Linux!!
      cursor.execute(query, (title, desc, meal, f.filename, dateObj.strftime("%Y-%#m-
%#d"),
                             dateObj.strftime("%H:%M:%S"), date, user_id))
    connection.commit()
    connection.close()
    cursor.close()
    return "{'status': 'success'}"
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


@food_diary.route('/uploads/<path:filename>', methods=['GET'])
def download_file(filename):
  return send_from_directory("uploads",
                             filename, as_attachment=True)


@food_diary.route('/diary/update_diary_entry/<diary_entry_id>', methods=['POST'])
def update_diary_entry(diary_entry_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    title = request.args["title"]
    desc = request.args["desc"]
    date = request.args["date"]
    dateObj = dateutil.parser.parse(date)
    meal = meal_list.index(request.args["meal"])
    f = request.files['file']
    if f.filename != "":
      f.save(os.path.join("uploads", secure_filename(f.filename)))
      query = """UPDATE diaryentry SET title = %s, description = %s,
                 meal = %s, image_path = %s, entry_date = %s, time = %s,
                 iso_datetime = %s WHERE entry_id = %s;"""
```

```
      cursor.execute(query, (title, desc, meal, f.filename, dateObj.strftime("%Y-%#m-
%#d"),
                             dateObj.strftime("%H:%M:%S"), date, diary_entry_id))
    else:
      query = """UPDATE diaryentry SET title = %s, description = %s,
                 meal = %s, entry_date = %s, time = %s, iso_datetime = %s WHERE entry_id =
%s;"""
      cursor.execute(query, (title, desc, meal, dateObj.strftime("%Y-%#m-%#d"),
                             dateObj.strftime("%H:%M:%S"), date, diary_entry_id))
    connection.commit()
    connection.close()
    cursor.close()
    return "{'status': 'success'}"
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


#Delete_diary_entry(diary_entry_id, user_id)
@food_diary.route('/diary/delete_diary_entry/<diary_entry_id>', methods=['POST'])
def delete_diary_entry(diary_entry_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    query = "DELETE FROM diaryentry WHERE entry_id = %s;"
    cursor.execute(query % (diary_entry_id))
    connection.commit()
    connection.close()
    cursor.close()
    return "{'status': 'success'}"
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


@food_diary.route('/diary/get_quick_add_entries/<meal>', methods=['GET'])
def get_quick_add_entries(meal):
  connection = config.db_manager.get_connection()
  cursor = connection.cursor()
  the_meal = meal_list.index(meal)
  query = """SELECT * FROM (SELECT * FROM diaryentry
             WHERE meal = %s ORDER BY entry_id DESC LIMIT 20)Var1 GROUP BY title
             ORDER BY entry_id ASC;"""
  cursor.execute(query % (the_meal))
  entries = []
  for (entry_id, title, description, meal, image, entry_date, time, iso_datetime, user_id)
in cursor:
    entries.append({"entry_id": entry_id, "title": title,
                    "description": description, "date": entry_date,
                    "time": str(time), "iso_datetime": iso_datetime, "meal": meal,
"image": image,
                    "user_id": user_id})
```

```
  connection.close()
  cursor.close()
  entries.reverse()
  return jsonify(entries)

#####################################################
#Recipes#
#####################################################

@food_diary.route('/recipes/add_recipe/<user_id>', methods=['POST'])
def add_recipe(user_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    insert_query = """INSERT INTO recipe (recipe_id, name, description,
                      cooking_time_hrs, cooking_time_min, shared, meal,
                      date, time, image_path, serves, user_id) VALUES
                      (NULL, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);"""
    recipe_data = request.args['data']
    data_dict = json.loads(recipe_data)
    title = data_dict['title']
    description = data_dict['description']
    cooktime_hrs = data_dict['cook_time_hours']
    cooktime_min = data_dict['cook_time_mins']
    shared = int(data_dict['share_with_community'])
    serves = data_dict['serves']
    meal = meal_list.index(data_dict['meal'])
    date = data_dict['datetime']
    dateObj = dateutil.parser.parse(date)
    #!!!!!!!!!!##
    user_id = 1#!
    ##!!!!!!!!!##
    f = request.files['file']
    f.save(os.path.join("uploads", secure_filename(f.filename)))
    cursor.execute(insert_query, (title, description, cooktime_hrs, cooktime_min,
                                  shared, meal, dateObj.strftime("%Y-%#m-%#d"),
                                  dateObj.strftime("%H:%M:%S"), f.filename, serves,
user_id))
    connection.commit()
    id_query = """SELECT recipe_id FROM recipe WHERE name = %s
                  AND description = %s AND date = %s AND time = %s;"""
    cursor.execute(id_query, (title, description, dateObj.strftime("%Y-%#m-%#d"),
                              dateObj.strftime("%H:%M:%S")))
    result = cursor.fetchone()
    recipe_id = result[0]
    ingredient_insert_query = """INSERT INTO ingredient (ingredient_id, name, quantity,
recipe_id) VALUES (NULL, %s, %s, %s);"""
    for ingredient in data_dict['ingredients_inputs']:
      cursor.execute(ingredient_insert_query, (ingredient['ingredient'],
ingredient['quantity'], recipe_id))
      connection.commit()
```

```
    step_insert_query = "INSERT INTO recipestep (step_id, step_number, content, recipe_id)
VALUES (NULL, %s, %s, %s);"
    i = 1
    for step in data_dict['step_inputs']:
      cursor.execute(step_insert_query, (i, step['step'], recipe_id))
      i += 1
      connection.commit()
    connection.close()
    cursor.close()
    return "{'status': 'success'}"
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


@food_diary.route('/recipes/get_personal_recipes/<user_id>', methods=['GET'])
def get_personal_recipes(user_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    meal = meal_list.index(request.args["meal"])
    query = "SELECT * FROM recipe WHERE user_id = %s AND meal = %s;"
    cursor.execute(query % (user_id, meal))
    recipes = []
    for (recipe_id, name, description, cooking_time_hrs, cooking_time_min, shared, meal,
date, time, image_path, serves, user_id) in cursor:
      recipes.append({"recipe_id": recipe_id, "name": name,
                      "description": description, "cooking_time_hrs": cooking_time_hrs,
                      "cooking_time_min": cooking_time_min, "shared": shared, "meal":
meal, "date": date,
                      "time": str(time), "image_path": image_path, "serves": serves,
"user_id": user_id})
    connection.close()
    cursor.close()
    recipes.reverse()
    return jsonify(recipes)
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


@food_diary.route('/recipes/get_recipe_steps_ingredients/<recipe_id>', methods=['GET'])
def get_recipe_steps_ingredients(recipe_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    query = "SELECT * from recipestep WHERE recipe_id = %s;"
    cursor.execute(query % (recipe_id))
    steps = []
    for (step_id, step_number, content, recipe_id) in cursor:
      steps.append({"step_id": step_id, "step_number": step_number,
                    "step": content, "recipe_id": recipe_id})
```

```
    query = "SELECT * from ingredient WHERE recipe_id = %s;"
    cursor.execute(query % (recipe_id))
    ingredients = []
    for (ingredient_id, name, quantity, recipe_id) in cursor:
      ingredients.append({"ingredient_id": ingredient_id, "ingredient": name,
                  "quantity": quantity, "recipe_id": recipe_id})
    connection.close()
    cursor.close()
    return jsonify({"steps": steps, "ingredients": ingredients})
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


#Get_community_recipes(title, meal, number_of_recipes, ingredients, my_recipes, rating)
@food_diary.route('/recipes/get_community_recipes/<meal>', methods=['GET'])
def get_community_recipes(meal):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    meal = meal_list.index(meal)
    query = "SELECT * FROM recipe WHERE meal = %s AND shared = 1;"
    cursor.execute(query % (meal))
    recipes = []
    for (recipe_id, name, description, cooking_time_hrs, cooking_time_min, shared, meal,
date, time, image_path, serves, user_id) in cursor:
      recipes.append({"recipe_id": recipe_id, "name": name,
                      "description": description, "cooking_time_hrs": cooking_time_hrs,
                      "cooking_time_min": cooking_time_min, "shared": shared, "meal":
meal, "date": date,
                      "time": str(time), "image_path": image_path, "serves": serves,
"user_id": user_id})
    connection.close()
    cursor.close()
    recipes.reverse()
    return jsonify(recipes)
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


@food_diary.route('/recipes/update_recipe/<recipe_id>', methods=['POST'])
def update_recipe(recipe_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    recipe_data = request.args['data']
    data_dict = json.loads(recipe_data)
    name = data_dict['title']
    description = data_dict['description']
    cooktime_hrs = data_dict['cook_time_hours']
```

```
    cooktime_min = data_dict['cook_time_mins']
    serves = data_dict['serves']
    shared = int(data_dict['share_with_community'])
    meal = meal_list.index(data_dict['meal'])
    date = data_dict['datetime']
    dateObj = dateutil.parser.parse(date)

    f = request.files['file']
    if f.filename != "":
      f.save(os.path.join("uploads", secure_filename(f.filename)))
      query = """UPDATE recipe SET name = %s, description = %s,
                cooking_time_hrs = %s, cooking_time_min = %s,
                shared = %s, meal = %s, image_path = %s, serves = %s WHERE recipe_id =
%s"""
      cursor.execute(query, (name, description, cooktime_hrs, cooktime_min, shared, meal,
f.filename, serves, recipe_id))
      connection.commit()
    else:
      query = """UPDATE recipe SET name = %s, description = %s,
                cooking_time_hrs = %s, cooking_time_min = %s,
                shared = %s, meal = %s, serves = %s WHERE recipe_id = %s"""
      cursor.execute(query, (name, description, cooktime_hrs, cooktime_min, shared, meal,
serves, recipe_id))
      connection.commit()

    deleted_steps = data_dict['deleted_steps']
    for deleted_step in deleted_steps:
      if "recipe_id" in deleted_step:
        query = "DELETE FROM recipestep WHERE step_id = %s;"
        cursor.execute(query % deleted_step['step_id'])
    deleted_ingredients = data_dict['deleted_ingredients']
    for deleted_ingredient in deleted_ingredients:
      if "ingredient_id" in deleted_ingredient:
        query = "DELETE FROM ingredient WHERE ingredient_id = %s;"
        cursor.execute(query % deleted_ingredient['ingredient_id'])

    steps = data_dict['step_inputs']
    step_number = 1
    for step in steps:
      if "recipe_id" in step:
        query = "UPDATE recipestep SET step_number = %s, content = %s, recipe_id = %s
WHERE step_id = %s"
        cursor.execute(query, (step['step_number'], step['step'], recipe_id,
step['step_id']))
        step_number += 1
        connection.commit()
      else:
        query = "INSERT INTO recipestep (step_id, step_number, content, recipe_id) VALUES
(NULL, %s, %s, %s)"
        cursor.execute(query, (step_number, step['step'], recipe_id))
        step_number += 1
        connection.commit()
```

```
    ingredients = data_dict['ingredients_inputs']
    for ingredient in ingredients:
      if "ingredient_id" in ingredient:
        query = "UPDATE ingredient SET name = %s, quantity = %s WHERE ingredient_id = %s;"
        cursor.execute(query, (ingredient['ingredient'], ingredient['quantity'],
ingredient['ingredient_id']))
        connection.commit()
      else:
        query = "INSERT INTO ingredient (ingredient_id, name, quantity, recipe_id) VALUES
(NULL, %s, %s, %s)"
        cursor.execute(query, (ingredient['ingredient'], ingredient['quantity'],
recipe_id))
        connection.commit()
    connection.close()
    cursor.close()
    return "{'status': 'success'}"
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


@food_diary.route('/recipes/delete_recipe/<recipe_id>', methods=['POST'])
def delete_recipe(recipe_id):
  try:
    connection = config.db_manager.get_connection()
    cursor = connection.cursor()
    query = "DELETE FROM recipe WHERE recipe_id = %s;"
    cursor.execute(query % (recipe_id))
    connection.commit()
    query = "DELETE FROM ingredient WHERE recipe_id = %s;"
    cursor.execute(query % (recipe_id))
    connection.commit()
    query = "DELETE FROM recipestep WHERE recipe_id = %s;"
    cursor.execute(query % (recipe_id))
    connection.commit()
    connection.close()
    cursor.close()
    return "{'status': 'success'}"
  except Exception as e:
    error_msg = "{'error': " + str(e) + "}"
    return error_msg


#Share_recipe(recipe_id)
@food_diary.route('/recipes/share_recipe/<recipe_id>', methods=['POST'])
def share_recipe(recipe_id):
  connection = config.db_manager.get_connection()
  cursor = connection.cursor()
  if int(request.args['share']) == 1:
    query = "UPDATE recipe SET shared = 1 WHERE recipe_id = %s"
    cursor.execute(query % recipe_id)
```

```
    connection.commit()
  else:
    query = "UPDATE recipe SET shared = 0 WHERE recipe_id = %s"
    cursor.execute(query % recipe_id)
    connection.commit()
  connection.close()
  cursor.close()
  return "{'status': 'success'}"


#Update_rating(recipe_id,user_id,rating)
@food_diary.route('/recipes/update_recipe_rating/<recipe_id>')
def update_recipe_rating():
  pass
```

app.py

```
from flask import Flask
from food_diary import food_diary
from api_sample import api_sample
from flask_cors import CORS
from database_manager import DatabaseManager

app = Flask(__name__)
app.secret_key = '_5#y2L"F4Qdkslppwkwn8z\ndkdn\xec]/'
CORS(app)

app.database_manager = DatabaseManager()

app.register_blueprint(food_diary)
app.register_blueprint(api_sample)

app.run(host='0.0.0.0', debug=True, use_reloader=True)
```

database_manager.py

*I didn't write this code, it is part of the shared Erasmus+ API.

```
from mysql.connector.pooling import MySQLConnectionPool
from mysql.connector import errorcode
import time
import traceback

class DatabaseManager:

    def __init__(self):

        self.createConnectionPool()

    def get_connection(self):
        return self.cnxpool.get_connection()
```

```
    def createConnectionPool(self):
        import configparser
        config_info = configparser.ConfigParser()
        config_info.read('docker_config.ini')

        dbconfig = {
        "user": "root",
        "password": config_info["mysql"]["password"],
        "host": config_info["mysql"]["host"], #set host to mysql (Docker service)
        "database":'desapi',
        "port":'3306'
        }


        try:

            self.cnxpool = MySQLConnectionPool(
                                                    pool_name = "mypool",
                                                    pool_size = 32,
                                                    **dbconfig)
        except:
            # sleep and retry - the MySQL container probably not up and running yet
            print("Exception... sleeping for 5 seconds then retry")
            tb = traceback.format_exc()
            print(tb)
            time.sleep(5)
            # try again
            return self.createConnectionPool()

    def insert_dict(self, table_name, my_dict):
        #Inserts a dictionary into table table_name
        #Based on https://stackoverflow.com/questions/9336270/using-a-python-dict-for-a-
sql-insert-statement
        connector = self.cnxpool.get_connection()
        cursor = connector.cursor(dictionary=True)

        columns = ', '.join(my_dict.keys())
        placeholders = ", ".join(["%s"] * len(my_dict))

        stmt = "insert into `{table}` ({columns}) values
({values});".format(table=table_name, columns=",".join(my_dict.keys()),
values=placeholders)
        print (stmt)

        cursor.execute(stmt, list(my_dict.values()))

        connector.commit()
        cursor.close()
        connector.close()
```

```
    def get_all_where(self, table_name, my_dict, sort_by = "", order = "", fetchall =
True):
        # returns all instances of a certain table
        # Performs a SELECT * from TABLE WHERE field_name=value ORDERBY
        #self.log.log_info("DatabaseManager - get_all_where")
        connector = self.cnxpool.get_connection()
        cursor = connector.cursor(dictionary=True)

        values = list(my_dict.values())
        print("Values to update:", values)

        columns_values = " = %s AND ".join(my_dict.keys()) + " = %s"
        print("Columns to update:", columns_values)

        sort = ""
        if sort_by:
            sort = " ORDER BY " + table_name + "." + sort_by #+ " " + order

        query = "SELECT * FROM `{table}` WHERE {columns}".format(table=table_name,
columns=columns_values)
        query += sort
        print("query:", query)

        cursor.execute(query, values)

        if fetchall:
            result = cursor.fetchall()
        else:
            result = cursor.fetchone()

        cursor.close()
        connector.close()

        return result
```