INSTITUTE *of* TECHNOLOGY CARLOW

Institiúid Teicneolaíochta Cheatharlach

Coeliac Disease Symptom Tracker

Project Technical Manual

Student: Niamh Coleman (C00205225)

Supervisor: Joseph Kehoe

Date: 08/04/2019

Contents

Dependencies & Versions

The language versions are as follows:

- Python 3.7
- Flask 1.0.2
- React 16.8.3

Dependencies are the libraries that are required by an application to successfully build and run (as well as the source code).

Following are the dependencies required by the application front-end, as well as the version numbers.

```
"react": "^16.8.3",
"react-calendar": "^2.18.1",
"react-dom": "^16.8.3",
"react-router-dom": "^4.3.1",
"react-scripts": "^2.1.5"
```

To install dependencies using npm:

- npm install dependency_name@version_number

To start the application locally:

- npm start

Following are the dependencies required by the API code:

```
import MySQLdb
from flask import Flask, jsonify, request
from datetime import datetime
from flask_cors import CORS, cross_origin
```

Deployment

The application API and database can be deployed on PythonAnywhere with the free account option by using the following instructions:

- Create a free PythonAnywhere account.
- Create a web application that uses Python 3.7 and Flask.
- Upload the API code to ‘flask_app.py’.
- Initialise a MySQL database using the ‘databases’ tab.
- Connect the database to the API code using the following code:

```
db = MySQLdb.connect("hostname", "pythonanywhere_username", "db_password", "database_name")
```

The application user interface can be deployed on Github Pages for free with the following instructions:

- Clone the Github repo with the code into an empty folder.
- Copy and paste the application files into the cloned folder.
- Install the gh-pages dependency into the project using the following command:

```
npm install gh-pages
```

- Add the following into the package.json file:

```
“homepage”: “http://github_username.github.io/repo_name/
“pre-deploy” : “npm run build”
“deploy” : “gh-pages -g run”
```

The homepage property specifies where Github pages deploys the application.

- To deploy the application run:

```
npm run deploy
```

Technologies & Tools

The following technologies and tools were used to create the application.

Technologies:

- ReactJS
- Python MySQL
- Flask Framework

Tools:

- Visual Studio Code (Front-end development)
- XAMPP (Server solution)
- PyCharm (API development)
- phpMyAdmin (Database)
- PythonAnywhere (Database hosting)
- Github Pages (ReactJS UI Deployment)
- Postman (API Testing)
- FluidUI (UI Prototyping)

Back-end Code (Python MySQL using Flask Framework)

```
import MySQLdb

from flask import Flask, jsonify, request

from datetime import datetime

from flask_cors import CORS, cross_origin

app = Flask(__name__)

cors = CORS(app)

app.config['CORS_HEADERS'] = 'Content-Type'

app.config["DEBUG"] = True


db = MySQLdb.connect("niamhcoleman0.mysql.pythonanywhere-services.com",
"niamhcoleman0", "###", "niamhcoleman0$fyp")


#return user email (account tab)

@app.route('/account/getuserinfo/<user_id>', methods=['GET'])

def getuserinfo(user_id):

    query = "SELECT user_name, user_email FROM niamhcoleman0$fyp.users
WHERE user_id = %s;"

    cursor = db.cursor()

    cursor.execute(query % user_id)

    result = []

    for (user_name, user_email) in cursor:

        result.append({"user_name": user_name, "user_email": user_email})

    cursor.close()

    return jsonify(result)


#change user password

@app.route('/account/changepassword/<new_password>/<user_id>',
methods=['POST'])

def changepassword(new_password,user_id):

    query = "UPDATE users SET user_password = %s WHERE user_id = %s;"

    cursor = db.cursor()
    cursor.execute(query % (new_password, user_id))

    db.commit()

    cursor.close()

    return "{'status': 'success'}"


#tracking symptoms

@app.route('/tracking/logentry', methods = ['POST'])

def logentry():

      try:

            cursor = db.cursor()

            user_id = request.args["user_id"]

            now = datetime.now()

            formatted_date = now.strftime('%Y-%m-%d')

            entry_tod = request.args["entry_tod"]

            entry_emo_id = request.args["entry_emo_id"]

            notes = request.args["notes"]

            symptoms = request.args.getlist('symptoms')


            query = "SELECT entry_id FROM entries WHERE user_id = '%s' AND 
entry_date = '%s' AND entry_tod = '%s';"

            cursor.execute(query  % (user_id, formatted_date, entry_tod))

            result = cursor.fetchall()


            #if an entry for that time of day already exists result will 
contain the entry id

            #if no entry exists result will contain []


            #if result is empty

            #there is no previous entry for this tod

            if not result:

                  # symptom_entry_id = max id in table + 1

                  query = "SELECT MAX(sym_entry_id) AS maximum from 
symptomEntry LIMIT 1;"
```

```
                  cursor.execute(query)
                  result = cursor.fetchall()

                  if result[0][0] is None:
                        new_sym_entry_id = 0
                  else:
                        new_sym_entry_id = result[0][0] + 1

                  query = "INSERT INTO entries (user_id, entry_date,
entry_tod, entry_emo_id, symptom_entry_id, notes) VALUES (%s, %s, %s, %s,
%s, %s);"
                  cursor.execute(query, (user_id, formatted_date,
entry_tod, entry_emo_id, new_sym_entry_id, notes))
                  db.commit()

                  # symptoms contains a list of strings e.g. [ "acne 2",
"insomnia 2" ]
                  # this needs to be changed to the format: [['acne', 2],
['insomnia', 2]] i.e. a list of lists

                  symList = []
                  for symptom in symptoms:
                        x = symptom.split()
                        x[1] = int(x[1])
                        symList.append(x)

                  # symList now contains the format : [ [ "acne", 2 ], [
"insomnia,", 2 ] ]

                  updatedlist = []

                  for symptom in symList:
                        sym_name = symptom[0]
                        query = "SELECT sym_id FROM symptoms WHERE 
sym_name = '%s';"
                        cursor.execute(query % sym_name)
                        result = cursor.fetchone()
                        updatedlist.append([result[0], symptom[1]])
      # at this point, updated list is a list of lists
      # with the format: [[symptom id, symptom severity],[symptom id, 
symptom severity], .....]


                  for record in updatedlist:
                        query = "INSERT INTO symptomEntry (sym_entry_id, 
symptom_entry_sev, sym_id) VALUES ( %s, %s, %s);"
                        cursor.execute(query, (new_sym_entry_id, 
record[1], record[0]))
                        db.commit()
                  cursor.close()
                  return "{'status': 'success'}"
            else:


                  # update the emotion
                  query = "UPDATE entries SET entry_emo_id = '%s' WHERE 
user_id = '%s' AND entry_date = '%s' AND entry_tod = '%s' "
                  cursor.execute(query % (entry_emo_id,user_id, 
formatted_date, entry_tod))
                  db.commit()


      #update the notes only if they have entered notes. This means that 
blank notes wont override the notes previously entered.
                  if notes:
                        # update the notes
                        query = "UPDATE entries SET notes = '%s' WHERE 
user_id = '%s' AND entry_date = '%s' AND entry_tod = '%s' "
                        cursor.execute(query % (notes, user_id, 
formatted_date, entry_tod))
                        db.commit()
                  query = "SELECT symptom_entry_id FROM entries WHERE
user_id = '%s' AND entry_date = '%s' AND entry_tod = '%s';"

                  cursor.execute(query % (user_id, formatted_date,
entry_tod))

                  result = cursor.fetchall()

                  symptom_entry_id = result[0][0]


                  query = "SELECT * FROM symptomEntry WHERE sym_entry_id =
'%s';"

                  cursor.execute(query % symptom_entry_id)

                  result = cursor.fetchall()


                  #result contains a list of the symptoms already logged
for previous entry this tod

                  # list is in format of [[symptom_entry_id,
symptom_entry_sev,sym_id] , [....] ]


                  #to get rid of the symptom entry id from the list

                  existing_Symptoms = []

                  for entry in result:

                        existing_Symptoms.append([entry[2],entry[1]])


                  symList = []

                  for symptom in symptoms:

                        x = symptom.split()

                        x[1] = int(x[1])

                        symList.append(x)


                  # symList now contains the format : [ [ "acne", 2 ], [
"insomnia,", 2 ] ]


                  newSymptoms = []
                  for symptom in symList:
                        sym_name = symptom[0]
                        query = "SELECT sym_id FROM symptoms WHERE
sym_name = '%s';"
                        cursor.execute(query % sym_name)
                        result = cursor.fetchone()
                        newSymptoms.append([result[0], symptom[1]])


      #at this point:
      #existing_Symptoms contains a list in the format of symptom_id,
symptom_severity that already EXISTS IN DB for this entry
      #newSymptoms contains the symptoms that were input by the user VIA
UI not yet logged in the format of symptom_id, symptom_severity that
already exists in db for this entry


      #to update any existing in the db and have newSymptoms left with
only new ones to add to db


                  i = []
                  for new in newSymptoms:
                        s_name = new[0]
                        s_sev = new[1]


                        for existing in existing_Symptoms:
                              if existing[0] == s_name:
                                    i.append(new)
                                    query = "UPDATE symptomEntry SET
symptom_entry_sev = '%s' WHERE sym_entry_id = '%s' AND sym_id = '%s';"
                                    cursor.execute(query % (s_sev,
symptom_entry_id, s_name))
                                    db.commit()
                  for entry in i:
                        newSymptoms.remove(entry)
```

```
	#newSymptos now contains the entries that are new (didn't need to be updated)
			for entry in newSymptoms:
				query = "INSERT INTO symptomEntry (sym_entry_id, symptom_entry_sev, sym_id) VALUES ( %s, %s, %s);"
				cursor.execute(query, (symptom_entry_id, entry[1], entry[0]))
				db.commit()

		cursor.close()
		return "{'status': 'success'}"

	except Exception as e:
		error = "{'error': " + str(e) + "}"
	return error

#returning info from a particular day
@app.route('/history/getdayinfo/<user_id>/<target_date>/', methods = ['GET'])
def getdayinfo(user_id, target_date):
	try:
		result_dict = {}
		cursor = db.cursor()
		query = "SELECT symptom_entry_id FROM entries WHERE user_id = '%s' AND entry_date = '%s' AND entry_tod = '%s';"
		notesquery = "SELECT notes FROM entries WHERE user_id = '%s' AND entry_date = '%s' AND entry_tod = '%s';"
		emoquery = "SELECT entry_emo_id FROM entries WHERE user_id = '%s' AND entry_date = '%s' AND entry_tod = '%s';"

		#morning
		cursor.execute(query % (user_id, target_date, 'morning'))
		morning = cursor.fetchall()
```

```
            #afternoon
            cursor.execute(query % (user_id, target_date, 'afternoon'))
            afternoon = cursor.fetchall()

            #evening
            cursor.execute(query % (user_id, target_date, 'evening'))
            evening = cursor.fetchall()

            #night
            cursor.execute(query % (user_id, target_date, 'night'))
            night = cursor.fetchall()



            if morning:
                  morning_symptom_entry_id = morning[0][0]
                  query = "SELECT symptom_entry_sev, sym_id FROM 
symptomEntry WHERE sym_entry_id = '%s';"
                  cursor.execute(query % morning_symptom_entry_id)
                  result = cursor.fetchall()
                  #[[symptom severity, symptom id]]
                  #swap symptom id for the symptom name
                  x = []
                  for entry in result:
                        id = entry[1]
                        severity = entry[0]
                        query = "SELECT sym_name FROM symptoms WHERE 
sym_id = '%s';"
                        cursor.execute(query % id)
                        result = cursor.fetchall()
                        name = result[0][0]
                        if severity == 1:
                              new_sev = "(low) "
                        elif severity == 2:
                              new_sev = "(moderate) "
                        elif severity == 3:
                              new_sev = "(severe) "
                        x.append([name+ " " + new_sev])
                  result_dict['morning'] = x
                  cursor.execute(notesquery % (user_id, target_date, 
'morning'))
                  notes = cursor.fetchall()
                  if notes:
                        result_dict['morning_note'] = notes
                  cursor.execute(emoquery % (user_id, target_date, 
'morning'))
                  emo_id = cursor.fetchall()
                  if emo_id:
                        result_dict['morning_emo'] = emo_id




            if afternoon:
                  afternoon_symptom_entry_id = afternoon[0][0]
                  query = "SELECT symptom_entry_sev, sym_id FROM 
symptomEntry WHERE sym_entry_id = '%s';"
                  cursor.execute(query % afternoon_symptom_entry_id)
                  result = cursor.fetchall()
                  # [[symptom severity, symptom id]]
                  # swap symptom id for the symptom name
                  x = []
                  for entry in result:
                        id = entry[1]
                        severity = entry[0]
                        query = "SELECT sym_name FROM symptoms WHERE 
sym_id = '%s';"
```

```
                        cursor.execute(query % id)
                        result = cursor.fetchall()
                        name = result[0][0]
                        if severity == 1:
                              new_sev = "(low) "
                        elif severity == 2:
                              new_sev = "(moderate) "
                        elif severity == 3:
                              new_sev = "(severe) "
                        x.append([name+ " " + new_sev])
                  result_dict['afternoon'] = x
                  cursor.execute(notesquery % (user_id, target_date,
'afternoon'))
                  notes = cursor.fetchall()
                  if notes:
                        result_dict['afternoon_note'] = notes
                  cursor.execute(emoquery % (user_id, target_date,
'afternoon'))
                  emo_id = cursor.fetchall()
                  if emo_id:
                        result_dict['afternoon_emo'] = emo_id


            if evening:
                  evening_symptom_entry_id = evening[0][0]
                  query = "SELECT symptom_entry_sev, sym_id FROM
symptomEntry WHERE sym_entry_id = '%s';"
                  cursor.execute(query % (evening_symptom_entry_id))
                  result = cursor.fetchall()
                  # [[symptom severity, symptom id]]
                  # swap symptom id for the symptom name
                  x = []
                  for entry in result:
                        id = entry[1]
                        severity = entry[0]
                        query = "SELECT sym_name FROM symptoms WHERE 
sym_id = '%s';"
                        cursor.execute(query % id)
                        result = cursor.fetchall()
                        name = result[0][0]
                        if severity == 1:
                              new_sev = "(low) "
                        elif severity == 2:
                              new_sev = "(moderate) "
                        elif severity == 3:
                              new_sev = "(severe) "
                        x.append([name+ " " + new_sev])
                  result_dict['evening'] = x
                  cursor.execute(notesquery % (user_id, target_date, 
'evening'))
                  notes = cursor.fetchall()
                  if notes:
                        result_dict['evening_note'] = notes
                  cursor.execute(emoquery % (user_id, target_date, 
'evening'))
                  emo_id = cursor.fetchall()
                  if emo_id:
                        result_dict['evening_emo'] = emo_id



            if night:
                  night_symptom_entry_id = night[0][0]
                  query = "SELECT symptom_entry_sev, sym_id FROM 
symptomEntry WHERE sym_entry_id = '%s';"
                  cursor.execute(query % (night_symptom_entry_id))
                  result = cursor.fetchall()
                  # [[symptom severity, symptom id]]
                  # swap symptom id for the symptom name
                  x = []
                  for entry in result:
                        id = entry[1]
                        severity = entry[0]
                        query = "SELECT sym_name FROM symptoms WHERE
sym_id = '%s';"
                        cursor.execute(query % id)
                        result = cursor.fetchall()
                        name = result[0][0]
                        if severity == 1:
                              new_sev = "(low) "
                        elif severity == 2:
                              new_sev = "(moderate) "
                        elif severity == 3:
                              new_sev = "(severe) "
                        x.append([name+ " " + new_sev])
                  result_dict['night'] = x
                  cursor.execute(notesquery % (user_id, target_date,
'night'))
                  notes = cursor.fetchall()
                  if notes:
                        result_dict['night_note'] = notes
                  cursor.execute(emoquery % (user_id, target_date,
'night'))
                  emo_id = cursor.fetchall()
                  if emo_id:
                        result_dict['night_emo'] = emo_id


            #result_dict consists of a dict with format: { "evening": [ [
2, "acne" ], [ 2, "insomnia" ] ], "morning": [ [ 2, "acne" ] ], "night": [
[ 2, "acne" ], [ 2, "insomnia" ] ] }
```

```
        cursor.close()

        return jsonify(result_dict)




  except Exception as e:

        error = "{'error': " + str(e) + "}"

        return error
```

Front-end Code (ReactJS)

Package.json

```
{
  "name": "tracker",
  "version": "0.1.0",
  "private": true,
  "homepage": "http://niamhcoleman.github.io/demo/",
  "dependencies": {
    "gh-pages": "^2.0.1",
    "react": "^16.8.3",
    "react-calendar": "^2.18.1",
    "react-dom": "^16.8.3",
    "react-router-dom": "^4.3.1",
    "react-scripts": "^2.1.5"
  },
  "scripts": {
    "predeploy":"npm run build",
    "deploy":"gh-pages -d build",
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": [
    ">0.2%",
    "not dead",
    "not ie <= 11",
    "not op_mini all"
  ]
}
```

Index.js

```
import React from 'react';

import ReactDOM from 'react-dom';

import { BrowserRouter as Router } from "react-router-dom";

import './index.css';

import App from './App';

import * as serviceWorker from './serviceWorker';


ReactDOM.render(
  <Router>

    <App />

  </Router>, document.getElementById('root'));

serviceWorker.unregister();
```

App.js

```
import React, { Component } from "react";
import { Route, Link } from "react-router-dom";
import ChooseADay from './ChooseADay';
import Settings from './Settings';
import TrackingForm from './TrackingForm';
import "./App.css";



const Tracking = () => (
  <div>
    <TrackingForm/>
  </div>
);

const Account = () => (
  <div>
    <Settings/>
  </div>
);

const History = () => (
  <div>
    <ChooseADay/>
  </div>
);

 class App extends Component {


  render()
  {
    return (
      <div id ="nav">
        <ul>
          <li>
            <Link to="/Tracking">Tracking</Link>
          </li>
          <li>
            <Link to="/History">History</Link>
          </li>
          <li>
            <Link to="/Account">Account</Link>
          </li>
        </ul>


        <Route path="/Tracking" exact component={Tracking} />
        <Route path="/History" component={History}/>
        <Route path="/Account" component={Account}/>
      </div>
    );
  }
}

export default App;
```

ChooseADay.js

```
import React, { Component } from 'react';
import './ChooseADay.css';
import Calendar from 'react-calendar';


class ChooseADay extends Component {
  state = {
    date: new Date(),
    response: 'Please Select a Day.',
    morning_sym: ' ',
    morning_n: ' ',
    morning_e: ' ',

    afternoon_sym: ' ',
    afternoon_n: ' ',
    afternoon_e: ' ',

    evening_sym: ' ',
    evening_n: ' ',
    evening_e: ' ',

    night_sym: '',
    night_n: ' ',
    night_e: ' ',
  }

  onClickDay = date => {
    //reset these values so they don't still appear or stack up when user
clicks another/reclicks day
    this.setState({morning_sym: " "});
    this.setState({morning_n: " "});
    this.setState({morning_e: " ",})

    this.setState({afternoon_sym: " "});
    this.setState({afternoon_n: " "});
    this.setState({afternoon_e: " ",})

    this.setState({evening_sym: " "});
    this.setState({evening_n: " "});
    this.setState({evening_e: " ",})

    this.setState({night_sym: " "});
    this.setState({night_n: " "});
    this.setState({night_e: " ",})

    this.setState({ date })
  }


  onChange = () => {
    //for formatting the input date to format YYYY/MM/DD
    var d = new Date(this.state.date)
    d = d.getFullYear() + "-" + (d.getMonth()+1) + "-" + d.getDate()

    fetch('http://127.0.0.1:5001/history/getdayinfo/1/' + d, {method: 'GET'})
    .then(response => {
      return response.json()
    })
    .then(data => {
      var s = JSON.stringify(data);

      if (s === "{}") //then there are no results
      {
        this.setState({response: "No symptoms were tracked on this day."});
      }
      else
      {
        this.setState({response: d});
        var x = JSON.parse(s);
        var i = 0;
        var current = '';

        //symptom results from json response

        //MORNING
        if (x.morning !== undefined)
        {
          for (i = 0; i< x.morning.length; i++)
          {
            current = x.morning[i];

            this.setState({morning_sym: this.state.morning_sym + " " + current
+ " ,"});
          }
        }
        if (x.morning_note !== undefined)
        {
          for (i = 0; i< x.morning_note.length; i++)
          {
            current = x.morning_note[i];

            this.setState({morning_n: this.state.morning_n + current + " "});
          }
        }
        if (x.morning_emo !== undefined)
        {
          for (i = 0; i< x.morning_emo.length; i++)
          {
            current = x.morning_emo[i];

            if (current == "1")
            {
              current = "unhappy";
            }
            else if (current  == "2")
            {
              current = "okay";
            }

            else if (current  == "3")
            {
              current = "happy";
            }
            this.setState({morning_e: this.state.morning_e + current + " "});
          }
        }

        //AFTERNOON
        if (x.afternoon !== undefined)
        {
          for (i = 0; i< x.afternoon.length; i++)
          {
            current = x.afternoon[i];

            this.setState({afternoon_sym: this.state.afternoon_sym + " " +
current + " ,"});
          }
        }
        if (x.afternoon_note !== undefined)
        {
          for (i = 0; i< x.afternoon_note.length; i++)
          {
            current = x.afternoon_note[i];

            this.setState({afternoon_n: this.state.afternoon_n + current + "
"});
          }
        }
        if (x.afternoon_emo !== undefined)
        {
          for (i = 0; i< x.afternoon_emo.length; i++)
          {
            current = x.afternoon_emo[i];

            if (current == "1")
            {
              current = "unhappy";
            }

            else if (current  == "2")
            {
              current = "okay";
            }

            else if (current  == "3")
            {
              current = "happy";
            }
            this.setState({afternoon_e: this.state.afternoon_e + current + "
"});
          }
        }

        //EVENING
        if (x.evening !== undefined)
        {
          for (i = 0; i< x.evening.length; i++)
          {
            current = x.evening[i];

            this.setState({evening_sym: this.state.evening_sym + " " + current
+ " ,"});
          }
        }
        if (x.evening_note !== undefined)
        {
          for (i = 0; i< x.evening_note.length; i++)
          {
            current = x.evening_note[i];
            this.setState({evening_n: this.state.evening_n + " " + current + "
"});
          }
        }
        if (x.evening_emo !== undefined)
        {
          for (i = 0; i< x.evening_emo.length; i++)
          {
            current = x.evening_emo[i];

            if (current == "1")
            {
              current = "unhappy";
            }

            else if (current  == "2")
            {
              current = "okay";
            }

            else if (current  == "3")
            {
              current = "happy";
            }
            this.setState({evening_e: this.state.evening_e + current + " "});
          }
        }

        //NIGHT
        if (x.night !== undefined)
        {
          for (i = 0; i< x.night.length; i++)
          {
            current = x.night[i];

            this.setState({night_sym: this.state.night_sym + " " + current + "
,"});
          }
        }
        if (x.night_note !== undefined)
        {
          for (i = 0; i< x.night_note.length; i++)
          {
            current = x.night_note[i];
            this.setState({night_n: this.state.night_n + " " + current + "
"});
          }
        }
        if (x.night_emo !== undefined)
        {
          for (i = 0; i< x.night_emo.length; i++)
          {
            current = x.night_emo[i];

            if (current == "1")
            {
              current = "unhappy";
            }

            else if (current  == "2")
            {
              current = "okay";
            }

            else if (current  == "3")
            {
              current = "happy";
            }
            this.setState({night_e: this.state.night_e + " " + current + "
"});
          }
        }
      }
    })
  }

  render() {
    return (
      <div>
        <Calendar
          onChange={this.onChange}
          value={this.state.date}
          maxDate={new Date()}
          onClickDay = {this.onClickDay}
        />

      <p id = "response">{this.state.response}</p>
      <hr></hr>
      <p id = "tod_heading">Morning:</p>
      <p id = "info-heading">Symptoms:</p> {this.state.morning_sym}<br/>
      <p id = "info-heading">Notes: </p>{this.state.morning_n}<br/>
      <p id = "info-heading">Emotion: </p>{this.state.morning_e}<br/>

      <hr></hr>

      <p id = "tod_heading">Afternoon:</p>
      <p id = "info-heading">Symptoms:</p> {this.state.afternoon_sym}<br/>
      <p id = "info-heading">Notes: </p>{this.state.afternoon_n}<br/>
      <p id = "info-heading">Emotion: </p>{this.state.afternoon_e}<br/>

      <hr></hr>

      <p id = "tod_heading">Evening:</p>
      <p id = "info-heading">Symptoms:</p> {this.state.evening_sym}<br/>
      <p id = "info-heading">Notes: </p>{this.state.evening_n}<br/>
      <p id = "info-heading">Emotion: </p>{this.state.evening_e}<br/>

      <hr></hr>

      <p id = "tod_heading">Night:</p>
      <p id = "info-heading">Symptoms:</p> {this.state.night_sym}<br/>
      <p id = "info-heading">Notes: </p>{this.state.night_n}<br/>
      <p id = "info-heading">Emotion: </p>{this.state.night_e}<br/>

      </div>
    );
  }
}

export default ChooseADay;
```

Settings.js

```
import React from 'react';

import './Settings.css';

import AccountInfo from './AccountInfo';

import ChangePassword from './ChangePassword';


const getinfoCall =  'http://127.0.0.1:5001/account/getuserinfo/1'


class Settings extends React.Component {

  constructor(props) {

    super(props);

    this.state = {

      showButton: false,

      name: '',

      email:'',

    };

  }


  toggle = () => {

    this.setState({ showButton: true, showButton2: false });

    fetch(getinfoCall)

    .then(response => {

      return response.json()

    })

    .then(data => {

      // Work with JSON data here via variable 'data'

      var s = JSON.stringify(data);
      var x = JSON.parse(s);

      this.setState({name: x[0].user_name});

      this.setState({email: x[0].user_email});

    })

  };


  toggle2 = () => {

    this.setState({ showButton: false, showButton2: true });

  };




  render() {

    const {name, email} = this.state;

    return (

      <div id = "accDiv">

        <button id = "viewinfo" onClick={this.toggle} >View Account
Info</button>

        <br></br><br></br><br></br>

        <button id = "changepass" onClick={this.toggle2}>Change
Password</button>

        <br></br><br></br><br></br>


        {this.state.showButton ? <AccountInfo

                                  name = {name}

                                  email = {email}/> : null}

        {this.state.showButton2 ? <ChangePassword/> : null}

      </div>

    );

  }

}


export default Settings;
```

ChangePassword.js

```
import React from 'react';
import './Settings.css'


class AccountInfo extends React.Component {
    constructor(props) {
      super(props);
      this.state = {value: ''};

      this.handleChange = this.handleChange.bind(this);
      this.handleSubmit = this.handleSubmit.bind(this);
    }

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

    handleSubmit() {
        var newpass = this.state.value
        if (newpass.length <= 3)
        {

          window.alert('Your Password was could not be changed.');
        }
        else {
          fetch(' http://127.0.0.1:5001/account/changepassword/"' + newpass +
'"/1', {method: 'POST'})
          window.alert('Your Password was successfully changed.');
        }

    }

    render() {
      return (
            <form onSubmit={this.handleSubmit}>
            <p id = "accHeading">Change Password:</p><br/>
            <label>
                New Password:
                <br></br>
                <input type="password" value={this.state.value}
onChange={this.handleChange} id = "passchange" required/>
            </label>
            <br></br> <br></br> <br></br>
            <input type="submit" value="Confirm" id = "pass_submit"/>
            </form>
      );
    }
  }
export default AccountInfo;
```

TrackingForm.js

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

import TimeOfDay from "./Steps/TimeOfDay"; //step 1

import CallSelectSymptoms from './Steps/CallSelectSymptoms'; //step2

import SelectEmotion from './Steps/SelectEmotion'; //step 3

import Notes from './Steps/Notes'; //step 4

import TrackingFinalStep from './TrackingFinalStep'; //step 5/success page


export class TrackingForm extends Component {

    state = {

        step: 1,

        timeofday: '',

        symptoms: '',

        emotion:'',

        notes:'',

    }


    //proceed to next step

    nextStep= () => {

        const{ step } = this.state;

        this.setState({

            step: step+1

        });

    }


    //go to previous step
    prevStep= () => {
        const{ step } = this.state;
        this.setState({
            step: step-1
        });
    }

//handle all field changes
    handleChange = input => e => {

        this.setState({[input]: e.target.value});
    }

    handleSymptomChange = input => e => {
        this.setState({symptoms: this.state.symptoms + "&symptoms=" +
e.target.value});
    }



    render() {
        const { step } = this.state;
        const { timeofday, symptoms,emotion,notes} = this.state;
        const values = { timeofday, symptoms,emotion,notes}

        switch(step){
            case 1:
                return (
                    <TimeOfDay
                    nextStep={this.nextStep}
                    handleChange={this.handleChange}
                    values={values}
                    />
                )
            case 2:
                return (
                    <CallSelectSymptoms
                    nextStep={this.nextStep}
                    prevStep={this.prevStep}
                    handleSymptomChange={this.handleSymptomChange}
                    values={values}
                    />
                )

            case 3:
                return (
                    <SelectEmotion
                    nextStep={this.nextStep}
                    prevStep={this.prevStep}
                    handleChange={this.handleChange}
                    values={values}
                    />
                )
            case 4:
                return (
                    <Notes
                    nextStep={this.nextStep}
                    prevStep={this.prevStep}
                    handleChange={this.handleChange}
                    values={values}
                    />
                )
            case 5:
                return (
                    <TrackingFinalStep
                    nextStep={this.nextStep}
                    prevStep={this.prevStep}
                    values={values}
                    />
                )
            default: return (
                <div>
                    <TrackingForm/>
                </div>

            )
        }


    }
}

export default TrackingForm
```

TrackingFinalStep.js

```
import React from 'react';


class TrackingFinalStep extends React.Component {


  back = e => {
    e.preventDefault();
    this.props.prevStep();
  }


  apiCall(tod, sym, emo, notes) {
    if (emo && tod && sym)

    {

      if (emo === "Happy")

        {

          var emo_id = 3

        }

    else if (emo === "ok")

        {

          emo_id = 2

        }

    else

        {

          emo_id = 1

        }


      fetch('http://127.0.0.1:5001/tracking/logentry?user_id=1&entry_tod=' +
tod + '&entry_emo_id=' + emo_id + '&notes=' + notes + sym , {method: 'POST'})

      window.alert("You have successfully logged symptoms.")

      this.props.nextStep();


    }

    else {

      window.alert("There was an issue with your form. The following steps are
compulsary: time of day, at least one symptom & mood.")

      this.props.prevStep();

    }

  }


  render() {


    const {values: {timeofday, symptoms,emotion,notes}} = this.props;

    return (
        <div id = "buts">
            <button id = "prevBut" onClick = {this.back}>
              Back
            </button>
            <button id = "nextBut" onClick = {this.apiCall(timeofday, 
symptoms, emotion, notes)}>
              Confirm
            </button>
          </div>


    );
  }
}

export default TrackingFinalStep;
```

CallSelectSymptoms.js

```
import React from 'react';
import Acne from './Acne';
import Vomit from './Vomit';
import Bloat from './Bloat';
import Indigestion from './Indigestion';


class CallSelectSymptoms extends React.Component {
  continue = e => {
    e.preventDefault();
    this.props.nextStep();
  }
  back = e => {
    e.preventDefault();
    this.props.prevStep();
```

```
}

render() {
    const {values, handleSymptomChange} = this.props;

    return (
      <div>
        <p id = "heading"><u>What Symptoms do you Have Today?</u></p>
        <div onChange = {handleSymptomChange('symptoms')}>
          <Acne values={values}/>
          <br></br>
          <Vomit values = {values}/>
          <br></br>
          <Bloat values = {values}/>
          <br></br>
          <Indigestion values = {values}/>
        </div>
        <br></br><br></br>
        <div id = "responsediv">
        </div>



        <div id = "buts">
        <button id = "prevBut" onClick = {this.back}>
            Back
          </button>
        <button id = "nextBut" onClick = {this.continue}>
            Next
          </button>
      </div>
```

```
        </div>
    );
}
}

export default CallSelectSymptoms;
```

SelectEmotion.js

```
import React from 'react';
import './SelectEmotion.css';

class SelectEmotion extends React.Component {
  continue = e => {
    e.preventDefault();
    this.props.nextStep();
  }
  back = e => {
    e.preventDefault();
    this.props.prevStep();
  }

  render() {
    const {handleChange} = this.props;

    return (
        <div onChange={handleChange('emotion')}>
        <p id = "heading"><u>How are you today?</u></p>
            <form>
                <div id = "emotionradio" >
```

```
                    <input type = "radio" value = "Happy" id = "Happy" name =
"selector"/>
                    <label id = "emotionlabel" for="Happy">Happy</label>
                    <br/><br/><br/>
                    <input type = "radio" value = "ok" id = "ok" name =
"selector"/>
                    <label id = "emotionlabel" for="ok">Okay</label>
                    <br/><br/><br/>
                    <input type = "radio" value = "Unhappy" id = "Unhappy"
name = "selector"/>
                    <label id = "emotionlabel" for="Unhappy">Unhappy</label>

                </div>
                </form>
                <div id = "buts">
                <button id = "prevBut" onClick = {this.back}>
                    Back
                  </button>
                <button id = "nextBut" onClick = {this.continue}>
                    Next
                  </button>
                </div>
        </div>
    );
  }
}

export default SelectEmotion;
```

TimeOfDay.js

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

```
export class TimeOfDay extends Component {

  continue = e => {
    e.preventDefault();
    this.props.nextStep();
  }
  render() {
    //so I cacn use values as a var instead of this.props.values
    const {handleChange} = this.props;
    return (
      <div>
        <p id = "heading"><u>Please Choose a Time of Day: </u></p>
        <form onChange={handleChange('timeofday')}>
            <div id="timeofdayradio">
              <input type="radio" id="option-one" value = "morning"
name="selector"/>
              <label id = "timeofdaylabel" for="option-one">Morning</label>
                <br/><br/><br/>
              <input type="radio" id="option-two" value = "afternoon"
name="selector"/>
              <label id = "timeofdaylabel" for="option-two">Afternoon</label>
                <br/><br/><br/>
              <input type="radio" id="option-three" value = "evening"
name="selector"/>
              <label id = "timeofdaylabel" for="option-three">Evening</label>
                <br/><br/><br/>
              <input type="radio" id="option-four" value = "night"
name="selector"/>
              <label id = "timeofdaylabel" for="option-four">Night</label>
            </div>
        </form>
        <div id = "buts">
```

```
          <button id = "nextBut" onClick = {this.continue}>
              Next
            </button>
        </div>
      </div>
    )
  }
}
export default TimeOfDay;
```

Acne.js

```
import React from 'react';
import './SelectSymptomsStyle.css';

class Acne extends React.Component {
    constructor(props) {
        super(props);

        this.style = {};

          this.style.low = {
            backgroundColor: "yellow",
          }

          this.style.mod = {
            backgroundColor: "orange",
          }

          this.style.sev = {
            backgroundColor: "red",
          }


      };


  render() {


      return (
        <div>
            <form>
            <p id = "symptomName">Acne</p>
                <div id = "syms">
                    <label class="container">
                        <input align = "left" type="radio" name="radio" value
= "Acne 1"/>
                        <span class="checkmark" style=
{this.style.low}></span>
                    </label>
                    <label class="container">
                        <input align = "center"  type="radio" name="radio"
value =  "Acne 2"/>
                        <span class="checkmark" style= {this.style.mod}
></span>
                    </label>
                    <label class="container">
                        <input align = "right"  type="radio" name="radio"
value =  "Acne 3"/>
                        <span class="checkmark" style=
{this.style.sev}></span>
                    </label>
                </div>
            </form>
        </div>
    );

}

}


export default Acne;
```

Notes.js

```
import React from 'react';
import './Notes.css';

class Notes extends React.Component {
  continue = e => {
    e.preventDefault();
    this.props.nextStep();
  }
  back = e => {
    e.preventDefault();
    this.props.prevStep();
  }


  render() {
    const { handleChange} = this.props;

    return (
        <div>
        <p id = "heading"><u>Is There Anything You Would Like to Add?</u></p>

          <form>
            <div id = "temp">
            </div>
            <div id = "notesdiv">
            <textarea name="notes" rows="6" cols="45"
onChange={handleChange('notes')}>

            </textarea>
            </div>

          </form>
          <div id = "buts">
```

```
            <button id = "prevBut" onClick = {this.back}>
              Back
            </button>
            <button id = "nextBut" onClick = {this.continue}>
              Next
            </button>
          </div>
        </div>
    );
  }
}

export default Notes;
```

SQL Table Creation Queries

```
CREATE TABLE IF NOT EXISTS 'users' (
	`user_id` INT(6),
	`user_name` VARCHAR(10),
	`user_email` VARCHAR(50) UNIQUE,
	`user_password` VARCHAR(50),
	 PRIMARY KEY (`user_id`)
)

CREATE TABLE IF NOT EXISTS 'symptoms' (
	`sym_id` INT(6) UNIQUE,
	`sym_name` VARCHAR(20) UNIQUE,
	`sym_desc` VARCHAR(100),
	PRIMARY KEY (`sym_id`)
)

CREATE TABLE IF NOT EXISTS 'emotions' (
	`emo_id` INT(6),
	`emo_name` VARCHAR(20) UNIQUE,
	PRIMARY KEY (`emo_id`)
)
CREATE TABLE IF NOT EXISTS 'entries' (
	`user_id` INT(6),
	`entry_id` INT(6) AUTO_INCREMENT,
	`entry_date` DATE,
	`entry_tod` VARCHAR(10),
	`symptom_entry_id` INT(6),
	`entry_emo_id` INT(6),
      `notes` VARCHAR (25),
      PRIMARY KEY (`entry_id`)
)

CREATE TABLE IF NOT EXISTS 'symptomentry' (
      `sym_entry_id` INT(6),
      `symptom_entry_sev` INT (6),
      `sym_id` INT(6)
)
```

SQL Table Data Initialising

The following information is inserted into the MySQL database on creation as it is needed by the application. The application currently has no log in functionality and therefore requires the insertion of a user so that the application can function correctly. The other insertion statements include inserting the symptoms and emotions that the application needs to function.

```
LOCK TABLES `emotions` WRITE;

INSERT INTO `emotions` VALUES (1,'unhappy'),(2,'okay'),(3,'happy');

UNLOCK TABLES;


LOCK TABLES `symptoms` WRITE;

INSERT INTO `symptoms` VALUES (1,'Acne','acne
description'),(2,'Vomit','vomit description'),(3,'Bloat','bloat
description'),(4,'Indigestion','indigestion description');

UNLOCK TABLES;


LOCK TABLES `users` WRITE;

INSERT INTO `users` VALUES (1,'John
Doe','example@example.com','password');

UNLOCK TABLES;
```