MVC Handlebars

```
"dependencies": {
  "express": "^4.16.3",
  "express-handlebars": "^3.0.0"
}
```

```
// Dependencies
const express = require('express');

// Import express-handlebars
const exphbs = require('express-handlebars');
const hbs = exphbs.create({});
const path = require('path');

// Sets up the Express App
const app = express();
const PORT = process.env.PORT || 3001;

// Describe what the following two lines of code are doing.
        // The following two lines of code are setting Handlebars.js
                as the default template engine.
app.engine('handlebars', hbs.engine);
app.set('view engine', 'handlebars');
app.use(express.static(path.join(__dirname, 'public')));
app.use(require('./controllers/dish-routes'));

// Starts the server to begin listening
app.listen(PORT, () => {
   console.log('Server listening on: http://localhost:' + PORT);
  });
```

```
// Is this a Model, a View, or a Controller?
        // This file is a Controller.
// What is it responsible for handling?
        // It routes commands to the Model and View parts.

const router = require('express').Router();

// Add a comment describing the purpose of the 'get' route
        // GET route for getting all of the dishes that are on the menu
router.get('/', async (req, res) => {
  // Add a comment describing the purpose of the render method
        // This method is rendering the 'all' Handlebars.js template.
                This is how we connect each route to the correct template.
  res.render('all');
});

module.exports = router;
```

```
// Here is where we set up our Dish model, for when we are
        ready to connect to a database in future activities.
// Add a comment indicating how this file fits into the MVC
        framework.
        // This file is a Model.
// What it is responsible for handling?
        // It is responsible for handling data and business logic.

const { Model, DataTypes } = require('sequelize');
const sequelize = require('../config/connection');

class Dish extends Model {}

Dish.init(
  {
   id: {
    type: DataTypes.INTEGER,
    allowNull: false,
    primaryKey: true,
    autoIncrement: true
   },
   dish_name: {
    type: DataTypes.STRING,
    allowNull: false,
   },
   description: {
    type: DataTypes.TEXT,
    allowNull: true
   },
   guest_name: {
      type: DataTypes.STRING,
      allowNull: false,
     },
   has_nuts: {
    type: DataTypes.BOOLEAN,
   }
  },
  {
   sequelize,
   freezeTableName: true,
   underscored: true,
   modelName: 'dish'
  }
);
module.exports = Dish;
```

```
<!-- This template includes hardcoded data.
        In future lessons, you will be rendering data dynamically. -->
<div class="menu-container">
  <div class="card-header col-md-10">
    <h2>
      &#127869 Current Menu
    </h2>
  </div>
    <div>
      <ol>
        <li>
          French Bread with Brie Cheese
        </li>
        <li>
          Cheese Plate with Spanish Chorizo
        </li>
        <li>
          Fish Tacos
        </li>
        <li>
          Tropical Fruit Salad
        </li>
        <li>
          Pork Gyoza
        </li>
        <li>
          Yebeg Tibs with Injera Bread
        </li>
        <li>
          Cape Malay Curry
        </li>
      </ol>
    </div>
  </div>
</div>
```

```
<!-- Add a comment indicating how this file fits into the MVC framework
        (is it a Model, a View, or a Controller?) and what it is responsible for handling. -->
        <!-- This file is part of the View. It handles UI/UX concerns (layout and display)-->
<head>
  <meta charset="UTF-8">
  <title>Potluck Party</title>
  <link rel="stylesheet" href="/css/jass.css">
  <link rel="stylesheet" href="/css/style.css">
</head>
<body>
  <div class="flex-column justify-space-around">
    <header class="display-flex justify-space-between align-center p-2">
      <h1>
        &#127881 Potluck Party! &#127881
      </h1>
    </header>
      <main class="container container-fluid mt-5">
      <!-- Add a comment describing what the following line of code is doing. -->
            <!-- The following line is a place holder for where the main content will be rendered. -->
      <!-- What should we expect to see when visiting http://localhost:3001/ in the browser? -->
            <!-- We would expect to see the contents of all.handlebars in the browser,
                    because that is what we are instructing it to render in dish-routes.js. -->
      {{{ body }}}
      </main>
  </div>
  </body>
</html>
```

Expressions

```
// Dependencies
const express = require('express');
const exphbs = require('express-handlebars');
const path = require('path');
const hbs = exphbs.create({});

// Sets up the Express App
const app = express();
const PORT = process.env.PORT || 3001;

// Set Handlebars as the default template engine.
app.engine('handlebars', hbs.engine);
app.set('view engine', 'handlebars');

app.use(express.static(path.join(__dirname, 'public')));
app.use(require('./controllers/dish-routes'));

// Starts the server to begin listening
app.listen(PORT, () => {
  console.log('Server listening on: http://localhost:' + PORT);
```

```
<head>
  <meta charset="UTF-8">
  <title>Potluck Party</title>
  <link rel="stylesheet" href="/css/jass.css">
  <link rel="stylesheet" href="/css/style.css">
</head>
<body>
  <div class="flex-column justify-space-around">
    <header class="display-flex justify-space-between align-center p-2">
      <h1>
        &#127881 Potluck Party! &#127881
      </h1>
    </header>
      <main class="container container-fluid mt-5">
        {{{ body }}}
      </main>
  </div>
  </body>
</html>
```

```
<div class="dish-card">
  <div class="dish-card-header">
    <!-- Below we are rendering the name of one dish
            using a Handlebars.js expression -->
    {{dish_name}}
  </div>
  <div class="card-body">
    <!-- We render the description below,
            using a Handlebars.js expression. -->
    <p>
      Description:
      {{description}}
    </p>
  </div>
</div>
```

```
// Here is where we set up our Dish model, for when we are ready
        to connect to a database in future activities.
const { Model, DataTypes } = require('sequelize');
const sequelize = require('../config/connection');

class Dish extends Model {}

Dish.init(
  {
    id: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
    },
    dish_name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    description: {
      type: DataTypes.TEXT,
      allowNull: true,
    },
    guest_name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    has_nuts: {
      type: DataTypes.BOOLEAN,
    },
  },
  {
    sequelize,
    freezeTableName: true,
    underscored: true,
    modelName: 'dish',
  }
);

module.exports = Dish;
```

```
const router = require('express').Router();
// We are using hardcoded data here, where would our data usually come from?
        Remember - we haven't yet set up a database or Sequelize in our app.
const dishes = [
  {
    dish_name: 'French Bread with Brie Cheese',
    description: 'French baguette with warm brie',
  },
  {
    dish_name: 'Cheese Plate with Spanish Chorizo',
    description:
      'Manchego, Iberico, Cabrales, fig jam, grapes, pecans, and Spanich Chorizo',
  },
  {
    dish_name: 'Fish Tacos',
    description:
      'Battered/fried fish, corn tortillas, fresh slaw with jalepenos and cilantro,
            pickled red onion',
  },
  {
    dish_name: 'Tropical Fruit Salad',
    description: 'Guava, papaya, pineapple, mango, and star fruit',
  },
  {
    dish_name: 'Pork Gyoza',
    description:
      'Homemade japanese dumplings stuffed with a pork and green onion filling',
  },
  {
    dish_name: 'Yebeg Tibs with Injera Bread',
    description:
      'Marinated lamb dish with rosemary, garlic, onion, tomato, jalapeño and the East
            African spice berbere',
  },
  {
    dish_name: 'Cape Malay Curry',
    description: 'Chicken and vegitable curry dish with basmati rice',
  },
];

// get all dishes
router.get('/', async (req, res) => {
  res.render('all');
});

// get one dish
router.get('/dish/:num', async (req, res) => {
// This method renders the 'dish' template, and uses params to select the correct dish
        to render in the template, based on the id of the dish.
// Now, we have access to a dish description in the 'dish' template.
  return res.render('dish', dishes[req.params.num - 1]);
});

module.exports = router;
```

Built-in Helpers

```
<div class="menu-container">
  <div class="card-header col-md-10">
    <h2>
      &#127869 Current Menu
    </h2>
  </div>
  <!-- We implement the built-in helper, #each. This helper will iterate over dishes -->
  {{#each dishes as |dish|}}
  <!-- For each dish in the array of dish objects, the following <div> will be rendered. -->
    <div class="dish col-md-5">
      <!-- We use dish.id instead of <od> and <li> elements. -->
      <p>
        {{dish.id}}.
        {{dish.dish_name}}
      </p>
    </div>
    <div class="col-md-7">
      <p>
        <!-- guest_name will eventually go here -->
      </p>
    </div>
    <!-- Below, we close the #each helper. -->
  {{/each}}
</div>
```

```
  <div class="dish-card">
    <div class="dish-card-header">
      {{dish_name}}
    </div>
    <div class="card-body">
      <p>
        Description:
        {{description}}
      </p>
      <!-- This is where we initiate the #if helper with has_nuts set as the condition -->
      {{#if has_nuts}}
      <!-- This code will be rendered if the condition (has_nuts) is truthy -->
        <p class="has-nuts">
          Contains Nuts
        </p>
        <!-- Below, we end the #if helper -->
      {{/if}}
    </div>
  </div>
```

```
<ul class="people_list">
  {{#each people}}
    <li>{{this}}</li>
  {{/each}}
</ul>
```

```
{{#each paragraphs}}
<p>{{this}}</p>
{{else}}
<p class="empty">No content</p>
{{/each}}
```

```
{{#each array}} {{@index}}: {{this}} {{/each}}
```

```
{{#each object}} {{@key}}: {{this}} {{/each}}
```

```
<div class="entry">
{{#if author}}
<h1>{{firstName}} {{lastName}}</h1>
{{/if}}
</div>
```

```
<div class="entry">
{{#if author}}
<h1>{{firstName}} {{lastName}}</h1>
{{else}}
<h1>Unknown Author</h1>
{{/if}}
</div>
```

```
{{#if (isdefined value1)}}true{{else}}false{{/if}}
{{#if (isdefined value2)}}true{{else}}false{{/if}}
```

```
{{#with person}}
{{firstname}} {{lastname}}
{{/with}}
```

```
{{#with city}}
{{city.name}} (not shown because there is no city)
{{else}}
No city found
{{/with}}
```

```
{{#with city as | city |}}
  {{#with city.location as | loc |}}
    {{city.name}}: {{loc.north}} {{loc.east}}
  {{/with}}
{{/with}}
```

```
{{#each people}}
    {{.}} lives in {{lookup ../cities @index}}
{{/each}}
```

```
{{log 'this is a simple log output'}}
```

```
<div class="entry">
{{#unless license}}
<h3 class="warning">WARNING: This entry does not have a license!</h3>
{{/unless}}
</div>
```

Serialization

```
const router = require("express").Router();
const Dish = require("../models/Dish");

// get one dish without serializing data
router.get("/dish/:id", async (req, res) => {
  try {
    const dishData = await Dish.findByPk(req.params.id);
    console.log(dishData);
    res.render("dish", dishData);
  } catch (err) {
    res.status(500).json(err);
  }
});

// get one dish with serialized data
router.get("/dish/:id", async (req, res) => {
  try {
    // Search the database for a dish with an id that matches params
    const dishData = await Dish.findByPk(req.params.id);
    console.log(dishData);
    // We use .get({ plain: true }) on the object to serialize it so that it only includes the data that we need.
    const dish = dishData.get({ plain: true });
    // Then, the 'dish' template is rendered and dish is passed into the template.
    res.render("dish", dish);
  } catch (err) {
    res.status(500).json(err);
  }
});

module.exports = router;
```

```
<div class="menu-container">
  <div class="card-header col-md-10">
    <h2>
      &#127869 Current Menu
    </h2>
  </div>
  <!-- For each dish in the array of dish objects,
              the following <div> will be rendered. -->
  {{#each dishes as |dish|}}
    <div class="dish col-md-5">
      <!-- We use dish.id instead of <od> and <li> elements. -->
      <p>
        {{dish.id}}.
        {{dish.dish_name}}
      </p>
    </div>
    <div class="col-md-7">
      <p>
        Guest:
        {{dish.guest_name}}
      </p>
    </div>
  {{/each}}
</div>
```

```
const router = require('express').Router();
const Dish = require('../models/Dish');

// route to get all dishes
router.get('/', async (req, res) => {
    // We find all dishes in the db and set the data equal to dishData
    const dishData = await Dish.findAll().catch((err) => {
      res.json(err);
    });
    // We use map() to iterate over dishData and
          then add .get({ plain: true }) each object to serialize it.
    const dishes = dishData.map((dish) => dish.get({ plain: true }));
    // We render the template, 'all', passing in dishes,
          a new array of serialized objects.
    res.render('all', { dishes });
    });

// route to get one dish
router.get('/dish/:id', async (req, res) => {
  try{
      const dishData = await Dish.findByPk(req.params.id);
      if(!dishData) {
          res.status(404).json({message: 'No dish with this id!'});
          return;
      }
      const dish = dishData.get({ plain: true });
      res.render('dish', dish);
    } catch (err) {
        res.status(500).json(err);
    };
});

module.exports = router;
```

Front End Logic

```
async function editFormHandler(event) {
  event.preventDefault();
  const dish_name = document.querySelector('#dish_name').value;
  const description = document.querySelector('#description').value;
  const guest_name = document.querySelector('#guest_name').value;

  // What will the value of has_nuts be if the box in the form is checked?
  // The value of has_nuts will be true if the box is checked.
  // What do we call this kind of operator?
  // We call this a ternary operator. It begins with a condition followed
        by a question mark and two code blocks separated by a :.
  const has_nuts = document.querySelector('#has_nuts:checked') ? true : false;

// window.location gives us access to the URL. We then use the .split() method
        to access the number at the end of the URL and set that equal to id.
  const id = window.location.toString().split('/')[
    window.location.toString().split('/').length - 1
  ];

  // What part of our application will handle this 'put' request?
  // The Controller will handle this 'put' request.

  const response = await fetch(`/api/dish/${id}`, {
    method: 'PUT',
    body: JSON.stringify({
      dish_name,
      description,
      guest_name,
      has_nuts,
    }),
    headers: {
      'Content-Type': 'application/json',
    },
  });

  // What happens if the response is ok?
  // If the response is ok, that means that the dish was updated successfully.
  if (response.ok) {
    document.location.replace(`/dish/${id}`);
  } else {
    alert('Failed to edit dish');
  }
}
document.querySelector('.edit-dish-form').addEventListener('submit', editFormHandler);
```

```
const router = require('express').Router();
const Dish = require('../../models/Dish');

// route to create/add a dish
router.post('/', async (req, res) => {
  try {
    const dishData = await Dish.create({
    dish_name: req.body.dish_name,
    description: req.body.description,
    guest_name: req.body.guest_name,
    has_nuts: req.body.has_nuts,
  });
  res.status(200).json(dishData)
} catch (err) {
  res.status(400).json(err);
}
});

// According to MVC, what is the role of this action method?
// This action method is the Controller. It accepts input and sends data to the Model and the View.
router.put('/:id', async (req, res) => {
  // Where is this action method sending the data from the body of the fetch request? Why?
  // It is sending the data to the Model so that one dish can be updated with new data in the database.
  try {
    const dish = await Dish.update(
    {
      dish_name: req.body.dish_name,
      description: req.body.description,
      guest_name: req.body.guest_name,
      has_nuts: req.body.has_nuts,
    },
    {
      where: {
        id: req.params.id,
      },
    });
    // If the database is updated successfully, what happens to the updated data below?
    // The updated data (dish) is then sent back to handler that dispatched the fetch request.
    res.status(200).json(dish);
  } catch (err) {
      res.status(500).json(err);
    };
});

module.exports = router;
```

```
"dependencies": {
  "dotenv": "^8.2.0",
  "express": "^4.17.1",
  "express-handlebars": "^3.1.0",
  "handlebars": "^4.7.3",
  "mysql2": "^2.1.0",
  "sequelize": "^5.21.5"
},
"devDependencies": {
  "eslint": "^7.13.0"
```

```
<div class="dish-card">
  <div class="dish-card-header">
    {{dish_name}}
  </div>
  <div class="card-body">
    <p>
      Description:
      {{description}}
    </p>
    <p>
      Guest:
      {{guest_name}}
    </p>
    {{#if has_nuts}}
      <p class="has-nuts">
        Contains Nuts
      </p>
    {{/if}}
  </div>
</div>
```

```
<!-- This is the form for updating a dish.
        According to MVC framework, what is this form responsible for? -->
<!-- This form is responsible for the UI logic,
        or the components in the View that the user will interact with. -->
<!--What isn't it responsible for? -->
<!--It is not responsible for data-related logic, requests, or business-logic. -->

<div class="edit-dish-container">
  <h2>
    Update Dish
  </h2>
  <form class="edit-dish-form">
    <label for="dish_name">
      Dish Name:
    </label>
    <br />
    <textarea id="dish_name" name="dish_name" rows="2" cols="30">
      {{dish_name}}
    </textarea>
    <br />
    <label for="description">
      Description:
    </label>
    <br />
    <textarea id="description" name="description" rows="5" cols="30">
      {{description}}
    </textarea>
    <br />
    <label for="guest_name">
      Guest Name:
    </label>
    <br />
    <textarea id="guest_name" name="guest_name" rows="2" cols="30">
      {{guest_name}}
    </textarea>
    <br />
    <br />
    <input type="checkbox" id="has_nuts" name="has_nuts" value="true" />
    <label for="has_nuts">
      Check box if this dish contains nuts
    </label>
    <br />
    <button type="submit" value="Submit">
      Submit
    </button>
  </form>
</div>
<script src="/js/edit-dish.js"></script>
```

Partials

```
<section class="card">

  <!-- The contents of the painting card links to the painting page using the painting ID -->
  <a href="/painting/{{id}}">

    <!-- The card contains the painting name as the card header -->
    <header>{{title}}</header>

    <!-- In the body of the card, the painting is rendered along with the artist name -->
    <div class="card-body">
      <img src="/images/{{filename}}" alt="{{description}}">
      <p>By: {{artist}}</p>
    </div>
  </a>
</section>
```

```
<section class="container">
  <h3 class="gallery-name">{{gallery.name}}</h3>
  <section class="flex-row justify-center">

    <!-- For each painting that belongs to this gallery,
              the painting-details partial
              is used to render each painting's data -->
    {{#each gallery.paintings as |painting|}}
    {{> painting-details}}
    {{/each}}
  </section>
</section>
```

```
<section class="card">

  <!-- The contents of this card links to the gallery page using the gallery ID -->
  <a href="/gallery/{{id}}">

    <!-- The card contains the gallery name as the card header -->
    <header>{{name}}</header>

    <!-- The card contains the first painting of the gallery -->
    <img src="/images/{{paintings.[0].filename}}" alt="{{paintings.[0].description}}">
  </a>
</section>
```

```
<section class="painting">
  <img src="/images/{{painting.filename}}" alt="{{painting.description}}">
  <div class="painting-details">
    <h3>{{painting.title}}</h3>
    <p>By: {{painting.artist}}</p>
  </div>
</section>
```

```
<!-- For each gallery, the 'gallery-details' partial is used to render the gallery data -->
{{#each galleries as |gallery|}}
{{> gallery-details}}
{{/each}}
```

Custom Helpers

```
"dependencies": {
  "dotenv": "^8.2.0",
  "express": "^4.17.1",
  "express-handlebars": "^3.1.0",
  "handlebars": "^4.7.3",
  "mysql2": "^2.1.0",
  "sequelize": "^5.21.5"
```

```
const path = require('path');
const express = require('express');
const exphbs = require('express-handlebars');

const routes = require('./controllers');
const sequelize = require('./config/connection');

// Import the custom helper methods
const helpers = require('./utils/helpers');

const app = express();
const PORT = process.env.PORT || 3001;

// Incorporate the custom helper methods
const hbs = exphbs.create({ helpers });
app.engine('handlebars', hbs.engine);

app.set('view engine', 'handlebars');
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));

app.use(routes);

sequelize.sync({ force: false }).then(() => {
  app.listen(PORT, () => console.log('Now listening'));
});
```

```
module.exports = {
  // the helper method 'format_time' will take in a timestamp
           and return a string with only the time
  format_time: (date) => {
    // We use the 'toLocaleTimeString()' method to format the time
           as H:MM:SS AM/PM
    return date.toLocaleTimeString();
  },

 // The custom helper 'format_date' takes in a timestamp
  format_date: (date) => {
    // Using JavaScript Date methods, we get and format the month, date, and year
    // We need to add one to the month since it is returned as a zero-based value
    return `${new Date(date).getMonth() + 1}/${new Date(date).getDate()}/${
      // We add five years to the 'year' value to calculate the end date
      new Date(date).getFullYear() + 5
    }`;
  },
};
```

```
<section class="container">
  <h3 class="gallery-name">{{gallery.name}}</h3>

  <!-- Without the custom helper, the entire timestamp is printed -->
  <h4>Opening Hour: {{gallery.starting_date}}</h4>

  <!-- Using a custom helper 'format_time', the time has been formatted -->
  <h4>Closing Hour: {{format_time gallery.ending_date}}</h4>
  <section class="flex-row justify-center">
    {{#each gallery.paintings as |painting|}}
    {{> painting-details}}
    {{/each}}
  </section>
</section>
```

```
<section class="painting">
  <img src="/images/{{painting.filename}}" alt="{{painting.description}}">
  <div class="painting-details">
    <h3>{{painting.title}}</h3>
    <p>By: {{painting.artist}}</p>

    <!-- The custom method 'format_date'
           is used to calculate the end date and format it correctly -->
    <p>Exhibition open until: {{ format_date painting.exhibition_date}}</p>
  </div>
</section>
```

Sessions

```
const router = require('express').Router();
const { Gallery, Painting } = require('../models');

// GET all galleries for homepage
router.get('/', async (req, res) => {
  try {
    const dbGalleryData = await Gallery.findAll({
      include: [
        {
          model: Painting,
          attributes: ['filename', 'description'],
        },
      ],
    });
    const galleries = dbGalleryData.map((gallery) =>
      gallery.get({ plain: true })
    );
    req.session.save(() => {
      // We set up a session variable to count the number of times
              we visit the homepage
      if (req.session.countVisit) {
        // If the 'countVisit' session variable already exists, increment it by 1
        req.session.countVisit++;
      } else {
        // If the 'countVisit' session variable doesn't exist, set it to 1
        req.session.countVisit = 1;
      }
      res.render('homepage', {
        galleries,
        // We send over the current 'countVisit' session variable to be rendered
        countVisit: req.session.countVisit,
      });
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});
```

```
<!-- We are tracking the number of times we visit this homepage -->
<h3>Number of times you've visited this homepage: {{countVisit}}</h3>
<section class="galleries">
  {{#each galleries as |gallery|}}
  {{> gallery-details}}
  {{/each}}
</section>
```

```
// GET one gallery
router.get('/gallery/:id', async (req, res) => {
  try {
    const dbGalleryData = await Gallery.findByPk(req.params.id, {
      include: [
        {
          model: Painting,
          attributes: [
            'id', 'title', 'artist', 'exhibition_date', 'filename', 'description',
          ],
        },
      ],
    });
    const gallery = dbGalleryData.get({ plain: true });
    res.render('gallery', {
      gallery,
      // We are not incrementing the 'countVisit' session variable here
      // but simply sending over the current 'countVisit' session variable
              to be rendered
      countVisit: req.session.countVisit,
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});

// GET one painting
router.get('/painting/:id', async (req, res) => {
  try {
    const dbPaintingData = await Painting.findByPk(req.params.id);
    const painting = dbPaintingData.get({ plain: true });
    res.render('painting', {
      painting,
      // We are not incrementing the 'countVisit' session variable here
      // but simply sending over the current 'countVisit' session variable
              to be rendered
      countVisit: req.session.countVisit,
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});

module.exports = router;
```

```
"dependencies": {
  "dotenv": "^8.2.0",
  "express": "^4.17.1",
  "express-handlebars": "^5.2.0",
  "express-session": "^1.17.1",
  "handlebars": "^4.7.6",
  "mysql2": "^2.2.5",
  "sequelize": "^6.3.5"
}
```

Sessions Solved

```
const router = require('express').Router();
const { Gallery, Painting } = require('../models');

// GET all galleries for homepage
router.get('/', async (req, res) => {
  try {
    const dbGalleryData = await Gallery.findAll({
      include: [
        {
          model: Painting,
          attributes: ['filename', 'description'],
        },],});
    const galleries = dbGalleryData.map((gallery) =>
      gallery.get({ plain: true })
    );
    // Send over the 'loggedIn' session variable to the 'homepage' template
    res.render('homepage', {
      galleries,
      loggedIn: req.session.loggedIn,
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }});

// GET one gallery
router.get('/gallery/:id', async (req, res) => {
  try {
    const dbGalleryData = await Gallery.findByPk(req.params.id, {
      include: [
        {
          model: Painting,
          attributes: [
            'id', 'title', 'artist', 'exhibition_date', 'filename', 'description',
          ],},],});
    const gallery = dbGalleryData.get({ plain: true });
    // Send over the 'loggedIn' session variable to the 'gallery' template
    res.render('gallery', { gallery, loggedIn: req.session.loggedIn });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }});

// GET one painting
router.get('/painting/:id', async (req, res) => {
  try {
    const dbPaintingData = await Painting.findByPk(req.params.id);
    const painting = dbPaintingData.get({ plain: true });
    // Send over the 'loggedIn' session variable to the 'homepage' template
    res.render('painting', { painting, loggedIn: req.session.loggedIn });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }});

// Login route
router.get('/login', (req, res) => {
  // If the user is already logged in, redirect to the homepage
  if (req.session.loggedIn) {
    res.redirect('/');
    return;
  }
  // Otherwise, render the 'login' template
  res.render('login');
});

module.exports = router;
```

```
const router = require('express').Router();
const { User } = require('../../models');

// CREATE new user
router.post('/', async (req, res) => {
  try {
    const dbUserData = await User.create({
      username: req.body.username,
      email: req.body.email,
      password: req.body.password,
    });
    // Set up sessions with a 'loggedIn' variable set to `true`
    req.session.save(() => {
      req.session.loggedIn = true;
      res.status(200).json(dbUserData);
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }});

// Login
router.post('/login', async (req, res) => {
  try {
    const dbUserData = await User.findOne({
      where: {
        email: req.body.email,
      },});
    if (!dbUserData) {
      res
        .status(400)
        .json({ message: 'Incorrect email or password. Please try again!' });
      return;
    }
    const validPassword = await dbUserData.checkPassword(req.body.password);
    if (!validPassword) {
      res
        .status(400)
        .json({ message: 'Incorrect email or password. Please try again!' });
      return;
    }
    // Once the user successfully logs in, set up the sessions variable 'loggedIn'
    req.session.save(() => {
      req.session.loggedIn = true;
      res
        .status(200)
        .json({ user: dbUserData, message: 'You are now logged in!' });
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }});

// Logout
router.post('/logout', (req, res) => {
  // When the user logs out, destroy the session
  if (req.session.loggedIn) {
    req.session.destroy(() => {
      res.status(204).end();
    });
  } else {
    res.status(404).end();
  }});

module.exports = router;
```

Sessions Solved pt.2

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Museum</title>
  <link rel="stylesheet" type="text/css" href="/css/jass.css">
  <link rel="stylesheet" type="text/css" href="/css/style.css">
</head>
<body>
  <header>
    <h1>Welcome to the Four Seasons Museum</h1>
  </header>
  <nav>
    <a href="/">Home</a>
    <!-- If the user is logged in, show a logout button -->
    {{#if loggedIn}}
    <button id="logout" class="btn-no-style">Logout</button>
    {{else}}
    <!-- Otherwise, show a link to login -->
    <a href="/login">Login</a>
    {{/if}}
  </nav>
  <main>
    <aside>
      <h2>Enjoy the galleries!</h2>
    </aside>
    <div class="gallery">
      {{{body}}}
    </div>
  </main>

  <!-- If the user is logged in, load the JS script
            for when they logout -->
  {{#if loggedIn}}
  <script src="/js/logout.js"></script>
  {{/if}}
</body>
</html>
```

```handlebars
<section class="container">
  <h3 class="gallery-name">{{gallery.name}}</h3>
  <h4>Opening Hour: {{format_time gallery.starting_date}}</h4>
  <h4>Closing Hour: {{format_time gallery.ending_date}}</h4>

  <!-- If the user is logged in, render the gallery content -->
  {{#if loggedIn}}
  <section class="flex-row justify-center">
    {{#each gallery.paintings as |painting|}}
    {{> painting-details}}
    {{/each}}
  </section>
</section>

<!-- Otherwise, provide a link to login -->
{{else}}
<a href="/login">You must login first to view the paintings</a>
{{/if}}
```

```javascript
const path = require('path');
const express = require('express');

// Import express-session
const session = require('express-session');
const exphbs = require('express-handlebars');

const routes = require('./controllers');
const sequelize = require('./config/connection');
const helpers = require('./utils/helpers');

const app = express();
const PORT = process.env.PORT || 3001;

// Set up sessions
const sess = {
  secret: 'Super secret secret',
  resave: false,
  saveUninitialized: true,
};

app.use(session(sess));

const hbs = exphbs.create({ helpers });

app.engine('handlebars', hbs.engine);
app.set('view engine', 'handlebars');

app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));

app.use(routes);
sequelize.sync({ force: false }).then(() => {
  app.listen(PORT, () => console.log('Now listening'));
});
```

Cookies

```
const path = require('path');
const express = require('express');
const session = require('express-session');
const exphbs = require('express-handlebars');

// Initializes Sequelize with session store
const SequelizeStore = require('connect-session-sequelize')
              (session.Store);

const routes = require('./controllers');
const sequelize = require('./config/connection');
const helpers = require('./utils/helpers');
const app = express();

const PORT = process.env.PORT || 3001;

// Sets up session and connect to our Sequelize db
const sess = {
  secret: 'Super secret secret',
  // Tells our session to use cookies
  cookie: {},
  resave: false,
  saveUninitialized: true,
  // Sets up session store
  store: new SequelizeStore({
    db: sequelize,
  }),
};

app.use(session(sess));

const hbs = exphbs.create({ helpers });

app.engine('handlebars', hbs.engine);
app.set('view engine', 'handlebars');

app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));

app.use(routes);

sequelize.sync({ force: false }).then(() => {
  app.listen(PORT, () => console.log('Now listening'));
});
```

```
const router = require('express').Router();
const { Gallery, Painting } = require('../models');

// GET all galleries for homepage
router.get('/', async (req, res) => {
  try {
    const dbGalleryData = await Gallery.findAll({
      include: [
        {
          model: Painting,
          attributes: ['filename', 'description'],
        },
      ],
    });
    const galleries = dbGalleryData.map((gallery) =>
      gallery.get({ plain: true })
    );
    res.render('homepage', {
      galleries,
      loggedIn: req.session.loggedIn,
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});

// GET one gallery
router.get('/gallery/:id', async (req, res) => {
  try {
    const dbGalleryData = await Gallery.findByPk(req.params.id, {
      include: [
        {
          model: Painting,
          attributes: [
            'id',
            'title',
            'artist',
            'exhibition_date',
            'filename',
            'description',
          ],
        },
      ],
    });
    const gallery = dbGalleryData.get({ plain: true });
    res.render('gallery', { gallery, loggedIn: req.session.loggedIn });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});

// GET one painting
router.get('/painting/:id', async (req, res) => {
  try {
    const dbPaintingData = await Painting.findByPk(req.params.id);
    const painting = dbPaintingData.get({ plain: true });
    res.render('painting', { painting, loggedIn: req.session.loggedIn });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});

// Login route
router.get('/login', (req, res) => {
  if (req.session.loggedIn) {
    res.redirect('/');
    return;
  }
  res.render('login');
});

module.exports = router;
```

```
const router = require('express').Router();
const { User } = require('../../models');

// CREATE new user
router.post('/', async (req, res) => {
  try {
    const dbUserData = await User.create({
      username: req.body.username,
      email: req.body.email,
      password: req.body.password,
    });
    req.session.save(() => {
      req.session.loggedIn = true;
      res.status(200).json(dbUserData);
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});

// Login
router.post('/login', async (req, res) => {
  try {
    const dbUserData = await User.findOne({
      where: {
        email: req.body.email,
      },
    });
    if (!dbUserData) {
      res
        .status(400)
        .json({ message: 'Incorrect email or password. Please try again!' });
      return;
    }
    const validPassword = await dbUserData.checkPassword(req.body.password);
    if (!validPassword) {
      res
        .status(400)
        .json({ message: 'Incorrect email or password. Please try again!' });
      return;
    }
    req.session.save(() => {
      req.session.loggedIn = true;
      res
        .status(200)
        .json({ user: dbUserData, message: 'You are now logged in!' });
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});

// Logout
router.post('/logout', (req, res) => {
  if (req.session.loggedIn) {
    req.session.destroy(() => {
      res.status(204).end();
    });
  } else {
    res.status(404).end();
  }
});

module.exports = router;
```

Middleware

```
const router = require('express').Router();
const { Gallery, Painting } = require('../models');

// Import the custom middleware
const withAuth = require('../utils/auth');

// GET all galleries for homepage
router.get('/', async (req, res) => {
  try {
    const dbGalleryData = await Gallery.findAll({
      include: [
        {
          model: Painting,
          attributes: ['filename', 'description'],
        },
      ],
    });
    const galleries = dbGalleryData.map((gallery) =>
      gallery.get({ plain: true })
    );
    res.render('homepage', {
      galleries,
      loggedIn: req.session.loggedIn,
    });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});
```

```
const withAuth = (req, res, next) => {
  // If the user is not logged in, redirect the user to the login page
  // This is directly from the `/gallery/:id` and `/painting/:id` routes
  if (!req.session.loggedIn) {
    res.redirect('/login');
  } else {
    // If the user is logged in, execute the route function
            that will allow them to view the gallery
    // We call next() if the user is authenticated
    next();
  }
};

module.exports = withAuth;
```

```
// GET one gallery
// Use the custom middleware before allowing the user to access the gallery
router.get('/gallery/:id', withAuth, async (req, res) => {
  try {
    const dbGalleryData = await Gallery.findByPk(req.params.id, {
      include: [
        {
          model: Painting,
          attributes: [
            'id',
            'title',
            'artist',
            'exhibition_date',
            'filename',
            'description',
          ],
        },
      ],
    });
    const gallery = dbGalleryData.get({ plain: true });
    res.render('gallery', { gallery, loggedIn: req.session.loggedIn });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});

// GET one painting
// Use the custom middleware before allowing the user to access the painting
router.get('/painting/:id', withAuth, async (req, res) => {
  try {
    const dbPaintingData = await Painting.findByPk(req.params.id);
    const painting = dbPaintingData.get({ plain: true });
    res.render('painting', { painting, loggedIn: req.session.loggedIn });
  } catch (err) {
    console.log(err);
    res.status(500).json(err);
  }
});
router.get('/login', (req, res) => {
  if (req.session.loggedIn) {
    res.redirect('/');
    return;
  }
  res.render('login');
});

module.exports = router;
```

Mini-Project pt.1

```
const Sequelize = require('sequelize');
require('dotenv').config();

let sequelize;

if (process.env.JAWSDB_URL) {
  sequelize = new Sequelize(process.env.JAWSDB_URL);
} else {
  sequelize = new Sequelize(
    process.env.DB_NAME,
    process.env.DB_USER,
    process.env.DB_PASSWORD,
    {
      host: 'localhost',
      dialect: 'mysql',
      port: 3306
    }
  );
}

module.exports = sequelize;
```

```
DROP DATABASE IF EXISTS crowdfund_db;
CREATE DATABASE crowdfund_db;
```

```
[
  {
    "name": "Sal",
    "email": "sal@hotmail.com",
    "password": "password12345"
  },
  {
    "name": "Lernantino",
    "email": "lernantino@gmail.com",
    "password": "password12345"
  },
  {
    "name": "Amiko",
    "email": "amiko2k20@aol.com",
    "password": "password12345"
  }
]
```

```
const sequelize = require('../config/connection');
const { User, Project } = require('../models');

const userData = require('./userData.json');
const projectData = require('./projectData.json');

const seedDatabase = async () => {
  await sequelize.sync({ force: true });
  const users = await User.bulkCreate(userData, {
    individualHooks: true,
    returning: true,
  });
  for (const project of projectData) {
    await Project.create({
      ...project,
      user_id: users[Math.floor(Math.random() * users.length)].id,
    });
  }
  process.exit(0);
};

seedDatabase();
```

```
DB_NAME=crowdfund_db
DB_PASSWORD=rootpass
DB_USER=root
```

```
[
  {
    "name": "Music Near Me",
    "description": "A mobile app that will send you
          notifications whenever a concert is playing in your area.",
    "needed_funding": 5000
  },
  {
    "name": "The Ultimate Tech Quiz",
    "description": "A web app that will give users
          10 new technical questions each day
          and track their progress in things like
          programming, cybersecurity, database architecture, and more!",
    "needed_funding": 10000
  },
  {
    "name": "Roll 'Em Up",
    "description": "A game for Windows and
          macOS where players move a ball through a series
          of increasingly challenging mazes.",
    "needed_funding": 800
  }
]
```

Mini-Project pt.2

```
const router = require('express').Router();
const userRoutes = require('./userRoutes');
const projectRoutes = require('./projectRoutes');

router.use('/users', userRoutes);
router.use('/projects', projectRoutes);

module.exports = router;
```

```
const router = require('express').Router();

const apiRoutes = require('./api');
const homeRoutes = require('./homeRoutes');

router.use('/', homeRoutes);
router.use('/api', apiRoutes);

module.exports = router;
```

```
const router = require('express').Router();
const { Project } = require('../../models');
const withAuth = require('../../utils/auth');

router.post('/', withAuth, async (req, res) => {
  try {
    const newProject = await Project.create({
      ...req.body,
      user_id: req.session.user_id,
    });
    res.status(200).json(newProject);
  } catch (err) {
    res.status(400).json(err);
  }
});

router.delete('/:id', withAuth, async (req, res) => {
  try {
    const projectData = await Project.destroy({
      where: {
        id: req.params.id,
        user_id: req.session.user_id,
      },
    });
    if (!projectData) {
      res.status(404).json
({ message: 'No project found with this id!' });
      return;
    }
    res.status(200).json(projectData);
  } catch (err) {
    res.status(500).json(err);
  }
});

module.exports = router;
```

```
const router = require('express').Router();
const { User } = require('../../models');

router.post('/', async (req, res) => {
  try {
    const userData = await User.create(req.body);
    req.session.save(() => {
      req.session.user_id = userData.id;
      req.session.logged_in = true;
      res.status(200).json(userData);
    });
  } catch (err) {
    res.status(400).json(err);
  }
});

router.post('/login', async (req, res) => {
  try {
    const userData = await User.findOne({ where: { email: req.body.email } });
    if (!userData) {
      res
        .status(400)
        .json({ message: 'Incorrect email or password, please try again' });
      return;
    }
    const validPassword = await userData.checkPassword(req.body.password);
    if (!validPassword) {
      res
        .status(400)
        .json({ message: 'Incorrect email or password, please try again' });
      return;
    }
    req.session.save(() => {
      req.session.user_id = userData.id;
      req.session.logged_in = true;

      res.json({ user: userData, message: 'You are now logged in!' });
    });
  } catch (err) {
    res.status(400).json(err);
  }
});

router.post('/logout', (req, res) => {
  if (req.session.logged_in) {
    req.session.destroy(() => {
      res.status(204).end();
    });
  } else {
    res.status(404).end();
  }
});

module.exports = router;
```

```
const router = require('express').Router();
const { Project, User } = require('../models');
const withAuth = require('../utils/auth');

router.get('/', async (req, res) => {
  try {
    // Get all projects and JOIN with user data
    const projectData = await Project.findAll({
      include: [
        {
          model: User,
          attributes: ['name'],
        },
      ],
    });
    // Serialize data so the template can read it
    const projects = projectData.map((project) => project.get({ plain: true }));
    // Pass serialized data and session flag into template
    res.render('homepage', {
      projects,
      logged_in: req.session.logged_in
    });
  } catch (err) {
    res.status(500).json(err);
  }
});

router.get('/project/:id', async (req, res) => {
  try {
    const projectData = await Project.findByPk(req.params.id, {
      include: [
        {
          model: User,
          attributes: ['name'],
        },
      ],
    });
    const project = projectData.get({ plain: true });
    res.render('project', {
      ...project,
      logged_in: req.session.logged_in
    });
  } catch (err) {
    res.status(500).json(err);
  }
});

// Use withAuth middleware to prevent access to route
router.get('/profile', withAuth, async (req, res) => {
  try {
    // Find the logged in user based on the session ID
    const userData = await User.findByPk(req.session.user_id, {
      attributes: { exclude: ['password'] },
      include: [{ model: Project }],
    });
    const user = userData.get({ plain: true });
    res.render('profile', {
      ...user,
      logged_in: true
    });
  } catch (err) {
    res.status(500).json(err);
  }
});

router.get('/login', (req, res) => {
  // If the user is already logged in, redirect the request to another route
  if (req.session.logged_in) {
    res.redirect('/profile');
    return;
  }
  res.render('login');
});

module.exports = router;
```

Mini-Project pt.3

```
const { Model, DataTypes } = require('sequelize');
const bcrypt = require('bcrypt');

const sequelize = require('../config/connection');

class User extends Model {
  checkPassword(loginPw) {
    return bcrypt.compareSync(loginPw, this.password);
  }
}

User.init(
  {
    id: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
    },
    name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    email: {
      type: DataTypes.STRING,
      allowNull: false,
      unique: true,
      validate: {
        isEmail: true,
      },
    },
    password: {
      type: DataTypes.STRING,
      allowNull: false,
      validate: {
        len: [8],
      },
    },
  },
  {
    hooks: {
      beforeCreate: async (newUserData) => {
        newUserData.password = await bcrypt.hash(newUserData.password, 10);
        return newUserData;
      },
      beforeUpdate: async (updatedUserData) => {
        updatedUserData.password = await bcrypt.hash(updatedUserData.password, 10);
        return updatedUserData;
      },
    },
    sequelize,
    timestamps: false,
    freezeTableName: true,
    underscored: true,
    modelName: 'user',
  }
);

module.exports = User;
```

```
const User = require('./User');
const Project = require('./Project');

User.hasMany(Project, {
  foreignKey: 'user_id',
  onDelete: 'CASCADE'
});

Project.belongsTo(User, {
  foreignKey: 'user_id'
});

module.exports = { User, Project };
```

```
const { Model, DataTypes } = require('sequelize');
const sequelize = require('../config/connection');

class Project extends Model {}

Project.init(
  {
    id: {
      type: DataTypes.INTEGER,
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
    },
    name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    description: {
      type: DataTypes.STRING,
    },
    date_created: {
      type: DataTypes.DATE,
      allowNull: false,
      defaultValue: DataTypes.NOW,
    },
    needed_funding: {
      type: DataTypes.FLOAT,
      allowNull: false,
    },
    user_id: {
      type: DataTypes.INTEGER,
      references: {
        model: 'user',
        key: 'id',
      },
    },
  },
  {
    sequelize,
    timestamps: false,
    freezeTableName: true,
    underscored: true,
    modelName: 'project',
  }
);

module.exports = Project;
```

Mini-Project pt.4

```
const loginFormHandler = async (event) => {
  event.preventDefault();
  // Collect values from the login form
  const email = document.querySelector('#email-login').value.trim();
  const password = document.querySelector('#password-login').value.trim();
  if (email && password) {
    // Send a POST request to the API endpoint
    const response = await fetch('/api/users/login', {
      method: 'POST',
      body: JSON.stringify({ email, password }),
      headers: { 'Content-Type': 'application/json' },
    });
    if (response.ok) {
      // If successful, redirect the browser to the profile page
      document.location.replace('/profile');
    } else {
      alert(response.statusText);
    }
  }
};

const signupFormHandler = async (event) => {
  event.preventDefault();
  const name = document.querySelector('#name-signup').value.trim();
  const email = document.querySelector('#email-signup').value.trim();
  const password = document.querySelector('#password-signup').value.trim();
  if (name && email && password) {
    const response = await fetch('/api/users', {
      method: 'POST',
      body: JSON.stringify({ name, email, password }),
      headers: { 'Content-Type': 'application/json' },
    });
    if (response.ok) {
      document.location.replace('/profile');
    } else {
      alert(response.statusText);
    }
  }
};

document
  .querySelector('.login-form')
  .addEventListener('submit', loginFormHandler);

document
  .querySelector('.signup-form')
  .addEventListener('submit', signupFormHandler);
```

```
const newFormHandler = async (event) => {
  event.preventDefault();
  const name = document.querySelector('#project-name').value.trim();
  const needed_funding = document.querySelector('#project-funding').value.trim();
  const description = document.querySelector('#project-desc').value.trim();
  if (name && needed_funding && description) {
    const response = await fetch(`/api/projects`, {
      method: 'POST',
      body: JSON.stringify({ name, needed_funding, description }),
      headers: {
        'Content-Type': 'application/json',
      },
    });
    if (response.ok) {
      document.location.replace('/profile');
    } else {
      alert('Failed to create project');
    }
  }
};

const delButtonHandler = async (event) => {
  if (event.target.hasAttribute('data-id')) {
    const id = event.target.getAttribute('data-id');
    const response = await fetch(`/api/projects/${id}`, {
      method: 'DELETE',
    });
    if (response.ok) {
      document.location.replace('/profile');
    } else {
      alert('Failed to delete project');
    }
  }
};

document
  .querySelector('.new-project-form')
  .addEventListener('submit', newFormHandler);
document
  .querySelector('.project-list')
  .addEventListener('click', delButtonHandler);
```

```
module.exports = {
  format_date: (date) => {
    // Format date as MM/DD/YYYY
    return date.toLocaleDateString();
  },
  format_amount: (amount) => {
    // format large numbers with commas
    return parseInt(amount).toLocaleString();
  },
  get_emoji: () => {
    const randomNum = Math.random();
    // Return a random emoji
    if (randomNum > 0.7) {
      return `<span for="img" aria-label="lightbulb">□</span>`;
    } else if (randomNum > 0.4) {
      return `<span for="img" aria-label="laptop">□</span>`;
    } else {
      return `<span for="img" aria-label="gear">□</span>`;
    }
  },
};
```

```
const withAuth = (req, res, next) => {
  // If the user is not logged in, redirect the request to the login route
  if (!req.session.logged_in) {
    res.redirect('/login');
  } else {
    next();
  }
};
module.exports = withAuth;
```

```
const logout = async () => {
  const response = await fetch('/api/users/logout', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
  });
  if (response.ok) {
    document.location.replace('/');
  } else {
    alert(response.statusText);
  }
};
document.querySelector('#logout')
        .addEventListener('click', logout);
```

Mini-Project pt.5

```
<!-- Loop over each object in the projects array -->
{{#each projects as |project| }}
<div class="row mb-4 project">
  <div class="col-md-5">
    <h2>
      <!-- Call helper function with triple brackets to render the emoji span as HTML -->
      {{{get_emoji}}}
      <a href="/project/{{project.id}}">{{project.name}}</a>
    </h2>
    <p>
      <span for="img" aria-label="money">□</span>
      <!-- Pass needed_funding value to the helper function -->
      <span class="dollar-amount">{{format_amount project.needed_funding}}</span>
      needed!
    </p>
    <p>Created by {{project.user.name}} on {{format_date project.date_created}}</p>
  </div>
  <div class="col-md-7">
    <p>
      {{project.description}}
    </p>
  </div>
</div>
{{/each}}
```

```
<div class="row">
  <div class="col-auto">
    <h2>Welcome, {{name}}!</h2>
  </div>
</div>
<div class="row mt-4">
  <div class="col-md-6">
    <h3>Create a New Project:</h3>
    <form class="form new-project-form">
      <div class="form-group">
        <label for="project-name">project name:</label>
        <input class="form-input" type="text" id="project-name" name="project-name" />
      </div>
      <div class="form-group">
        <label for="project-funding">needed funding ($):</label>
        <input class="form-input" type="number" id="project-funding" name="project-funding" />
      </div>
      <div class="form-group">
        <label for="project-desc">description:</label>
        <textarea class="form-input" id="project-desc" name="project-desc"></textarea>
      </div>
      <div class="form-group">
        <button type="submit" class="btn btn-primary">Create</button>
      </div>
    </form>
  </div>
  {{#if projects.length}}
  <div class="col-md-6 project-list">
    <h3>Current Projects:</h3>
    {{#each projects as |project|}}
    <div class="row mb-2">
      <div class="col-md-8">
        <h4><a href="/project/{{project.id}}">{{project.name}}</a></h4>
      </div>
      <div class="col-md-4">
        <button class="btn btn-sm btn-danger" data-id="{{project.id}}">DELETE</button>
      </div>
    </div>
    {{/each}}
  </div>
  {{/if}}
</div>
<script src="./js/profile.js"></script>
```

```
<div class="row">
  <div class="col-md-6">
    <h2>Login</h2>
    <form class="form login-form">
      <div class="form-group">
        <label for="email-login">email:</label>
        <input class="form-input" type="text" id="email-login" />
      </div>
      <div class="form-group">
        <label for="password-login">password:</label>
        <input class="form-input" type="password" id="password-login" />
      </div>
      <div class="form-group">
        <button class="btn btn-primary" type="submit">login</button>
      </div>
    </form>
  </div>
  <div class="col-md-6">
    <h2>Signup</h2>
    <form class="form signup-form">
      <div class="form-group">
        <label for="name-signup">name:</label>
        <input class="form-input" type="text" id="name-signup" />
      </div>
      <div class="form-group">
        <label for="email-signup">email:</label>
        <input class="form-input" type="text" id="email-signup" />
      </div>
      <div class="form-group">
        <label for="password-signup">password:</label>
        <input class="form-input" type="password" id="password-signup" />
      </div>
      <div class="form-group">
        <button class="btn btn-primary" type="submit">signup</button>
      </div>
    </form>
  </div>
</div>
<script src="./js/login.js"></script>
```

```
<div class="text-center">
  <h2>{{name}}</h2>
  <p>{{{get_emoji}}}</p>
  <p>{{description}}</p>
  <p>Created by {{user.name}} on {{format_date date_created}}</p>
  <p>
    <span for="img" aria-label="money">□</span>
    <span class="dollar-amount">{{format_amount needed_funding}}</span>
    needed
  </p>
  <p>
    <button class="btn btn-primary">Fund this Project!</button>
  </p>
</div>
```

Mini-Project pt.6

```
const path = require('path');
const express = require('express');
const session = require('express-session');
const exphbs = require('express-handlebars');
const routes = require('./controllers');
const helpers = require('./utils/helpers');

const sequelize = require('./config/connection');
const SequelizeStore = require('connect-session-sequelize')(session.Store);

const app = express();
const PORT = process.env.PORT || 3001;

// Set up Handlebars.js engine with custom helpers
const hbs = exphbs.create({ helpers });
const sess = {
  secret: 'Super secret secret',
  cookie: {},
  resave: false,
  saveUninitialized: true,
  store: new SequelizeStore({
    db: sequelize
  })
};

app.use(session(sess));

// Inform Express.js on which template engine to use
app.engine('handlebars', hbs.engine);
app.set('view engine', 'handlebars');

app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, 'public')));

app.use(routes);

sequelize.sync({ force: false }).then(() => {
  app.listen(PORT, () => console.log('Now listening'));
});
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Fund My Project!</title>
  <link rel="stylesheet" href="/css/jass.css">
  <link rel="stylesheet" href="/css/style.css">
</head>
<body>
  <div class="flex-column justify-space-around">
    <header class="display-flex justify-space-between align-center p-2">
      <h1>
        <span role="img" aria-label="money">□</span>
        <a href="/">Fund My Project!</a>
      </h1>
      <nav>
        <!-- Conditionally render login or logout links -->
        {{#if logged_in}}
        <a href="/profile">profile</a> |
        <button class="no-button" id="logout">logout</button>
        {{else}}
        <a href="/login">login</a>
        {{/if}}
      </nav>
    </header>
    <main class="container container-fluid mt-5">
      <!-- Render the sub layout -->
      {{{ body }}}
    </main>
    <footer>

    </footer>
  </div>
  <!-- Render script for logged in users only -->
  {{#if logged_in}}
  <script src="/js/logout.js"></script>
  {{/if}}
</body>
</html>
```

```
{
  "name": "crowd-funding",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js",
    "seed": "node seeds/seed.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.0",
    "connect-session-sequelize": "^7.0.4",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "express-handlebars": "^5.2.0",
    "express-session": "^1.17.1",
    "mysql2": "^2.2.5",
    "sequelize": "^6.3.5"
  },
  "devDependencies": {
    "eslint": "^7.12.1",
    "eslint-config-prettier": "^6.15.0",
    "prettier": "^2.1.2"
  }
}
```