BTI325 Assignment 3

Due: Sunday, Oct 30, 2022 @ 11:59 PM

Objective:

Build upon the foundation established in Assignment 2 by providing new routes / views to support adding new employees and uploading images.

NOTE: If you are unable to start this assignment because Assignment 2 was incomplete - email me for a clean version of the Assignment 2 files to start from.

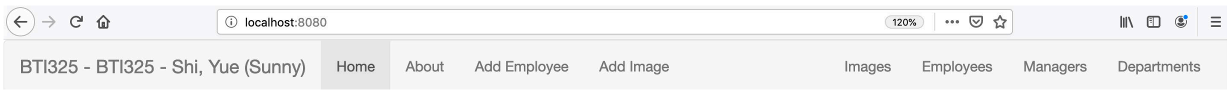
Specification:

For this assignment, we will be enhancing the functionality of Assignment 2 to include new routes & logic to handle file uploads and add employees. We will also add new routes & functionality to execute more focused queries for data (ie: fetch an employee by id, all employees by a department or manager number, etc)

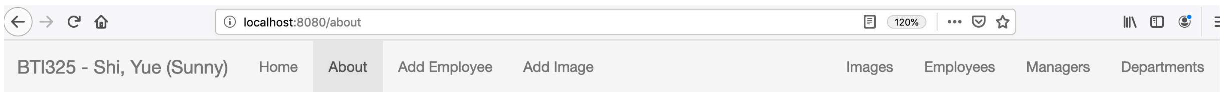
Part 1: Adding / Updating Static (.html) Files & Directories

Step 1: Modifying home.html & about.html (refer to the following screenshots)

- Open the home.html file from within the "views" folder
- Add the following two entries to the **<ul class="nav navbar-nav">** element:
    - <li><a href="/employees/add">Add Employee</a></li>
    - <li><a href="/images/add">Add Image</a></li>
- Add the following entry as the **first child** element of the **<ul class="nav navbar-nav navbar-right">** element
    - <li><a href="/images">Images</a></li>
- Your "Home" page should now have a menu bar that looks like the following:



- Update your "About" page with the same changes. When complete, it should look like the following:



Step 2: Adding new routes in server.js to support the new views

- Inside your server.js file add the following routes (HINT: do not forget __dirname & path.join):
    - GET /employees/add
        - This route simply sends the file "/views/addEmployee.html " (available below)
    - GET /images/add
        - This route simply sends the file "/views/addImage.html (available below)

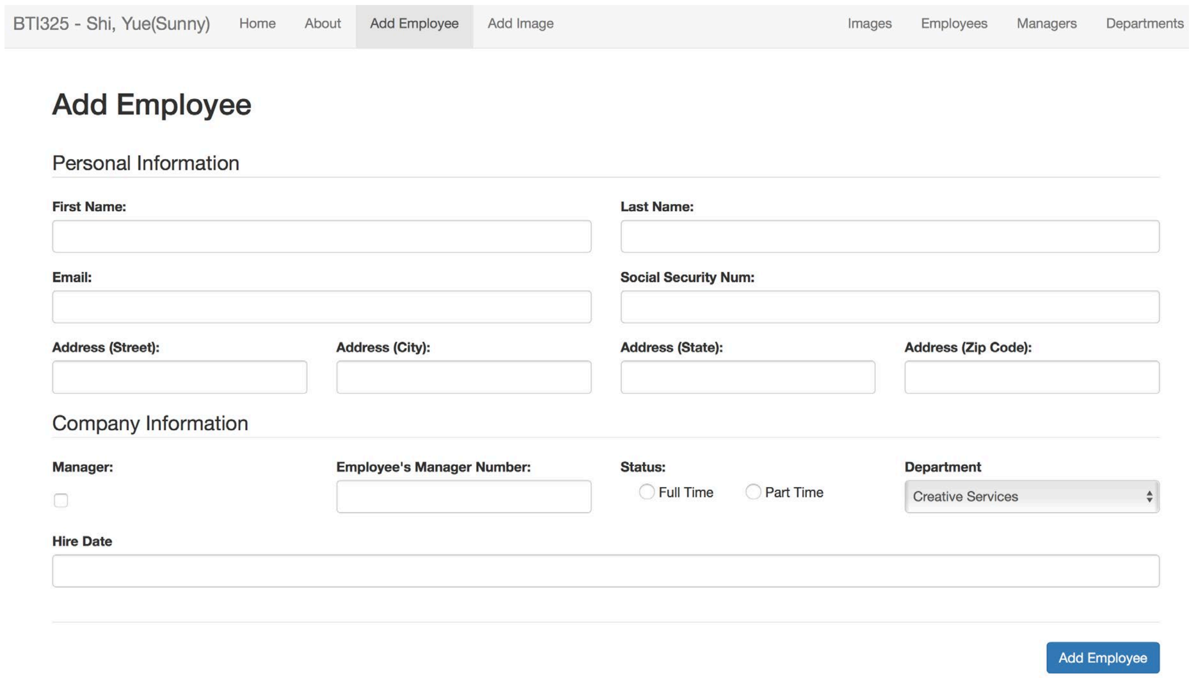
Step 3: Adding new file 1: addEmployee.html

- Create a new file in your "views" directory called "**addEmployee.html**" and open it for editing
- use the following sample html

  https://seneca-my.sharepoint.com/:u:/g/personal/sunny_shi_senecacollege_ca/EZU2zKKPzbFGky0XGz8wpeIBkc6JJ-VvaOWiwxo1bmO57A?e=8HikI2

  to reconstruct the "Add Employee" form. Change my name after <span class="navbar-brand" href="#"> with your name.

- Ensure that the "Add Employee" item in the **<ul class="nav navbar-nav"> …</ul>** element is the **only** <li> with the class "active" (this will make sure the correct navigation element is "highlighted")



Add Employee

Personal Information

First Name:
Last Name:
Email:
Social Security Num:
Address (Street):
Address (City):
Address (State):
Address (Zip Code):

Company Information

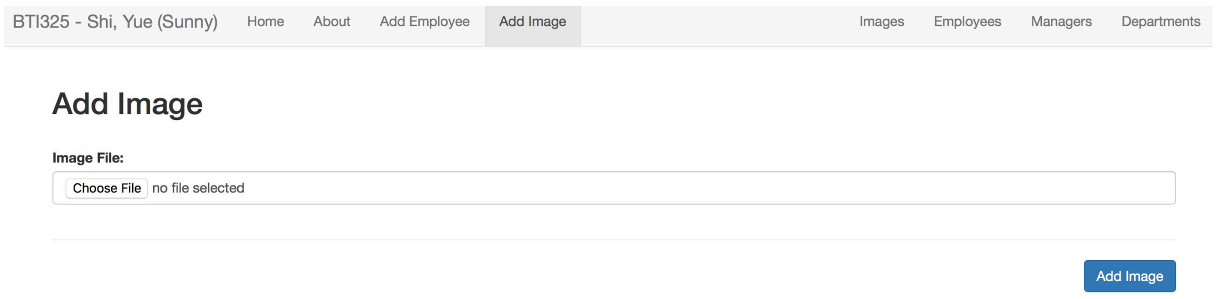
Manager:
Employee's Manager Number:
Status:
Full Time
Part Time
Department
Creative Services
Hire Date

Step 4: Adding new file 2: addImage.html

- Create a new file in your "views" directory called "addImage.html" and open it for editing
- use the following sample html

  https://seneca-my.sharepoint.com/:u:/g/personal/sunny_shi_senecacollege_ca/ESMmGXMybv9KgR-9RHjIFsgB5FVY0raNPN1i9vEa5KgSSw?e=Jg2IZm

  to reconstruct the "Add Image" form. Change my name after <span class="navbar-brand" href="#"> with your name.



Step 5: Adding a home for the uploaded Images

- Create a new folder in your "public" folder called "images"
- Within the newly created "images" folder, create an "uploaded" folder

Part 2: Adding Routes / Middleware to Support Image Uploads

Step 1: Adding multer

- Use npm to install the "multer" module
- Inside your server.js file "require" the "multer" module as "multer"
- Define a "storage" variable using "multer.diskStorage" with the following options (HINT: see "Step 5: (server) Setup…" in the week 5 course notes for additional information)
  - **destination** "./public/images/uploaded"
  - **filename** function (req, file, cb) {
    cb(null, Date.now() + path.extname(file.originalname));
    }
- Define an "upload" variable as **multer({ storage: storage });**

Step 2: Adding the "Post" route

- Add the following route:
  - POST /images/add
    - This route uses the middleware: **upload.single("imageFile")**
    - When accessed, this route will redirect to route "/images" (defined below)

Step 3: Adding "Get" route /images using the "fs" module

- Before we can add the below route, we must include the **"fs" module** in our **server.js** file (previously only in our data-service.js module)
- Next, Add the following route:
  - GET /images
    - This route will return a JSON formatted string (**res.json()**) consisting of a single "images" property, which contains the contents of the "./public/images/uploaded" directory as an array, ie { "images": ["1518109363742.jpg", "1518109363743.jpg"] }.
      HINT: You can make use of the **fs.readdir** method,
      ```
      fs.readdir("./public/images/uploaded", function(err, items){
      ```
      Note: items is the list of the content of the directory

Step 4: Verify your Solution

At this point, you should now be able to upload images using the "/images/add" route and see the full file listing on the "/images" route in the format, e.g.,: { "images": ["1518109363742.jpg", "1518109363743.jpg"] }

Part 3: Adding Routes / Middleware to Support Adding Employees

Step 1: body-parser

- In express@4.16.0, the body-parser middleware is included in express, so we don't need to install body-parser separately anymore.
- Inside your server.js file, add the following to handle form data without file upload.

```
app.use(express.json());
app.use(express.urlencoded({extended: true}));
```

Step 2: Adding "Post" route

- Add the following route:
  - POST /employees/add
    - This route makes a call to the (promise-driven) **addEmployee(employeeData)** function from your **data-service.js** module (function to be defined below). It will provide **req.body** as the parameter, ie "data.addEmployee(req.body)".
    - When the addEmployee function resolves successfully, redirect to the "**/employees**" route.

      ```
      res.redirect("/employees");
      ```

Step 3: Adding "addEmployee" function within data-service.js

- Create the function "**addEmployee(employeeData)**" within **data-service.js** according to the following specification: (**HINT**: do not forget to add it to **exports**)
  - Like all functions within data-service.js, this function must return a **Promise**
  - If **employeeData.isManager** is undefined, explicitly set it to **false**, otherwise set it to **true** (this gets around the issue of the checkbox not sending "false" if it's unchecked)
  - Explicitly set the **employeeNum** property of **employeeData** to be the **length of the "employees"** array **plus one (1)**. This will have the effect of setting the first new employee number to 281, and so on.
  - **Push** the updated **employeeData** object onto the **"employees"** array and **resolve** the promise. (you don't need to resolve any data here.)

Step 4: Verify your Solution

At this point, you should now be able to add new employees using the "/employees/add" route and see the full employee listing on the "/employees" route.

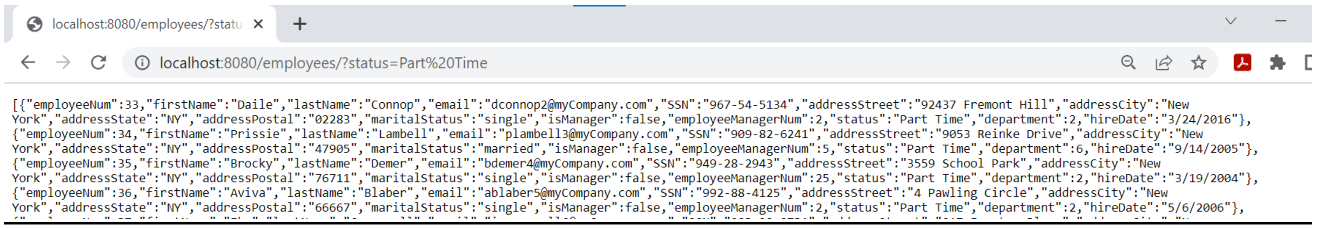
Part 4: Adding New Routes to query "Employees"

Step 1: Update the "/employees" route

- In addition to providing all of the employees, this route must now also support the following optional filters (via the **query string.** defined under the route /employees)
    - **/employees?status=*value***
        - If the query string is for **status** (i.e., **req.query.status** exists, true)
        - return a JSON string consisting of all employees where ***value*** for **status** could be either "Full Time" or "Part Time". This can be accomplished by calling the function **getEmployeesByStatus(status)** of your data-service (defined below)
        - Test with

          http://localhost:8080/employees/?status=Part%20Time

          the result includes only Part Time employees



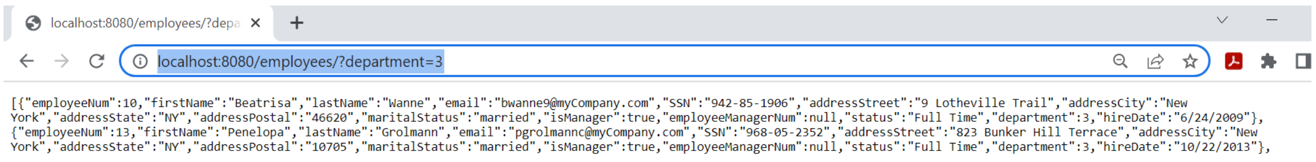
localhost:8080/employees/?status=Part%20Time

```
[{"employeeNum":33,"firstName":"Daile","lastName":"Connop","email":"dconnop2@myCompany.com","SSN":"967-54-5134","addressStreet":"92437 Fremont Hill","addressCity":"New York","addressState":"NY","addressPostal":"02283","maritalStatus":"single","isManager":false,"employeeManagerNum":2,"status":"Part Time","department":2,"hireDate":"3/24/2016"},
{"employeeNum":34,"firstName":"Prissie","lastName":"Lambell","email":"plambell3@myCompany.com","SSN":"909-82-6241","addressStreet":"9053 Reinke Drive","addressCity":"New York","addressState":"NY","addressPostal":"47905","maritalStatus":"married","isManager":false,"employeeManagerNum":5,"status":"Part Time","department":6,"hireDate":"9/14/2005"},
{"employeeNum":35,"firstName":"Brocky","lastName":"Demer","email":"bdemer4@myCompany.com","SSN":"949-28-2943","addressStreet":"3559 School Park","addressCity":"New York","addressState":"NY","addressPostal":"76711","maritalStatus":"single","isManager":false,"employeeManagerNum":25,"status":"Part Time","department":2,"hireDate":"3/19/2004"},
{"employeeNum":36,"firstName":"Aviva","lastName":"Blaber","email":"ablaber5@myCompany.com","SSN":"992-88-4125","addressStreet":"4 Pawling Circle","addressCity":"New York","addressState":"NY","addressPostal":"66667","maritalStatus":"single","isManager":false,"employeeManagerNum":2,"status":"Part Time","department":2,"hireDate":"5/6/2006"},
```

-
    - **/employees?department=*value***
        - If the query string is for **department** (i.e., **req.query.department** exists, true)
        - return a JSON string consisting of all employees where ***value*** for **department** could be one of 1, 2, 3, … 7 (there are currently 7 departments in the dataset) This can be accomplished by calling the function **getEmployeesByDepartment(department)** of your data-service (defined below)
        - Test with

          http://localhost:8080/employees/?department=3

          the result includes only departments whose departmentId is 3.



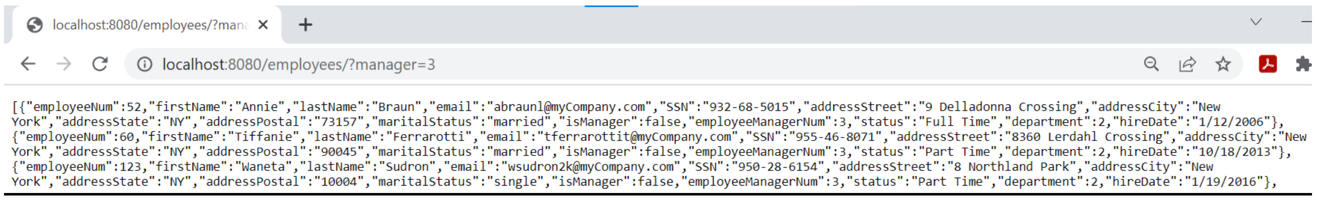
localhost:8080/employees/?department=3

```
[{"employeeNum":10,"firstName":"Beatrisa","lastName":"Wanne","email":"bwanne9@myCompany.com","SSN":"942-85-1906","addressStreet":"9 Lotheville Trail","addressCity":"New York","addressState":"NY","addressPostal":"46620","maritalStatus":"married","isManager":true,"employeeManagerNum":null,"status":"Full Time","department":3,"hireDate":"6/24/2009"},
{"employeeNum":13,"firstName":"Penelopa","lastName":"Grolmann","email":"pgrolmannc@myCompany.com","SSN":"968-05-2352","addressStreet":"823 Bunker Hill Terrace","addressCity":"New York","addressState":"NY","addressPostal":"10705","maritalStatus":"married","isManager":true,"employeeManagerNum":null,"status":"Full Time","department":3,"hireDate":"10/22/2013"},
```

- **/employees?manager=*value***
  - If the query string is for **manager** (i.e., **req.query.manager** exists, true)
  - return a JSON string consisting of all employees where ***value*** for **manager** could be one of 1, 2, 3, … 30 (there are currently 30 managers in the dataset) " - this can be accomplished by calling the function **getEmployeesByManager(manager)** of your data-service (defined below)
  - Test with

    http://localhost:8080/employees/?manager=3

    the result includes only employees whose employeeManagerNum is 3.

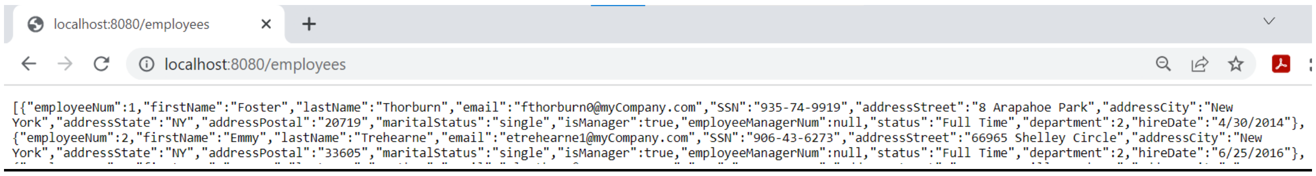

localhost:8080/employees/?man
localhost:8080/employees/?manager=3

```
[{"employeeNum":52,"firstName":"Annie","lastName":"Braun","email":"abraunl@myCompany.com","SSN":"932-68-5015","addressStreet":"9 Delladonna Crossing","addressCity":"New
York","addressState":"NY","addressPostal":"73157","maritalStatus":"married","isManager":false,"employeeManagerNum":3,"status":"Full Time","department":2,"hireDate":"1/12/2006"},
{"employeeNum":60,"firstName":"Tiffanie","lastName":"Ferrarotti","email":"tferrarottit@myCompany.com","SSN":"955-46-8071","addressStreet":"8360 Lerdahl Crossing","addressCity":"New
York","addressState":"NY","addressPostal":"90045","maritalStatus":"married","isManager":false,"employeeManagerNum":3,"status":"Part Time","department":2,"hireDate":"10/18/2013"},
{"employeeNum":123,"firstName":"Waneta","lastName":"Sudron","email":"wsudron2k@myCompany.com","SSN":"950-28-6154","addressStreet":"8 Northland Park","addressCity":"New
York","addressState":"NY","addressPostal":"10004","maritalStatus":"single","isManager":false,"employeeManagerNum":3,"status":"Part Time","department":2,"hireDate":"1/19/2016"},
```

- **/employees**
  - Otherwise, the request doesn't contain query string.
  - return a JSON string consisting of all employees without any filter (existing functionality)
  - Test with

    http://localhost:8080/employees

    the result includes all employees



```
[{"employeeNum":1,"firstName":"Foster","lastName":"Thorburn","email":"fthorburn0@myCompany.com","SSN":"935-74-9919","addressStreet":"8 Arapahoe Park","addressCity":"New
York","addressState":"NY","addressPostal":"20719","maritalStatus":"single","isManager":true,"employeeManagerNum":null,"status":"Full Time","department":2,"hireDate":"4/30/2014"},
{"employeeNum":2,"firstName":"Emmy","lastName":"Trehearne","email":"etrehearne1@myCompany.com","SSN":"906-43-6273","addressStreet":"66965 Shelley Circle","addressCity":"New
York","addressState":"NY","addressPostal":"33605","maritalStatus":"single","isManager":true,"employeeManagerNum":null,"status":"Full Time","department":2,"hireDate":"6/25/2016"},
```

Step 2: Add the "/employee/value" route

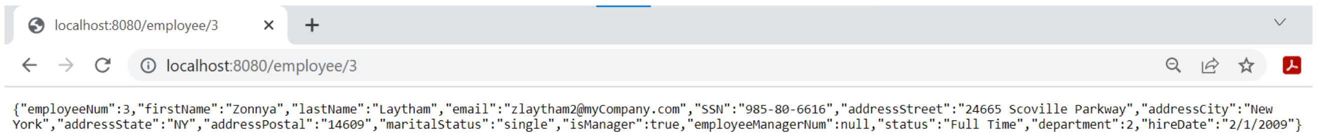
- This route (/employee) takes one **parameter** (/value) which means **employeeNum**.
- Note: this route **/employee** is singular. The previous route is **/emplyees**
- return a JSON formatted string containing the employee (**one object**) whose **employeeNum** matches the ***value***.

  For example, once the assignment is complete, **localhost:8080/employee/6** would return the manager: **Cassy Tremain -** - this can be accomplished by calling the **getEmployeeByNum(num)** function of your data-service (defined below).

- Test with

  http://localhost:8080/employee/3

  the result includes the (ONE) employee whose emplyeeNum is 3



localhost:8080/employee/3

```
{"employeeNum":3,"firstName":"Zonnya","lastName":"Laytham","email":"zlaytham2@myCompany.com","SSN":"985-80-6616","addressStreet":"24665 Scoville Parkway","addressCity":"New
York","addressState":"NY","addressPostal":"14609","maritalStatus":"single","isManager":true,"employeeManagerNum":null,"status":"Full Time","department":2,"hireDate":"2/1/2009"}
```

Part 5: Updating "data-service.js" to support the new "Employees" routes

Note: All of the following functions must return a **promise** (continuing with the pattern from the rest of the data-service.js module)

❖ getEmployeesByStatus(status)

- This function will provide an array of "employee" objects whose **status** property matches the ***status*** parameter (ie: if ***status*** is "Full Time" then the array will consist of only "Full Time" employees) using the **resolve** method of the returned promise (i.e., resolve the filtered array).
- If for some reason, the length of the array is 0 (no results returned), this function must invoke the **reject** method and pass a meaningful message, ie: "no results returned".

❖ getEmployeesByDepartment(department)

- This function will provide an array of "employee" objects whose **department** property matches the ***department*** parameter (ie: if ***department*** is 5 then the array will consist of only employees who belong to department 5) using the **resolve** method of the returned promise.
- If for some reason, the length of the array is 0 (no results returned), this function must invoke the **reject** method and pass a meaningful message, ie: "no results returned".

❖ getEmployeesByManager(manager)

- This function will provide an array of "employee" objects whose **employeeManagerNum** property matches the **manager** parameter (ie: if ***manager*** is 14 then the array will consist of only employees who are managed by employee 14) using the **resolve** method of the returned promise (i.e., resolve the filtered array).

- If for some reason, the length of the array is 0 (no results returned), this function must invoke the **reject** method and pass a meaningful message, ie: "no results returned".

❖ getEmployeeByNum(num)

- This function will provide a **single** "employee" object whose **employeeNum** property matches the ***num*** parameter (ie: if ***num*** is 261 then the "employee" object returned will be "Glenine Focke") using the **resolve** method of the returned promise (i.e., resolve the filtered **ONE object**).

- If for some reason, there is no result, this function must invoke the **reject** method and pass a meaningful message, ie: "no results returned".

~~Part 6: Deploy to Cyclic~~

- ~~Once you are satisfied with your application, deploy it to Cyclic as you did in Assignment 1.~~
- ~~**NOTE:** if you have been experimenting on Cyclic and have created multiple apps already, you may delete the unused ones. If you have received a grade for Assignment 1, it is safe to delete this app also. (login to the Cyclic, click on your app, go to "settings", … **Delete app**).~~

Part 6: Publish to Heroku

Once you are satisfied with your application, instead of deploying it to Cyclic, we deploy it to Heroku in this assignment. Steps as follows.

Instruction video of Heroku is available in **Blackboard -> weekly module -> week2 -> video**.

- You work as usual to create and finish your application locally. Then you are going to deploy it to Heroku rather than Cyclic.

- Sign up for a free account on Heroku: https://signup.heroku.com/

- Install Heroku CLI: https://devcenter.heroku.com/articles/heroku-cli#install-the-heroku-cli

  Use the following command on your Visual Studio Code terminal to check it's installed successfully.

  heroku --version

- Ensure that you have **git** installed properly (command from Visual Studio Code terminal).

  git --version

- Initialize local git repository (command from Visual Studio Code terminal).

  git init

  You'll see blue circle with a number on the left pane of your Visual Studio Code showing the number of files to commit. Click that number. Type in proper commit message. Click the "check mark" to commit. This is same as before while using Cyclic.

- **NOTE:** If, at this point, you receive the error: "Git: Failed to execute git", try executing the following commands in the integrated terminal:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

- Log in to your Heroku account. On VS Code terminal, type the following command and follow the instruction to login Heroku.

  heroku login

- Create a new app on Heroku using the command on VS Code terminal:

  heroku create

  Note: the url ending with …herokuapp.com (not the one ending with .git) is the link to access your application. Also that's the link you need to submit.

- Push your code to Heroku using the command on VS Code terminal:

  git push heroku master

  Again, you'll see information showing your app is deployed to Heroku.

  The URL ending with …**herokuapp.com** is your application.

- If you make change after publishing your app to Heroku, you need to
  - **git commit** (click the blue circle with a number on VS Code with proper message).
  - **git push heroku master**

Assignment Submission:

- Add the following declaration at the top of your **server.js** file:

  /***
  * BTI325– Assignment 3
  * I declare that this assignment is my own work in accordance with Seneca Academic Policy.
  No part of this assignment has been copied manually or electronically from any other source.
  * (including 3rd party web sites) or distributed to other students.
  *
  * Name: ______________________________ Student ID: _______________ Date: ________
  *
  * **Your app's URL (from ~~Cyclic~~ Heroku) that I can click to see your application:**
  * ____
  *
  ***/

- TEST & Place your app's URL (from ~~Cyclic~~ **Heroku**) on Blackboard as text submission
- If your app doesn't work, please DON'T include this ~~Cyclic~~ **Heroku** URL. Put a note on Blackboard submission mentioning your problems instead.
- Submit the following zip file.
- Compress (.zip) your bti325-app folder. Name it as **a3_yourName.zip** and submit **a3_yourName.zip** to Blackboard under **Assignments** -> **A**3.

- Late submission will be penalized with 10% of this assignment marks for each school day up to 5 school days, after which it will receive 0 marks.