CSCI 4131 – Internet Programming
Homework Assignment 6

Due Friday April 15th at 11:59pm

Late Submissions (with Penalty) accepted until 12pm (that is, 12 noon) on April 17th

Description

If your Assignment 5 is not complete, please come to office hours to review the solutions to the features you could not implement.

Assignment 5 provided an introduction to web-development using Node.js. This assignment will build upon what you have developed with assignment 5. The objective of this assignment is to develop a basic website using Express. Express is an application framework that simplifies the development of node.js server-side applications, and it is the most widely used application framework for doing so. Typical features of Express are:

- routing: a way to map URLs and http verbs to code paths
- easy methods for parsing http requests and building http responses:

*The following are **some of the resources** you should use to familiarize yourself with Express:*

- Essential
  - Installing Express
  - Hello world example of Express
  - Basic routing in Express
  - Serving static files in Express
- Additional Referenced
  - Express website
  - Books and blogs
  - FAQ
  - Routing in Express
  - API Reference

This assignment will also introduce you to SQL and the MySQL database.

The following are resources you should review to get familiar with SQL, MYSQL and MYSQL/Node.js

- ➢ zyBook Chapter 11, Sections 1 through 8, and Section 15
- ➢ https://www.w3schools.com/sql/
- ➢ https://www.w3schools.com/sql/sql_ref_mysql.asp
- ➢ https://www.w3schools.com/nodejs/nodejs_mysql.asp
- ➢ Optional Text: SQL/MYSQL: Chapter 13 Sebesta

Preparation and Provided Files

I. The first step will be to get Node.js and MySQL running on CSE lab machines. This can be accomplished as follows:

1. Log into a CSE lab machine remotely (by SSH or VOLE).
2. Most of the CSE lab machines run version 12.18.3 (or similar version) of Node.js.
3. To display the currently installed version of node.js on a CSELabs computer, type the following command to check the availability and the version

   ```
   node -v
   ```
4. To use the MYSQL database, you will need a database id and password. Your MYSQL database id and numeric password can be found on the class Canvas site, with your grades, in the assignment named: **MySQL Database Information**. Click on the item to view the comments. Your database id is listed after the comment: "User and Database Name" – it is an alphanumeric value that begins with the letter "C". Your password is the numeric value listed after the word: "Password".

5. At the terminal, type the following command to login to MySQL in order to ensure you can successfully access the database:

   ```
   mysql -udatabase_id -hcse-mysql-classes-01.cse.umn.edu -P3306 -p database_id
   ```

   Replace database_id with the database id provided to you before hitting enter. For the day section, your database_id will be in the format: C4131S22UXXX
   For the night section, your database_id will be in the format: C4131SN22UXXX

6. After successfully logging in, you should see **mysql>** prompt. Enter the NUMERIC password provided to you after the prompt.

II. The second step is to create a Node.js (Express) project for this assignment. This can be accomplished as follows:

1. Open the terminal on a CSE lab machine.

2. Create a directory named <x500id_hw06> by typing the following command:

   ```
   mkdir yourx500id_hw06
   ```

3. Go inside the directory by typing the command: `cd yourx500id_hw06`

4. Creating a file named *package.json* in a Node.js project makes it easy to manage module dependencies and makes the build process easier. To create the *package.json* file, type the following command: `npm init`

5. The *npm init* command will prompt you to enter the information. Use the following guidelines to enter the information (The items that you need to enter are in **bold** font. Some fields can be left blank.):

```
name: (yourx500id_hw08) yourx500id_hw06

version: (1.0.0) <Leave blank>

description: Assignment 6

entry point: (index.js) <Leave blank> (You will create an index.js file
for your use)

test command: <Leave blank>

git repository: <Leave blank>

keywords: <Leave blank>

author: yourx500id

license: (ISC) <Leave blank>
```

6. After filling in the above information, you will be prompted to answer the question: "`Is this ok? (yes)`". Type **yes** and hit enter.

7. Listing all the available files in the directory should display the following:

```
-rw------- 1 yourid CS-Grad 209 Nov 11 17:33 package.json
```

8. Install Express by typing the following command:

```
npm install --save express
```

9. You can use any npm module that you deem fit for this assignment. The npm modules that will be useful for this assignment ***and should be installed*** are:

   - mysql (`npm install --save mysql`)
   - body-parser (`npm install --save body-parser`)
   - express-session (`npm install --save express-session`)
   - additional modules you should employ are noted in the discussion that follows.

10. You are free to decide your own project structure for this assignment.

NOTE: We have provided a sample file for the node.js/express server (named: index.js) which can be used for reference, or as the basis for your solution.

III. Database setup:

1. The following files have been provided to you for this assignment:
   - create_accounts_table.js
   - insert_into_accounts_table.js
   - create_contact_table.js
2. Download these files and move them to **`yourx500id_hw06`** directory.

3. Edit each of the files to include your database id and numeric password, which you can find on Canvas in your grades in the comments portion of the column named: **MySQL Database Information**
4. Set the permissions on the files (create_accounts_table.js, insert_into_accounts_table.js, create_contact_table.js) to rwxr-xr-x (e.g., **`chmod 755 filename`**)
5. At the terminal, type the following command to create the MySQL table: **tbl_accounts**

   ```
   node create_accounts_table.js
   ```

   This table will be used to store your encrypted login credentials.

6. At the terminal, type the following command to insert values for acc_name, acc_login, acc_password into tbl_accounts table:

   ```
   node insert_into_accounts_table.js
   ```

   You will use the values chosen for acc_login and acc_password to login to the website. Keep the values in a safe place and do not share them with anyone.

7. At the terminal, type the following command to create the MySQL table: **contact_table**

   ```
   node create_contact_table.js
   ```

   This table will be used to store the details of the events.

At this point you are ready to start the website development.

3 Functionality

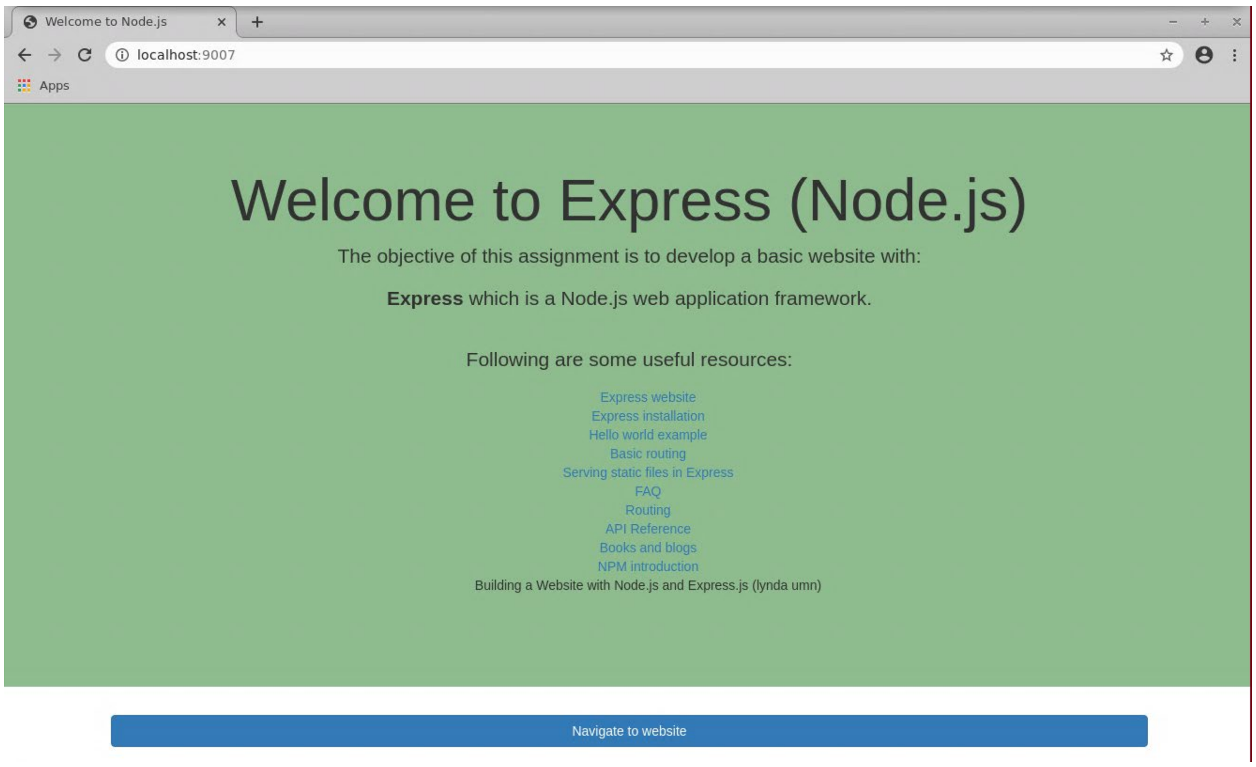
Your website will have 6 pages:

- A **Welcome** Page (provided)
- A **Login** page
- An **All Contacts** page
- A **My Contacts** Page
- An **Add Contact** page
- A **Stock** page (provided)

The All *Contact*, *My Contact*, *Add Contact*, and *Stock* pages should have a navigation bar with a logout button.
NOTE: For this assignment you will need to develop the entire website including frontend (HTML pages, CSS, Javascript) and most of the backend in node.js/Express server. *You should be able to reuse the MyContacts.html page and AddContact.html page from assignment 5, and we have provided the file index.js as a reference for your node.js/Express server.*

The following pages specify the functionality provided, and the functionality you must develop.

Welcome Page


Welcome to Node.js

localhost:9007

Apps

Welcome to Express (Node.js)

The objective of this assignment is to develop a basic website with:

Express which is a Node.js web application framework.

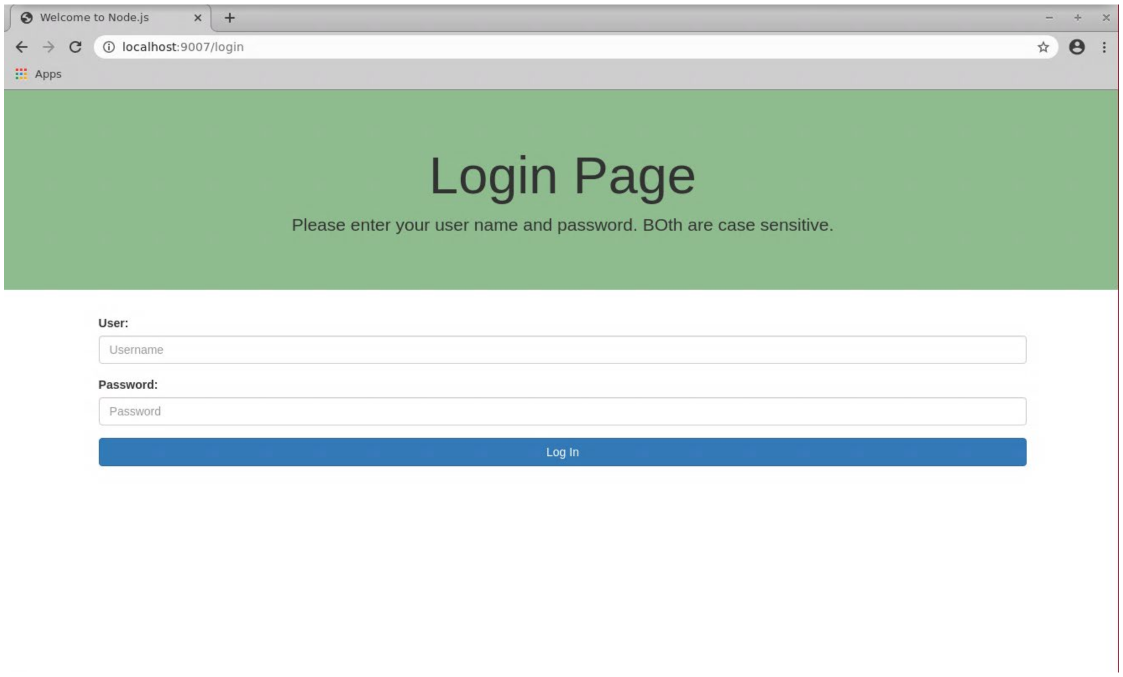
Following are some useful resources:

Express website
Express installation
Hello world example
Basic routing
Serving static files in Express
FAQ
Routing
API Reference
Books and blogs
NPM introduction
Building a Website with Node.js and Express.js (lynda umn)

Navigate to website

- The Welcome page is already provided to you and should be displayed when the default route "/" is called.
- When you click on the Navigate to website button, the /login route on the Express (Node.js) will be called. You need to develop all the remaining functionality. (See the description for the login page, and for the specification of the behavior of the /login route on the next page).

Login page



Login Page

Please enter your user name and password. BOth are case sensitive.

User:

Username

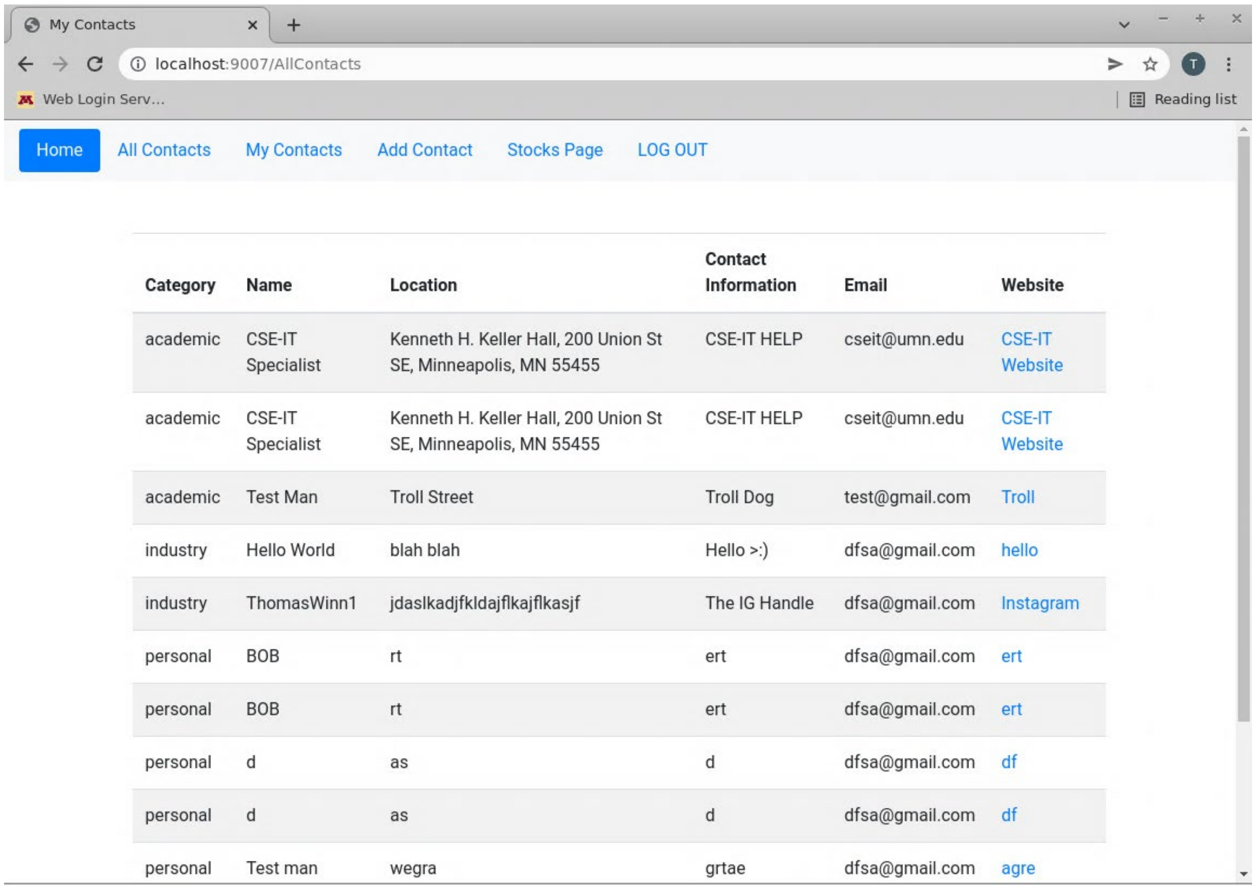
Password:

Password

Log In

- If the login page is accessed by a user that has already logged in (that is, the session variable is set), the server should return the "All Contacts" page – otherwise, the server should return the Login Page (which you are responsible for constructing).
- The Login page should have a form with two fields: "User", and "Password"
- Both of these fields are mandatory (that is, required).
- When the submit button is clicked, an AJAX request carrying the value entered for "User" and "Password" should be sent to the server for user credentials validation before allowing further access to the website. (You determine the route on the server
- The server will validate the values obtained from the form against the *acc_login*, and *acc_password* fields stored in *tbl_accounts*. The Server should compare the bcrypt password-hash of the password string it obtains from the form with the one stored in the database table `tbl_accounts`.
  - ***You should*** *use the bcrypt module from node.js - see section 11.15 in your zybook, and the link: https://www.npmjs.com/package/bcryptjs*
- Upon successful validation, server should
  - Create a user session (*Hint: you can use express-session module*).
  - Send a response back to the client indicating successful validation.
- If the validation fails, server should:
  - Send a response back to the client indicating validation failure.
- If successful response is received from server, user should be routed to "All Contacts" page, otherwise an appropriate error message should be displayed to the user (Check the screenshots towards the end of this assignment for the required behavior.)

All Contacts page

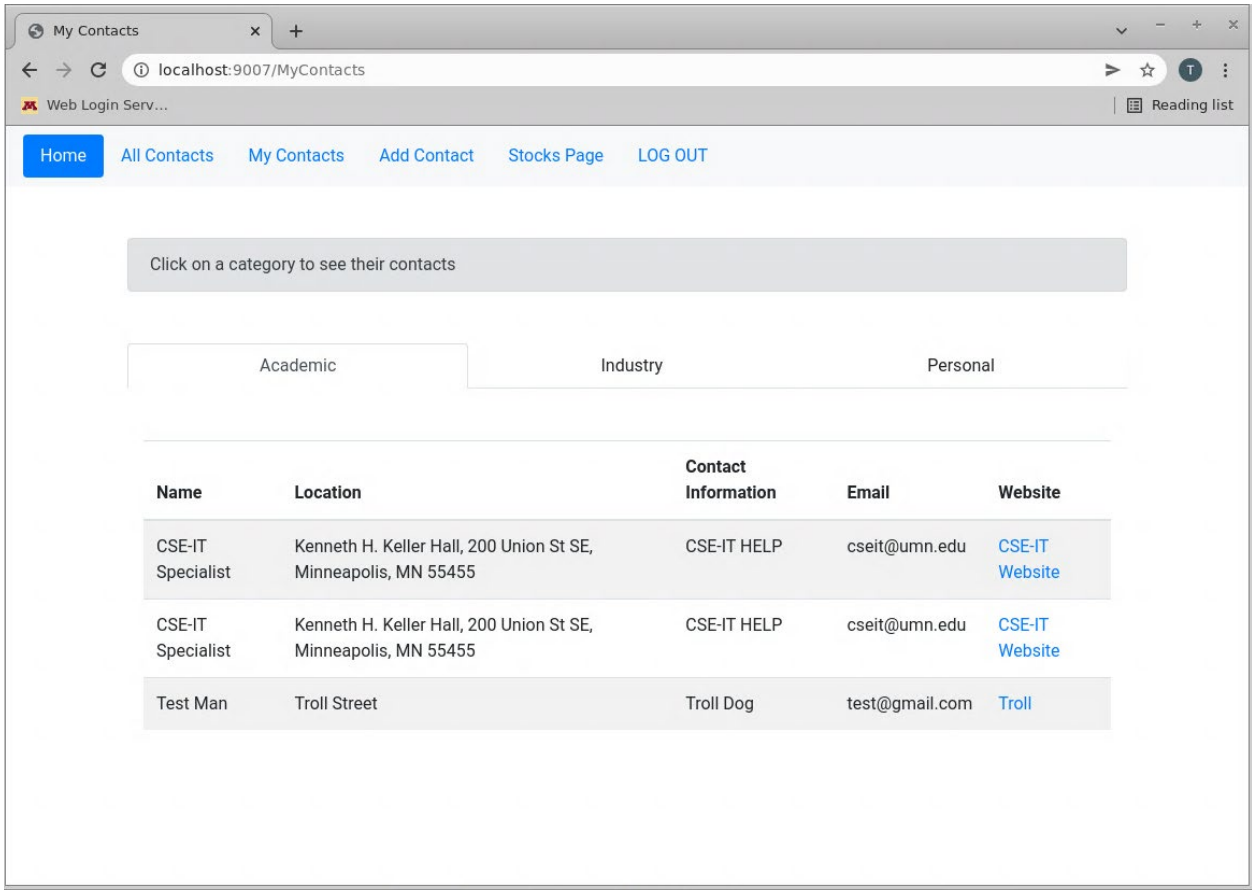


Category	Name	Location	Contact Information	Email	Website
academic	CSE-IT Specialist	Kenneth H. Keller Hall, 200 Union St SE, Minneapolis, MN 55455	CSE-IT HELP	cseit@umn.edu	CSE-IT Website
academic	CSE-IT Specialist	Kenneth H. Keller Hall, 200 Union St SE, Minneapolis, MN 55455	CSE-IT HELP	cseit@umn.edu	CSE-IT Website
academic	Test Man	Troll Street	Troll Dog	test@gmail.com	Troll
industry	Hello World	blah blah	Hello >:)	dfsa@gmail.com	hello
industry	ThomasWinn1	jdaslkadjfkldajflkajflkasjf	The IG Handle	dfsa@gmail.com	Instagram
personal	BOB	rt	ert	dfsa@gmail.com	ert
personal	BOB	rt	ert	dfsa@gmail.com	ert
personal	d	as	d	dfsa@gmail.com	df
personal	d	as	d	dfsa@gmail.com	df
personal	Test man	wegra	grtae	dfsa@gmail.com	agre

- The "Home" button in the navigation bar will redirect the user to the Welcome page.
- If a user tries to access this page without a valid login, the user should be routed to the "**Login**" page.
- The page should have a navigation bar with a logout button.
- The table in this page should be dynamically populated.
- Add a "Category" column
- To achieve this, the server should provide a GET API which returns the list of contacts. This API will be very similar to the one developed in assignment 5. It will get the list of events by querying the table: *tbl_events*.
- The client will call this API and populate the table using the data received from the server.

Note: you need to make sure the contacts in the All Contacts Page are sorted in ascending order (by their category AND name as shown on the screenshots). *Recall, you can obtain the data from the database and sort the data with a SQL query!*

My Contacts Page



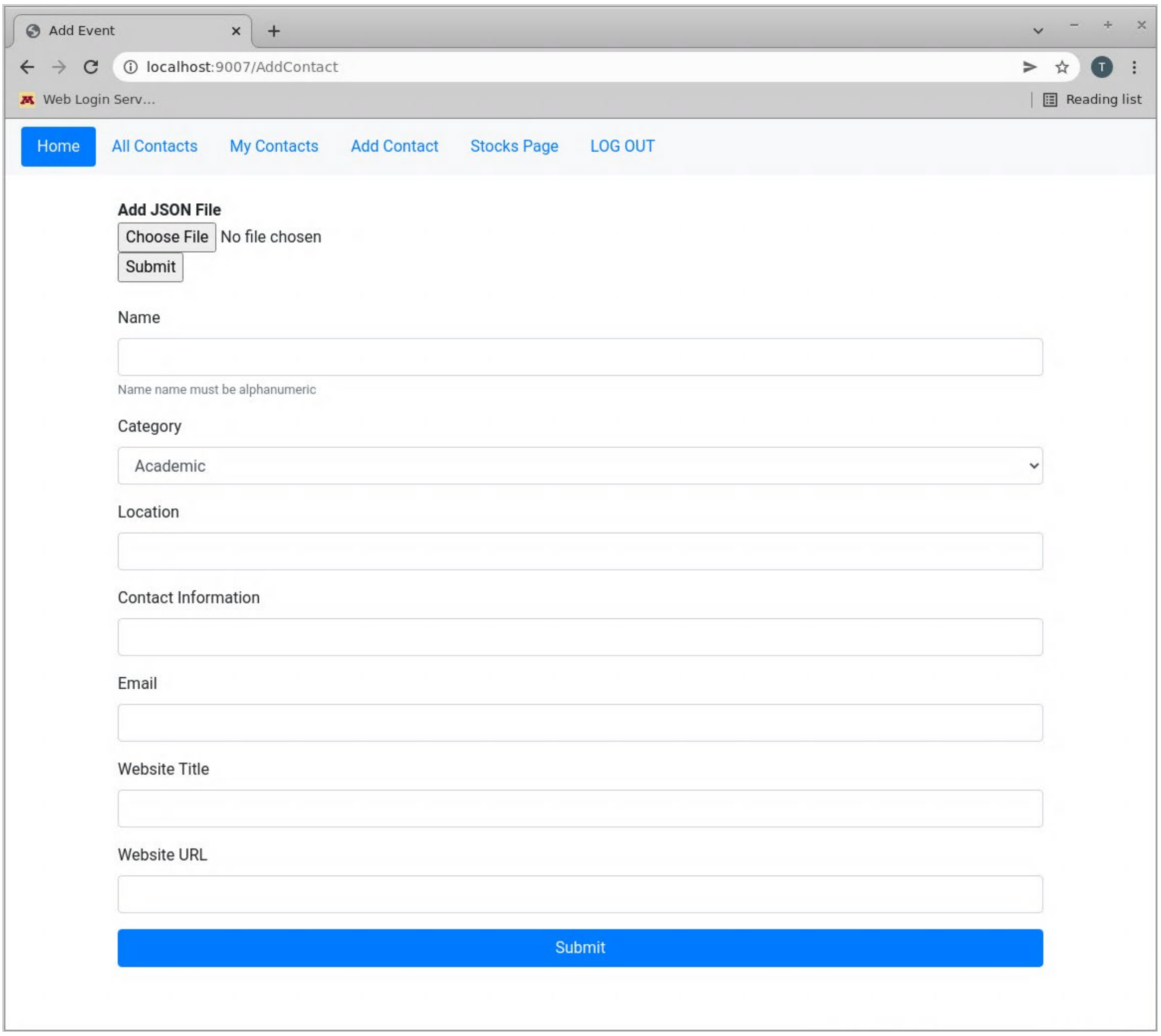
Click on a category to see their contacts

Name	Location	Contact Information	Email	Website
CSE-IT Specialist	Kenneth H. Keller Hall, 200 Union St SE, Minneapolis, MN 55455	CSE-IT HELP	cseit@umn.edu	CSE-IT Website
CSE-IT Specialist	Kenneth H. Keller Hall, 200 Union St SE, Minneapolis, MN 55455	CSE-IT HELP	cseit@umn.edu	CSE-IT Website
Test Man	Troll Street	Troll Dog	test@gmail.com	Troll

Also, this page should be very similar to the one developed in assignment 5. But this time the server will get the contacts from a database instead of from a local json file. The screenshot shows the My Contacts page after clicking "Academic".

Note: you need to make sure the contacts in My Contacts Page are sorted (in ascending order, by their name, as shown on the screenshots). Remember, SQL can be used to accomplish this…

Add Contacts page



Add JSON File

No file chosen

Name

Name name must be alphanumeric

Category

Academic

Location

Contact Information

Email

Website Title

Website URL

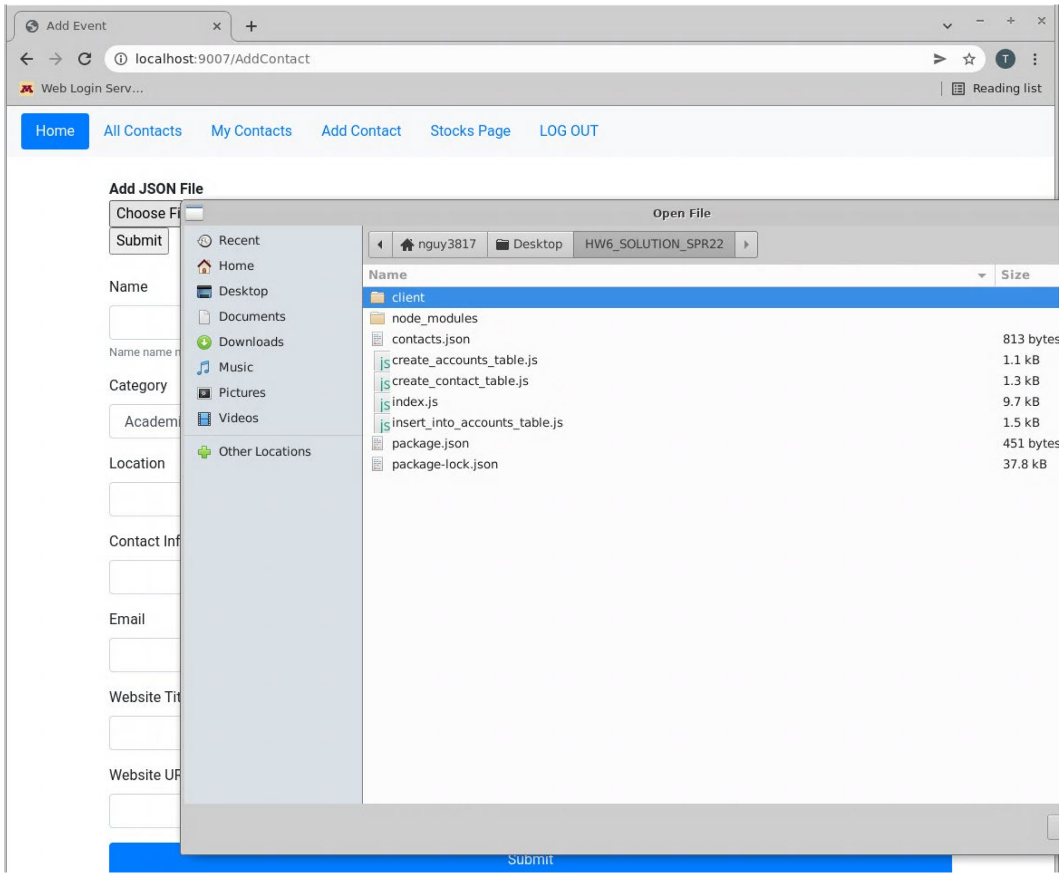
- You can use the form developed in the earlier assignments for the 'Add Contact' page, but server will insert the data into your MySQL database via a SQL query rather than storing it in the file contacts.json
- If this page is accessed without a valid login, the user should be routed to the "**Login**" page.
- The page should have a navigation bar with a logout button.
- Upon clicking submit, the form data should be posted to the server.
- The server should insert the received data into the following table: *contact_table* (*Hint: you can use mysql module*)
- The mapping between form fields and table columns could be (you can modify this mapping if you want):
  - category: contact_category
  - name: contact_name

    - location: contact_location
    - info: contact_info
    - email: contact_email
    - website_title: website_title
    - url: website_url

- Upon successful insertion, the server should return a redirect response to the browser to display the "All Contacts" page (as done with Homework assignment 5)

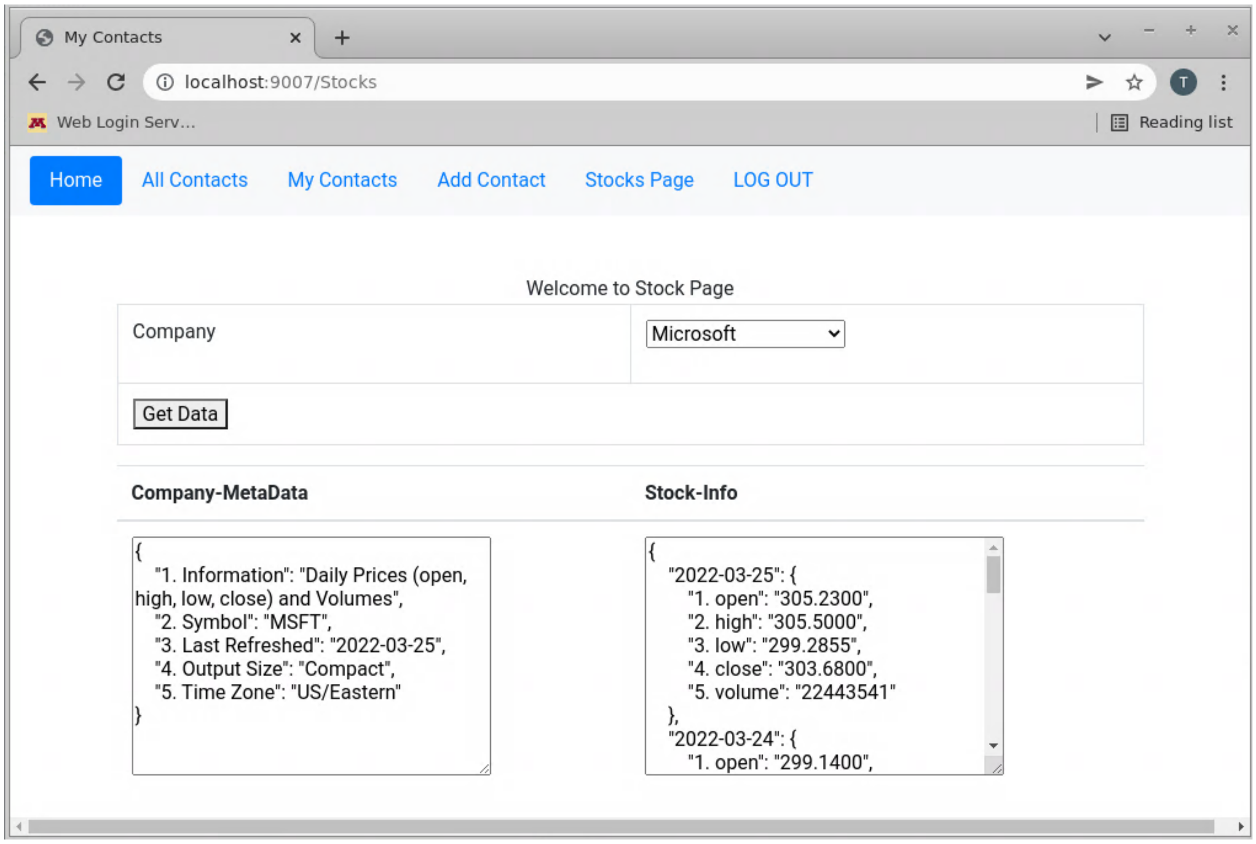
- **(BONUS)** If you want, you can add a "addJSON" (or whatever you want to call it) widget for that enables you to import your data stored in contacts.json from Programing Assignment 5 to your MySQL database. It should function as follows:

    - When the "Choose File" button is clicked, it should let you pick a file from your local file system, and after you select the contacts.json from Programming Homework Assignment 5, clicking "Submit" will insert all the contacts stored in contacts.json into your MySQL database, and then redirect you to the "All Contacts" page.
      (*Hint: you can npm install formidable to parse form fields*)
      https://www.npmjs.com/package/formidable



Name	Size
client	
node_modules	
contacts.json	813 bytes
create_accounts_table.js	1.1 kB
create_contact_table.js	1.3 kB
index.js	9.7 kB
insert_into_accounts_table.js	1.5 kB
package.json	451 bytes
package-lock.json	37.8 kB

(BONUS) Stock page (this screenshot was taken after “Get Data” is “clicked”)



```
{
  "1. Information": "Daily Prices (open, high, low, close) and Volumes",
  "2. Symbol": "MSFT",
  "3. Last Refreshed": "2022-03-25",
  "4. Output Size": "Compact",
  "5. Time Zone": "US/Eastern"
}
```

```
{
  "2022-03-25": {
    "1. open": "305.2300",
    "2. high": "305.5000",
    "3. low": "299.2855",
    "4. close": "303.6800",
    "5. volume": "22443541"
  },
  "2022-03-24": {
    "1. open": "299.1400",
```

- If this page is accessed without a valid login, the user should be routed to the “**Login**” page.
- The page should have a navigation bar with a logout button.
- Visit https://www.alphavantage.co/ and create a free API key.
- On the button “Click”, use AJAX to do a `GET` request to the `TIME_SERIES_DAILY` endpoint of the alphavantage API for the company that was selected in the dropdown list box (select).
- Display the generated JSON response in `textarea` elements as indicated in the screenshot below. Half of the container will be meta-data on the company while the other half will be the daily stock information.
- You can use jQuery or JavaScript (or a mix of both) to achieve this.

Logout button

Upon clicking the logout button on the menu-bar (pictured below) , the session should be destroyed and the server should send a redirect message to the browser to display the **Login** page. Then, the Login page should be correctly displayed by the Browser.

Submission Instructions

PLEASE ENSURE TO TEST YOUR CODE ON CSE LAB MACHINES.

You will need to submit all the files used to develop the website. This includes all the HTML, CSS, JavaScript, package.json, index.js and any other files.

Towards this end, make a copy of your working directory: `yourx500id_hw06`. Rename the copied folder as **yourx500id_express**.

Create a README file inside yourx500id_express directory. This README file should include: Your x500id, and your acc_login, and acc_password values from your fil :
`insert_into_accounts_table.js`
Finally, compress (e.g., tar or zip) the **yourx500id_express** directory and submit it **via Canvas.**

We will use the acc_login and acc_password values to login to your website. Ensure that these values are correct and can be used to access your website.

Please remove the node_modules/ folder from your submission! Do not leave it in your submission!

Evaluation

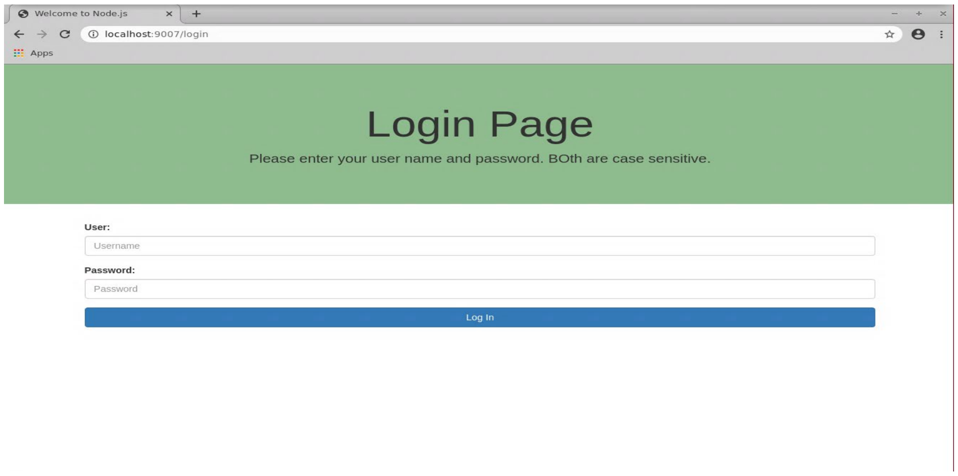
Your submission will be graded out of 100 points on the following items:

- Submission instructions are met. **Remember to remove the *node_modules* folder (4 points, yes, 4 points!)**
- The "All Contacts" and “My Contacts” and "Add Contacts" pages - and, if implemented, “Stock” page of your website redirect the user to "Login" page automatically before authentication (that is, if a user has not yet successfully logged in). **(10 points)**
- The "Login" page shows the fields required for login and the submit button. **(5 points)**
- The login page uses AJAX to send user credentials (id and password) to the server, and the server correctly redirects the browser to the “My Contacts” page after successful login validation by the server. (check the additional screenshots) **(10 points)**
- If server login validation fails, an error message is displayed on the "Login" page and the browser displays the login page and an error message. **(10 points)**
- After successful login, the "All Contacts" page displays correctly. **(5 points)**
- The "All Contacts" page gets the list of contacts from the server (which the server gets from the database). The contacts are correctly and dynamically added to the table, and correctly displayed in the user’s browser. **(10 points)**
- The "My Contacts" page gets the list of contacts from the server (which the server gets from the database). The contacts are correctly and dynamically added to the table, and correctly displayed in the user’s browser. **(10 points)**
- The contacts shown on “All Contacts” and “My Contacts” pages are correctly sorted. **(6 points)**
- The user can correctly add a new contact to the database using the form present in the "Add Contacts" page. **(10 points)**

- When a new contact is added through the “Add Contact” page, the user is redirected to the "All Contacts" page, and the user’s Contacts are correctly displayed. **(5 points)**
- The “All Contacts” and “My Contacts” and “Add Contact” pages - and, if implemented, “Stock” page - display and have an operational navigation bar that functions correctly. **(5 points)**
- The logout functionality works correctly. **(10 points)**
- **The addJSON functionality works correctly. (10 BONUS)**
- **Stock Page works correctly on “click” of ‘Get Data’ (15 BONUS)**

Additional Screenshots (Below, and on the following pages)

Login Page



Login Page

Please enter your user name and password. BOth are case sensitive.

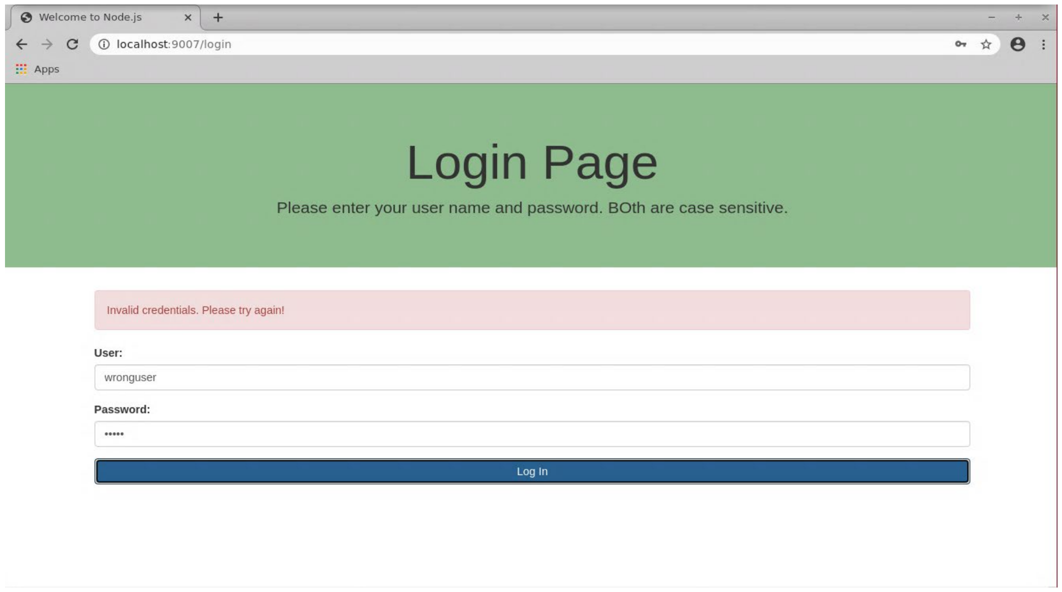
User:

Username

Password:

Password

Login Page after Invalid Login



Login Page

Please enter your user name and password. BOth are case sensitive.

Invalid credentials. Please try again!

User:

wronguser

Password:

•••••