# ECE 477 Extra Documentation

**Web Stuff** - Flow of the Program

- Get the query string from the URL
    - If no query string
        - load login page; NO failure
    - if query string contains too few parameters or failure message JSON
        - load login page; YES failure
    - else success message JSON
        - get account
        - get temperature list
        - load temperature page

## Parsing a JSON

**API Stuff - Basic Info:**

- Custom *struct* objects for each JSON object in our API.
- Custom *ENUM* to refer to which JSON type we are handling
- Custom Functions to assign the variables from a *parseObj* to the *struct* object, essentially constructors.

- **connect.c** → gets a JSON from the API
    - Synopsis: connects to the API using the correct URL and HTML headers to get a JSON
    - The URL contains the following:

        - **Host:** avarizia.duckdns.org:####/temp_cgi
        - **Port:** 8080
        - **Method:** ex. /users/login
        - **Parameters:** ex. username and password
    - The JSON is obtained from the function as a String. In the program, there were segmentation faults when trying to access the String after it was returned from the function. However, we fixed this by passing a reference to a String as a parameter to the function, and set the parameter to a memory allocated copy of the string. This allowed us to access the String with that parameter after the function is called, thus acting as a return statement.
- **util.c** → contains a series of functions to perform operations on strings
    - Moreover just an extension to json.c for better organization.
- **json.c** → Parsing JSONs into objects using the function *parseJson()*
    - Synopsis: returns a *struct* corresponding to the type of JSON being parsed.
        - Parameters:
            - *ENUM* to recognize the type of JSON we are parsing
            - the JSON String to parse
            - a void pointer pointing to the memory location of where we are storing the *struct* object.
        - In order to use the *struct* after being returned, it must be cast into the corresponding *struct object*. Thus return the memory address so we can reassign the pointer to the return value, with a type cast. i.e.:
            - *v = malloc(sizeof(struct type));*
              *v = (struct type) parseJson(ENUM, JSON, v);*

- Switch statement corresponding to which JSON we are parsing, matching the given *ENUM*.
- Call *parseString()* to get a list of the element values in the JSON
  - remove curly brackets '{' '}' at the start and end of the String
  - split the string at ',' to get each element using our custom function *splitString()*. This gives us a linked list of strings of object type *struct strItem*, using pointers.
  - get the value from each of the String elements by getting the string to the right of the ':'. This is done with the function *getMember()*. These strings are stored in an object of type *struct parseObj*.
  - return the *struct parseObj*.
- Use the corresponding function to map each value to the corresponding *struct* to create the object. These functions also map the type as well, i.e. *userId* is an integer, so we convert it to an *int* in these functions when mapping. This is what we store in the given memory pointer parameter.

- **json.c** continued → Parse a JSON array into the *struct* object.
  - The temperature list JSON has a slightly different format than the others since it is a JSONArray. Thus the *struct* we have referring to it is a linked list.
    - remove the square brackets '[' ']' at the start and end of the String.
    - Split the string at all ','s outside of JSONs by ignoring ','s inside curly brackets '{' '}'. This uses the same function *splitString()*.
    - Parse each JSON using the method above, while building the linked list by allocating memory for the next object after each parse.

- In this function when we allocated memory, for some reason each list element became disconnected, thus we used a second pointer to keep track of the previous element and rebuild the connection at each step.