Computer and Society:
To-do-list (Phase2)

Names and IDs:

1- Bassent Mostafa Saad Zaghloul Ahmed Abozeid (24)
2- Mira Samir Ragheb (81)

About the project (phase2):

- ### How our code is organized:

By now we have finished creating the front end of the 2DoList web application.

In this phase of the project, we completed the design and implementation of the back end (server-side) of our 2DoList, made minor updates to the front end, and deployed our back end. We will test this phase together with your front end.

Our to-doList back end supports the following features:

• The back end supports multiple users. Each of these users has his/her own list of tasks stored on the server.

• New users can only be added through a registration form, and users need to log in to start adding tasks.

• log in and a registration form for new users are added as a home page.

• Once a user has successfully registered, his/her log in information is permanently stored in a JSON file on the server to allow him/her to log in to their account in the future.

• Each user has access to only his/her own list of tasks after logging in. If a user logs out and another user logs in from the same browser, the displayed tasks belong to the currently logged user.

• Upon registering a new user, the user will be redirected to the log in page/dialog. Once the user is successfully logged in, this user's list of tasks is pulled from the server and displayed in the 2DoList homepage.

• Changes made to a user's list of tasks are permanent. Changes are permanent means that any change made by the user to the list of tasks are momentarily pushed to the list of tasks stored in JSON files on the server.

⇨ *Validations on the Registration Form:*

- User name cannot be empty and must only contain the characters [A-Z] – [a-z].
- User e-mail must be in proper e-mail format.
- No two users can have the same e-mail but they may have the same name.
- Password must be at least 5 characters and is CASE-SENSITIVE.

⇨ *Validations on the log-in form:*

The user should enter a valid email and password. The validity of the data is checked at the server side for all users registration information. This means that this email should be previously registered and the password entered during registration is the same as the one the user is using during logging in

Our To-do List main functions(client-side):

Our 2ToList web application is based on functions to handle and interact with a user.

Each task has its name (a must), description and date (optional), it also has it's unique ID and status which indicates whether it's in progress , completed or archived.

It's provided with a log out button for signing out.

Function	Description
Function redrawingTable (array){};	It re-creates the table each time according to the desired table contents.
function fillingArrays(id) {};	It adds tasks(elements) to the arrays whether it's "All tasks" ,"Completed" ,"In Progress","Archived"according to the user's actions.
Function list(id){};	This function is responsible for the actions found in the options dropdown button, it controls all the functions offered by the list : • Archive: when the user selects this option, the task is marked as archived and moved to the Archived list, the task will be removed from the current list. • Edit Task: when this option is selected, the same dialogue box of the Add Task appear. However, this time the information of the task appears in it, and the user is allowed to change them. • Delete Task: when this option is selected, the task is deleted and all its information is forgotten.

	A confirmation dialog box appears to the user with two options: (1) Delete and (2) Cancel. • Mark as Done: when this option is selected, the task is marked as completed and a strike through is run through the name of task.
Function getid(obj){};	• it's a starter function used to return the unique id of the element(object) selected on the web page.
Function sort(id){};	• It's responsible for sorting the table shown either by name or date. It's worth noticing that: • Sort by Name: if this option is selected, the list of tasks is sorted in an ascending order according to the initials of the task names. • Sort by Date: if this option is selected, the list of tasks is sorted according to the due date set to each task. Tasks with the nearest due date appear at the top. Tasks with no due date set by the user is listed with no order at the end of the list.
A Jquery date picker	Function imported locally to supply the add task form with a user-friendly calender.
function godelete() {};	This function is called once a checkbox is checked, it shows a delete button instead of the normal add task button and allows the user to delete checked tasks on the table.

Our To-do List main functions(server-side):

→ app.get('/', function (req, res) {

res.sendFile(__dirname + "/" + "Registeration.html");

})

- Redirects to the home page.

→ app.get('/To-doList.html', function (req, res) {

```
        if(req.session.email){
    res.sendFile( __dirname + "/" + "To-doList.html" );
        }
        else{
                res.sendFile( __dirname + "/" + "Registeration.html" );
        }
})
```

- Checks whether user is signed in to load the to do list page upon request. If not , it redirects to the home page.

→ app.post('/ajaxRequest', function(req, res){});

- Saves users' tasks.

→ app.post('/register', urlencodedParser, function (req, res) {});

- Handles and validates registeration requests.

➔ app.post('/signout', urlencodedParser, function (req, res) {});

- Handles log out requests.

→ app.post('/signin', urlencodedParser, function (req, res) {});

- Handles and validates sign in requests.

→ app.post('/create', function(req, res){});

-sends back to the client side the users' data upon request (eg reload ,sign in)

How the work is divided between the team members (who did what):

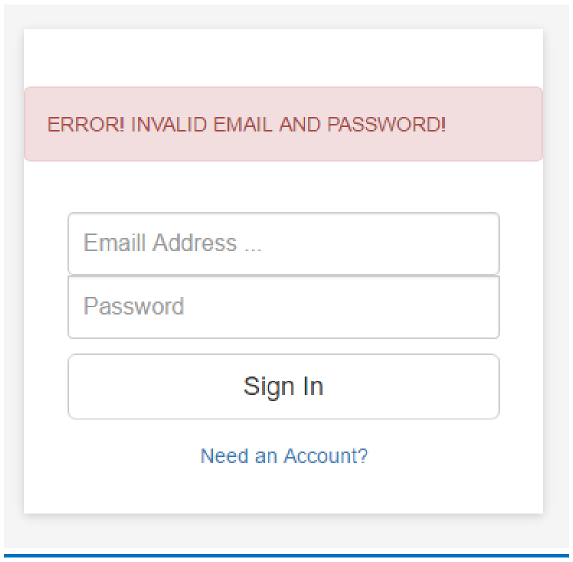
We both worked coherently on the server side. Whether session requests, ajax syntax requests or get and post requests.

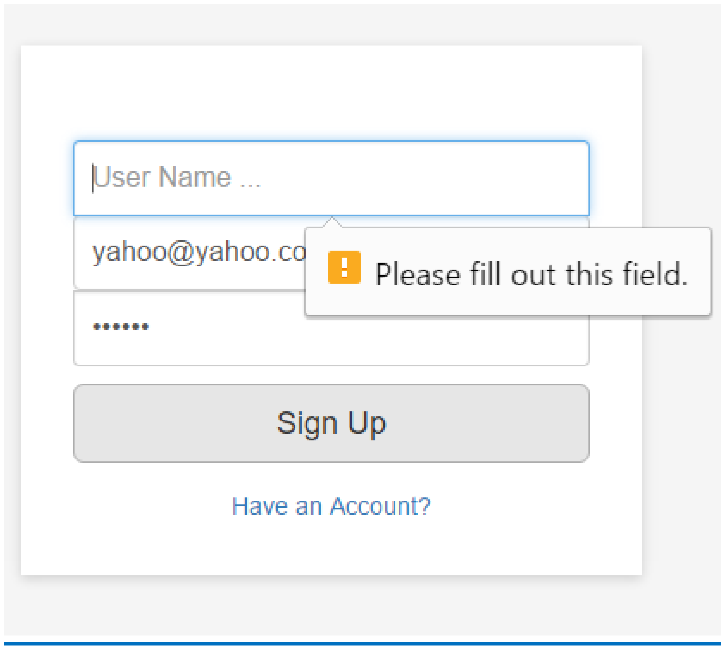
Bassent: Client side validations on sign in, error messages on forms.

Mira: Server side validations and reading and writing to Json file.

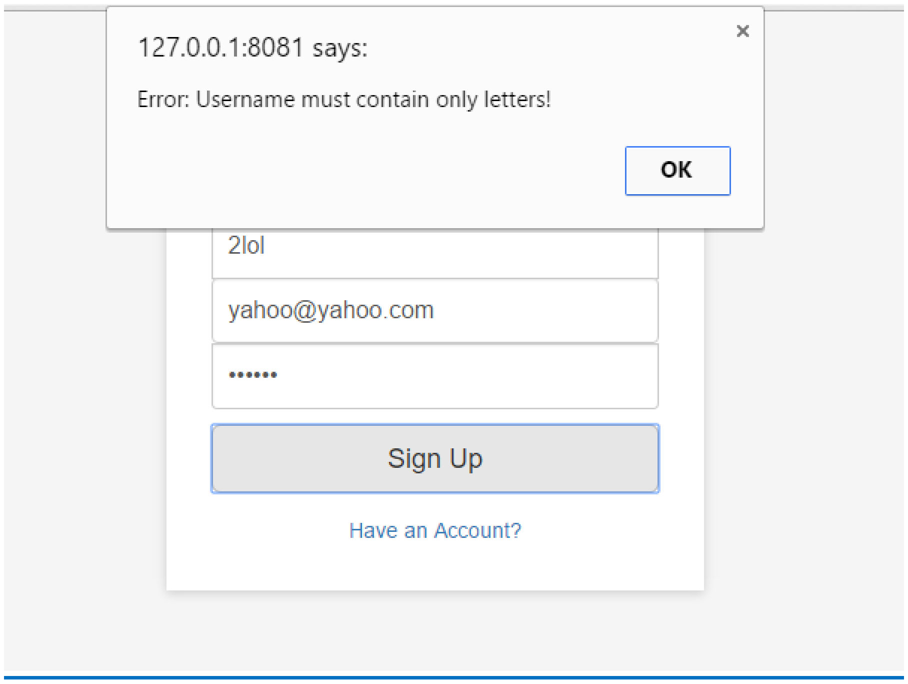
https://ide.c9.io/bassent/todolistp2#openfile-README.md

C9: http://todolistp2-bassent.c9users.io:8081/


ERROR! INVALID EMAIL AND PASSWORD!



Please fill out this field.

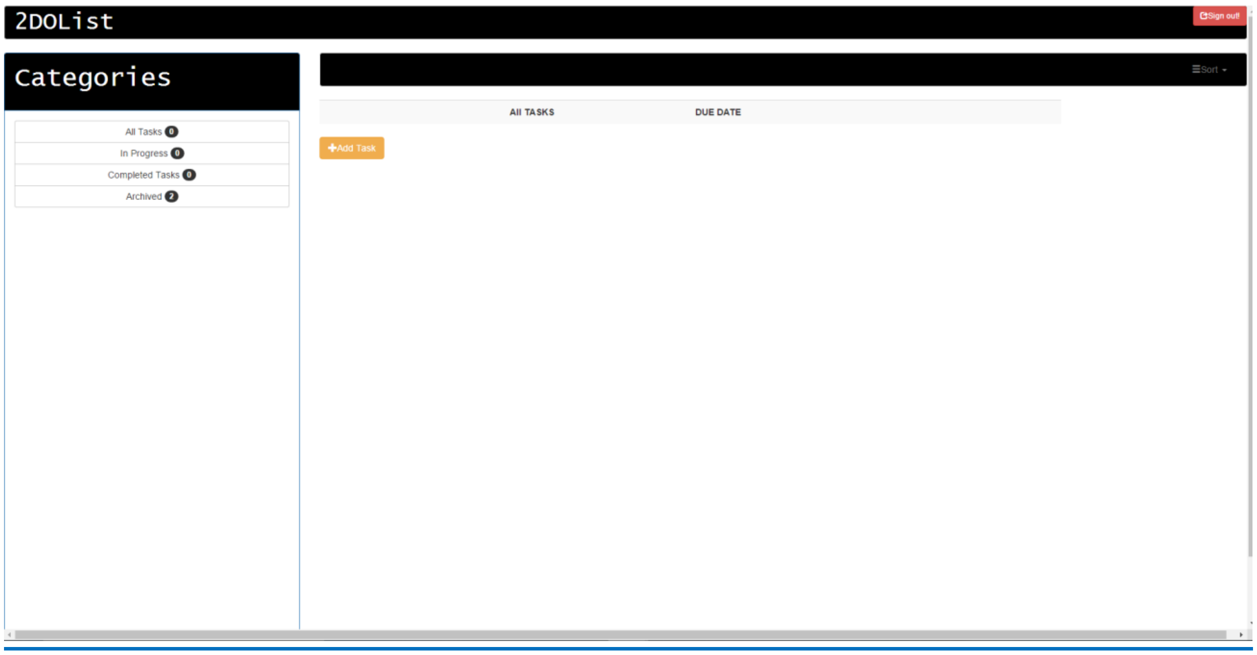

127.0.0.1:8081 says:

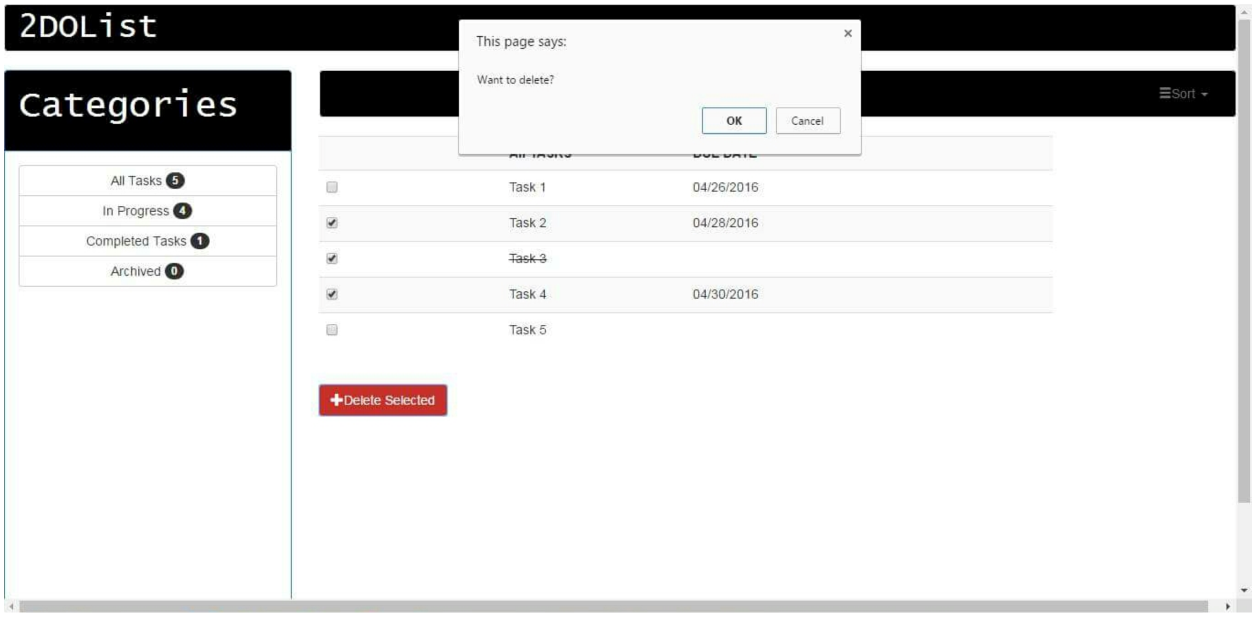
Error: Username must contain only letters!

2lol

yahoo@yahoo.com

••••••





	All TASKS	DUE DATE
☐	Task 1	04/26/2016
☑	Task 2	04/28/2016
☑	~~Task 3~~	
☑	Task 4	04/30/2016
☐	Task 5	