

# CSE 316 Spring 2024: Individual Web Programming Assignment 4

**Due: May 23, 2024 at 11:59 PM**

[Read This Right Away](#)

This assignment is due at **11:59 pm on May 23, 2024 KST**.

## Directions

- At the top of every file you submit, include the following information in a comment
  - Your first and last name
  - Your Stony Brook email address
- Your programs should be formatted in a way that is readable. In other words, indent appropriately, use informative names for variables, etc. If you are uncertain about what a readable style is, see the examples from class and textbook as a starting point for a reasonable coding style.

## Tasks

This assignment will expand the project, adding in a profile page that follows a similar design to the other pages. All of the data should be persisted like in the previous assignment.

For this project, I want to continue to see good version control practices. You will turn in a link to your GitHub repository (you can re-use the same repository from the previous assignment) and I want to see multiple commits that are separated by the different work tasks. Make sure your commits are appropriately labeled and descriptive.

## Requirements

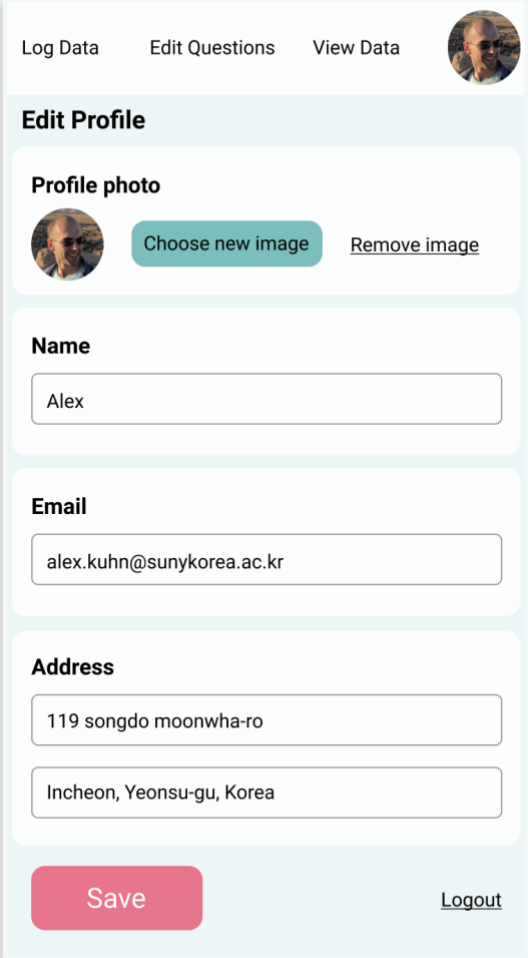
Implement the profile page as shown in the mockup in Figure 1. This page should have the same responsive behavior as your Edit Questions and Log Data page.

As part of this page, you will be creating a User object.

If you are using SQL, this would be a new 'user' table. However, since the app in this case is a single user app, there is no login per se and there should only be 1 user record in the table at any time.

The application will store the following profile information in the database:

- **Name**
- **Email address:** provide basic validation to ensure the email address has at least 1 character before an @ symbol, followed by a domain name before saving it.
- **Profile image:** your react client should upload the profile photo to your Cloudinary account and only store the URL to the image in your database. You will need to make a free account on Cloudinary to get this working.
- **Address:** the address should be stored as two separate strings, one for each text field. The address strings should be stored with the User table.



The mockup shows a mobile application interface for editing a user profile. At the top, there is a navigation bar with three links: 'Log Data', 'Edit Questions', and 'View Data'. To the right of these links is a circular profile picture of a man. Below the navigation bar is a section titled 'Edit Profile'. This section contains four form fields: 'Profile photo' (with a 'Choose new image' button and a 'Remove image' link), 'Name' (with the text 'Alex'), 'Email' (with the text 'alex.kuhn@sunykorea.ac.kr'), and 'Address' (with two lines of text: '119 songdo moonwha-ro' and 'Incheon, Yeonsu-gu, Korea'). At the bottom of the form is a large pink 'Save' button and a 'Logout' link.

Figure 1. Mockup of Edit Profile Page

Before there is a profile, (or after you have disconnected from a session) you will need to start the user on a type of identification page. There will not be a login screen as such since we won't require a password for this assignment. Simply a 'User' id panel. It can be a simple box where the user can enter their name. I do not provide a mock up here so use your imagination for the panel design.

If they are found in the user database (table or document), then display the profile page as shown. If not (**and after verifying there are no records in the user table**), ask the user to create their profile and give them a blank copy of the profile page.

The user moves between the different pages (edit questions, log data, view data) with the menu buttons at the top. The 'Logout' button on the profile page simply means disconnect from the database and display the User id panel to restart the session.

If they click on their image while using the app, the profile page comes up. They can edit and save the profile again to update data. And of course as mentioned above, they can disconnect with the Logout button.

## Notes on Images

Take note that you are now storing an image with the profile. It is not a typical practice to store images directly in a database due to their size. Mysql has a datatype called LONGBLOB that can store a large chunk of binary but this has size limits as well.

I strongly recommend you use Cloudinary to store a profile image. It will provide a URL that can be put into the database in the profile image field.

This means:

1. When a new image is selected, you would need a dialog to allow the user to pick a file from the system. Your code would have to upload that to Cloudinary and receive the URL. Finally, you would store that URL in the profile table.
2. When you are retrieving the profile, you would use Cloudinary to retrieve the image and then display it in the profile page.

## Thinking about Design

Here's a thought experiment that you do NOT have to turn in but I encourage all of you to think about. This app was designed as a single user app (per install). How much complexity would be added if we had real logins and the answers/questions were tied to a specific user.

Besides having to securely store and check passwords (and later use cookies or other session control mechanisms), we would have to tie the answers to a user as well as to questions. In SQL, this would involve a 3 way join between user, answer, and question tables.

How might the complexity add to the number and types of error conditions which may arise and may have to be handled?

## Interactivity

The following behaviors should be implemented:

- You can get to the profile page by clicking on the User's profile image in the navigation bar at the top of the webpage.
- On the Edit profile page:
- Clicking on "Choose new image" OR clicking on the profile photo image allows the user to select a new image. Perform basic image validation to ensure the newly selected image is a common image type before uploading the image to Cloudinary.
- Clicking on "Remove image" removes the image from the User's account and updates the page to show some default profile image (select whatever image you want as the default). You do not have to try to remove the image from Cloudinary.
- Clicking on "Save" should submit all the changes to the User's account to the backend server.
- Clicking on "Logout" closes the session and displays the User ID panel.

Note that once the user save's their changes on the edit profile page, it should persist that to the database and show that information on the next page refresh (i.e. making a request to the backend server to get the latest information).

Also, once the profile photo has been changed, that profile photo should show up on the navigation bar for any page the user visits (e.g. on "Edit Questions" page it shows the new photo). Make sure that the profile image will display properly in the same small size with rounded corners, even if the user uploads a large image.

**Important:** For this assignment, you will continue using github. You may create a new repository or continue using the repo from assignment 2 or 3. If you continue using your prior repo, please clearly indicate in your first assignment 4 checkin that 'ASIGNMENT 4 Starts Here' as part of the message field. The same guidelines apply as I gave you for assignments 2 and 3. If this is a new repo, then please share it with me as a collaborator (github name: anmione) so that I can assess your github checkins as part of your grade.

## On Testing

I will evaluate your assignment on the latest version of Chrome on OS X (Mac). I do not expect any browser compatibility issues since this is a relatively simple website, but please test your code on Chrome before submission.

## Submission

When the code is working properly, you should do the following before submitting:

1. Clean up any dead code or debug code (remove it)
2. Make sure to **remove the node\_modules folder**! This becomes very large over the course of development. I can recreate the folder easily with *npm install* using your package.json file. Leaving the node\_modules folder in the submission will cause a deduction on your grade.
3. After inviting me as a collaborator on your github repo, *put the link in the Notes of the code submission so I know which particular collaboration invite belongs to you.*
4. Write some meaningful comments to make the code easy to follow (mainly in your javascript/typescript but it doesn't hurt to make some notes in the HTML as well.)
5. Move your development folder to a top level folder called CSE316\_HW4\_<name>\_<idnumber> where <name> is replaced by your name and <idnumber> is replaced with your student id. If your name is Joseph Kim with a student number of 12345678, then your folder name for submission is CSE316\_HW4\_JosephKim\_12345678.
6. Compress your top level folder containing entire development tree (producing a zip file). Upload the zip file to the assignment page in Brightspace.
7. Navigate to the course Brightspace site. Click **Assignments** in the top navbar menu. Look under the category 'Assignments'. Click **Assignment4**.
  - a. Scroll down and under **Submit Assignment**, click the **Add a File** button.
  - b. Click **My Computer** (first item in list).
  - c. Find the zip file and drag it to the 'Upload' area of the presented dialog box.
  - d. Click the **Add** button in the dialog.
  - e. You may write comments in the comment box at the bottom.
  - f. Click **Submit**. **← Be sure to do this so I can retrieve the submission!**

## Grading

I will be testing your submissions on Google Chrome under Mac OSX.

Grading will be based on the styling and functionality of the submission. The points are assigned as follows:

- **Static design:** Layout matches the mockups and style is similar to the same mockups. Content is correctly displayed: **10 points**
- **Interactivity:** Checkbox, mouse click and form submission functionality all work correctly. Persistence to the backend database works correctly: **15 points**
- **Responsiveness:** Good style when resizing when resizing page as described above. **10 points**
- **Database:** The database follows the above given schema and guidelines. **5 points.**
- **Cloud use:** The image is correctly uploaded to cloundinary and appears on the profile page correctly. Profile image change functionality works. **5 points.**
- **Source Control:** Good source control practice. I significant number of commits with meaningful comments: **5 points.**

Besides the above grading scheme, the code submitted should be professional in quality. It should be well structured, consistently styled and be free of warnings. If the code contains warnings in the build or execution, it could result in a deduction of up to 5% of the grade.

**Additional reminder: remove your node\_modules folder before zipping and submitting code. Failing to do this will lose 5% of the grade!**