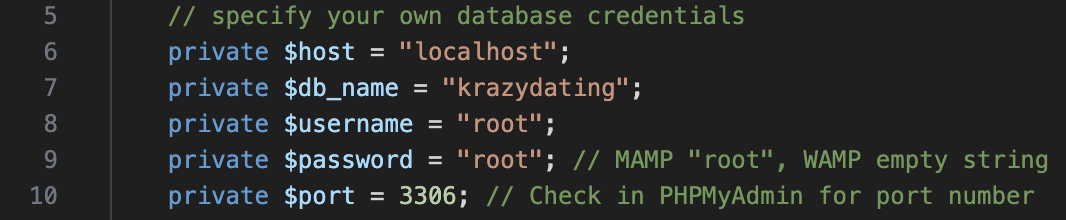
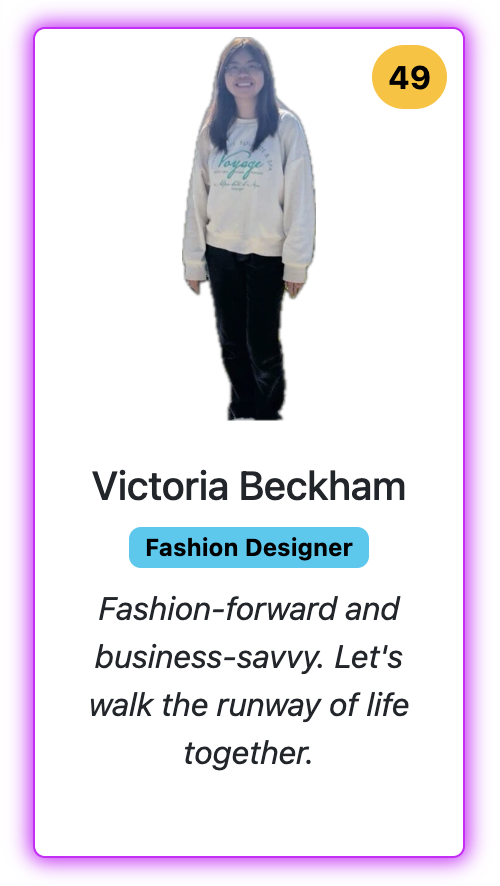
### **Challenge #2: JavaScript - Krazy Dating API (30 minutes + *optional* 10 mins)**

Create a web application that interacts with the **Krazy Dating API** to retrieve and display user profiles. You will build functionality to:

1. Fetch all profiles.
2. Fetch profiles by gender (men or women).
3. Dynamically display profiles using Bootstrap cards.

#### **Instructions**

1. **Locally hosted API application (“Krazy Dating API”)**
   * Go to **eLearn → Content → Week 5 → In Class → (unzip) Week5\_InClass**
   * In **VSCode**, open **krazydating → api → config → database.php**
     + Edit **Lines 8-10** as it fits your local environment.  
       
   * In **VSCode**, open **krazydating → api → db → load.sql**
     + Open **PHPMyAdmin** (e.g. <http://localhost/phpmyadmin/>) in web browser or use MySQL Workbench (or similar DB client app).
     + Run **all SQL statements** inside of **load.sql** such that:
       1. New schema (krazydating) is created.
       2. New table (person\_table) is created.
       3. Sample user profile data is inserted into the table.
   * Save all of the above.
   * Next, move **krazydating** folder directly under your **Webroot** such that this **API app** can be accessed at: <http://localhost/krazydating/> (if you’re using a different HTTP port such as 8888, you need to access it at <http://localhost:8888/krazydating>).
     + **[WAMP]** c:\wamp64\www\**krazydating**
     + **[MAMP]** /Applications/MAMP/htdocs/**krazydating**
   * ***Don’t WORRY about the PHP code – it’s NOT tested in IS216. Treat it as a black box for the purposes of front-end web app development.***
2. **Front-End Web Application’s Skeleton Files**:
   * Go to **eLearn → Content → Week 5 → In Class → (unzip) Week5\_InClass → Challenge2**
   * You will find **3 skeleton files** along with image files in **images** sub-folder:
     + home.html: The main HTML file.
     + styles.css: The CSS file used to style the main page.
     + script.js: The JavaScript file you’ll use to add interactivity to the main page.
3. Don’t forget to use **Chrome Browser’s InCognito mode**.
4. Please remember to **hard-reload/refresh** (the main HTML page in the web browser upon making changes to the HTML file (if any), the CSS file (if any), and the JavaScript file (you will make changes here for sure).
   * **Windows**: CTRL + SHIFT + R
   * **Mac**: command + SHIFT +R
5. **Task 1 - Fetch and Display All Profiles**
   * Add more JavaScript code to populate\_cards() function, which is called from inside get\_all() function.
   * Read the code inside of the get\_all() function to better understand the workflow.
     + The API response contains an array of profiles, which includes data like the profile picture, name, occupation, quote, gender, and age.
     + Use **Postman** to send an API request to the specified API endpoint and visually inspect what’s inside the **API response**.
   * In populate\_cards() function, console.log() the parameter passed to the function. What’s inside? What is its data type? How can you use it to dynamically create Bootstrap cards and display them on the page?
   * Also, note that the profile’s **gender** determines whether the card’s **border** is highlighted in **blue** (for males) or **pink** (for females).
     + Look in styles.css for the relevant **CSS code**.

  
 **One Male One Female**

1. **Task 2 - Display Profiles by Gender**
   * Inspect the HTML page to find 2 buttons each of which has **onclick** enabled:
     + Get all men
     + Get all women
   * Add more JavaScript code to get\_by\_gender(gender) function to filter profiles by gender.
   * Instead of rewriting the entire logic, think about how you can utilize the code from get\_all() function.
2. **Task 3 - Automatic Loading of All Profiles at the time of fresh page loading**
   * Modify the HTML code at Line 24 inside the <body> tag so that, when the webpage is loaded for the first time, all user profiles (both male and female) are automatically retrieved from the Krazy Dating API and displayed as Bootstrap cards.
3. **(Optional) Task 4 - Asynchronously Fetch and Display Weather Data for Each Profile**
   * In this task, you will modify the populate\_cards() function to asynchronously fetch the current weather information for each profile's city using the OpenWeatherMap API. For each profile:
     + Retrieve the profile's "city" from the API response.
     + Make an asynchronous API call to OpenWeatherMap to get the temperature for that city.
     + After fetching the temperature, update the profile's card to include the city name and temperature, along with the other profile details (name, age, occupation, etc.).
     + Ensure that the weather data is fetched and displayed sequentially for each profile.

*See below for example output screens*

| home.html **Tasks 1/2/3 are completed.**  **Task 4 is NOT completed.**  (upon loading this page for the first time - the user hasn’t clicked on any buttons) |
| --- |
| *Scroll down to see more profiles…* |

*See more examples in the next pages*

| home.html **Tasks 1/2/3 are completed.**  **Task 4 is NOT completed.** (after the user clicks on **Get all women** button - **only** **female profiles must be shown**) |
| --- |
| *Scroll down to see more profiles…* |

*See more examples in the next pages*

| home.html **Tasks 1/2/3 are completed.**  **Task 4 is NOT completed.** (after the user clicks on **Get all men** button - **only** **male profiles must be shown**) |
| --- |
| *Scroll down to see more profiles…* |

*See more examples in the next pages*

| home.html **Tasks 1/2/3/4 are completed** *(City and Temperature information is displayed for each profile)* (upon loading this page for the first time - the user hasn’t clicked on any buttons) |
| --- |
| *Scroll down to see more profiles…* |