README

Team Sun members: Aarushi ‘Ash’ Singh, Brian Dang, Chang Jin, Chinguun Erdenebileg, Guilin ‘Bill’ Zhang, Lucy Bodtman, and Segev Moritz.

---

1. Brief overview of the code

- **server.js:** This JavaScript file is the heart of the application; it connects all routes and controllers, and stores information necessary for redirections. Running this file boots the application.
- **database_manager.py:** This is the Database Management script we use to communicate with, set up and maintain our AWS database. We run the script from databaseConsole.ipynb for ease of use.
- **\controllers** is a folder that holds all controller.js files. Each file within this folder receives requests from their corresponding route.
- **\routes** is a folder that contains all routes.js files. The controllers and routers work with each other for a successful web application.
- **\public** is a folder that contains all front end-documents. This includes the .js files for any functions needed, the JSON file for testing the data, as well as .css files for styling the webpages.
- **\views** is where the HTML files are stored. They work with stylesheets and .js files from the \public folder mentioned above.

---

2. Instructions for running the application

Dependencies: These are listed in the file package-lock.json, starting at line 10.

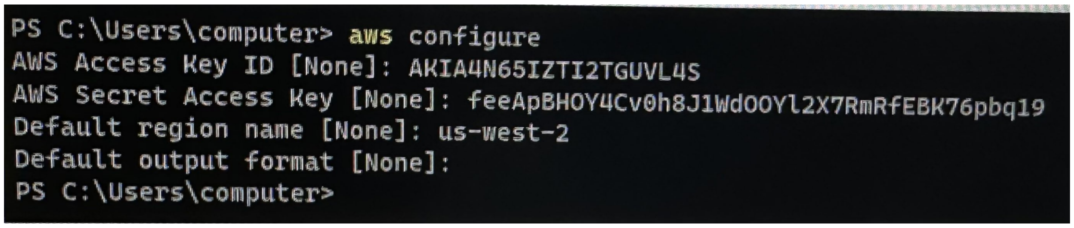
Running the application:

1. Download Node.js as well as NPM (Node Package Manager)
    a. Can download from (https://nodejs.org)
    b. Note: NPM is automatically installed with Node.js on Mac; select NPM during installation on Windows.
2. Clone the repository

    a. Open terminal
    b. Go to directory where you want to clone the repository
    c. Run “git clone https://github.com/abdulkhalil54/Team-Sun”
3. Install dependencies stated in package-lock.json
    a. Open terminal
    b. Run “npm install” to install all the required dependencies (packages).
4. AWS credentials:

```
[default]
aws_access_key_id = AKIA4N65IZTI2TGUVL4S
aws_secret_access_key = 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
```

    a. **On Mac:**
        i. Open the terminal
        ii. Navigate to home directory using the terminal
        iii. Create the “.aws” folder using “mkdir .aws” command
        iv. To see if the folder was successfully created, type ls -a, since “.” makes a folder hidden
        v. Create a file called “credentials” in “.aws” and paste the highlighted text above into the file
    b. **Windows:**
        i. Download AWS CLI: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
        ii. Open command prompt and use the command ‘aws configure’ as shown below, using the access keys highlighted above.



```
PS C:\Users\computer> aws configure
AWS Access Key ID [None]: AKIA4N65IZTI2TGUVL4S
AWS Secret Access Key [None]: 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
Default region name [None]: us-west-2
Default output format [None]:
PS C:\Users\computer>
```

5. Run the server
    a. Open terminal in VS Code. Ensure that you’re in the directory of the repository.
    b. Run the command “npm run dev”
    c. Access the application at “http://localhost:3000” in your web browser

Timeline: Through customer consultation, the following timeline was developed:

- **Deadline 1:** This is the deadline by which students must submit their applications.

- **Deadline 2:** This is the deadline by which professors must submit their preferences after viewing the applications.
- **Deadline 3:** This is the deadline by which the matching between applicants and sections is released.

Step 1. Navigating the Application as a Student:

1. Register / log in as a Student.
2. Submit a new application, including your preferences for the available sections.
3. To update your application before the deadline, you must submit a new application.
4. Wait for the matchings to be released.

Step 2. Navigating the Application as a Professor:

1. Register / log in as a Professor.
2. View applicants and their applications for your section(s). Provide preferences for the applicants in your sections.
3. Wait for the matchings to be released.

Step 3: Matchings

1. Navigate to http://localhost:3000/api/match in your browser to execute the matching. The matching will be displayed on the same webpage, where the ‘id’ corresponds to the section ID, and the ‘username’ corresponds to the username(s) of the student(s) who was / were matched into that section.
2. Log in as a student to see the updated status of your applications.

---

3. Test data

- Some of the data we used to test and debug the project can be found under public > javascripts > studentMockData.json.
- Mock accounts and applications are also present in the database. These can be viewed through running databaseConsole.ipynb.

---

4. Link to GitHub repository

Here is the link to the Team Sun repository: https://github.com/abdulkhalil54/Team-Sun

5. Attributions for sources of code/design from outside

- **Application Form:** We took inspiration for the visual appearance of our Application Form from the generic Google Forms structure.
- **Matching Algorithm:** Our matching algorithm took the Gale-Shapley algorithm and modified it to our needs. Unlike the Gale-Shapley algorithm, ours does not return a stable matching, and our preference lists are not the same size.

6. Known bugs/limitations

- There is no in-app back button, and the one on the browser returns unintended results.
- There is no functionality to amend the situation where a user has forgotten their password.
- The home button may return unintended results: this is the case in the Professor portal.
- We're currently using an honor system to determine whether a user is a student or a professor. A future update could include verification of the same.
- We currently expect students to rank every section, and for professors to rank every application, with no error-handling other than ensuring that the entered preferences are valid. A future update could include error-handling for the same, or perhaps a more flexible system in which students and professors can outright reject sections or applications.
- If there is an error on the site, the only way of checking what the error is is through the terminal.
- The matching algorithm is not fully integrated with the rest of the project. A fully-functioning app would execute the matching algorithm after the deadline for the Professor preferences has passed.
- Deadlines should be enforced so that students / professors cannot make further submissions after the respective deadlines have passed.
- Currently, all professor accounts are able to see all sections and applicants. We're using an honor system where professors only submit preferences for their section. A future update could include verification for the same.
- Running the matching algorithm before all students / professors have made their submissions returns unintended results.