**A4 API**

**GET http://127.0.0.1:3000/api/users**

* Get a list of users in JSON format

**GET http://127.0.0.1:3000/api/users/:user\_id**

* Get a particular user with the specified user\_id in JSON format
* The id is the registration email
* 404 if user not found

**GET http://127.0.0.1:3000/api/users/:user\_id/behaviors**

* Get all behaviors of a particular user with the specified user\_id in JSON format
* The id is the email used for registration
* 404 if user not found

**POST http://127.0.0.1:3000/api/users**

* Create a new user and return the new user in JSON format
* User email and password are transmitted through HTTP request body as key-value pairs in x-www-form-urlencoded format. The keys are *email* and *password*
* 409 if an user with the same email already exists

**POST http://127.0.0.1:3000/api/users/:user\_id/behaviors**

* Create a behavior for a particular user with user\_id
* Returns the user with the newly inserted behavior in JSON format
* The id is the email used for registration
* Behavior details are transmitted through HTTP request body as key-value pairs in x-www-form-urlencoded format. The keys are *date*, *ip*, *device\_height*, and *device\_width*
* 404 if user not found

**PUT http://127.0.0.1:3000/api/users/:user\_id**

* Update the user with user\_id
* Returns the updated user in JSON format
* The id is the email used for registration
* Update details are transmitted through HTTP request body as key-value pairs in x-www-form-urlencoded format. The keys are *description, password*, *display\_name*, *profile\_pic*, and *type*
* 404 if user not found

**DELETE http://127.0.0.1:3000/api/users/:user\_id**

* Delete a user with the specified user\_id
* The id is the email used for registration
* 404 if user not found