

File Handler Help:

The file handler is where you implement the “Web API Test Page” section of the specs. It uses the url given to find and return a particular file inside the web/ folder of the project files we give you. For example, if your server is sent a url like this:

```
http://192.168.23.12:8080/HTML/404.html
```

http:// has to do with protocols, 192.168.23.12 is the address of the computer you're running on (the exact number will almost certainly be different; this is just an example), 8080 is the port number, and HTML/404.html is a relative path telling you which file the client wants, so you will return the file from this location:

```
path/to/your/web/folder/web/HTML/404.html
```

Where path/to/your/web/folder is whatever gets you to the web folder we give you, web/ to get inside the web folder, and everything after that is the relative path.

If the path coming from the URL is empty, that means the client is asking for /index.html

When your file handler is working, you should be able to run your server on port 8080 (or whatever other port you choose), type

```
http://localhost:8080/index.html
```

into a web browser, and get a page that looks something like this:

Family Map app server x +

localhost:3000

This is the Family Map server used to power the Family Map Android app.

Built to power BYU CS 240 Family Map Application for Android

The web API that will be used is described below. Some of the APIs require a request body to be sent (namely "/user/login" and "/user/register"), while others require an Authorization token (received at login). To view the JSON format required for "/user/login" and "/user/register" simply click on the link below and look in the request body. To try out an API, click on its link below. It will fill the boxes below with the data to be sent with the request. Edit the data as needed and click Submit.

Database Commands

- [/clear/](#) This API will clear ALL data from the database, including users and all generated data. This API can be run from a browser by simply typing it in the address bar or by clicking this link followed by pressing the Submit button below. No authorization token is required.
- [/load/](#) This API will load the server's database with data provided by json text in the response body. The json text must contain an array of users as defined in the register details in addition to a personID, an array of persons, and an array of events. WARNING: all data from the database is wiped when this is called. The json file will be specified in the body of the request. A example.json file is provided to get you started with loading specific data. No authorization token is required.
- [/fill/{username}](#) This API will fill the server's database with fake data for the specified user name. The "username" parameter is required and must be a user already registered with the server. It can be any user name you choose. If there is any data in the database associated with the given user name, it is erased. This API can be run from a browser by simply typing it in the address bar (or by pressing the link, filling in the details, and clicking submit). The base person generated should be a person representing the user. No authorization token is required.
- [/fill/{username}/\(generations\)](#) This API will fill the server's database with fake data for the specified user name. The "username" parameter is required and must be a user already registered with the server. All the ancestor data associated with this user is erased. This API can be run from a browser by simply typing it in the address bar (or by pressing the link, filling in the details, and clicking submit). The base person generated should be a person representing the user. No authorization token is required.

User Commands

- [/user/login](#) Use this to log in a user. A request body must be supplied specifying the username and password. If login succeeds, an authorization token will be returned. Use this token on other API calls that require authorization. The returned JSON object contains "Authorization" (the authorization token) and "username" (the username that was just logged in). No authorization token is required.
- [/user/register](#) Use this to register a user. An authorization token is returned. Use it just as you would a token from login. Returns the same json object as log in. It should be noted that when you register a user the database will automatically be filled.(Meaning you do not need to call the /fill API noted above). No authorization token is required.
- [/events/](#) This API will return ALL events for ALL family members of the current user. The current user is determined from the provided authorization token (which is required for this call). The returned JSON object contains "data" which is an array of event objects. Authorization token is required.
- [/event/{eventId}](#) This API will return the single event with the specified ID. The event must belong to a relative of the user associated with the authorization token. The returned JSON contains the requested event object. Authorization token is required.
- [/person/](#) This API will return ALL people associated with the current user. The current user is determined from the provided a authorization token (which is required for this call). The returned JSON object contains "data" which is an array of person objects. Authorization token is required.
- [/person/{personID}](#) This API will return the single person with the specified ID. The person must be related to the user associated with the authorization token. The returned JSON contains the requested person object. Authorization token is required.

A few notes:

- The authorization token is returned from the server in the "Authorization" attribute of the JSON object returned by the "/user/register" and "/user/login" APIs. The authorization token must be placed in the "Authorization" header on all subsequent HTTP requests.
- If something fails, the returned JSON object contains a "message" attribute which contains a message describing what happened. Watch for these as they will give helpful insight into why the server did not work as expected.

Try it out here

Handle:

Authorization token:

Request Body:

Response From the server:

There should be an icon at the top (a tiny picture of Dr. Rodham); the links in Database Commands and User Commands will be blue; and the Request and Response boxes at the bottom will be a nice size instead of tiny and cramped.