Software Product Sprint

Team 32 | Datastore

Table of Contents

Datastore-API.js

Import this module to facilitate the fetch calls. All are async functions.

```
login(String username, String unhashedPassword){
```

Returns a json, either json.error or json.success.

Json.success contains the user matching the username and password credentials.

```
}
```

```
register(String username, String unhashedPassword, String email,
String firstName, String middleName, String lastName){
```

Returns a json, either json.error or json.success.

Registers a new user if the username and email are not already in the database.

}

getUser(String username, String hashedPassword){

If the username and hashedPassword credentials are valid, returns a json matching the user defined in the database.

If not, alerts: alert("Something Went Wrong: " + response.error);

}

deleteCurrentUser(String username, String hashedPassword){

Returns a json, either json.error or json.success.

Deletes the user matching the username and hashedPassword credentials.

}

addDayToCalendar(String username, String hashedPassword, String date ("yyyy-mm-dd"), Int mood){

Returns a json, either json.error or json.success.

Adds the given day to the calendar of the user matching the username and hashedPassword credentials.

}

addEntryToJournal(String username, String hashedPassword, String message, String date ("yyyy-mm-dd")){

Returns a json, either json.error or json.success.

Adds the given entry to the journal of the user matching the username and hashedPassword credentials.

}

addOneToPanicButton(String username, String hashedPassword){

Returns a json, either json.error or json.success.

Adds 1 to the panic button counter of the user matching the username and hashedPassword credentials.

```
}

updateUser(String username, String hashedPassword, String prevEmail,
User: {

      username: string,

      passwordHash: string,

      calendar: array[day],

      firstName: string,

      middleName: string,

      lastName: string,

      panicButtonPressed: int,

      journal: array[string]

      } user){
```

Returns a json, either json.error or json.success.

Updates the user matching the username and password, the new user data is the data from the user parameter.

```
}

getUserCalendar(String username, String hashedPassword){
```

Returns the calendar(array[day]) of the user matching the username and password.

```
}

getUserJournal(String username, String hashedPassword){
```

Returns the journal(array[string]) of the user matching the username and password.

```
}

getUserPanicButton(String username, String hashedPassword){
```

Returns the panic button counter(int) of the user matching the username and password.

}

queryJournalEntries(String username, String hashedPassword, String startingDate (“yyyy-mm-dd”), String endingDate (“yyyy-mm-dd”)){

Returns the journal entries between a given range for the user matching the credentials.
}

Java Servlets

/api/datastore-debug: GET method, returns a table with all the users.
/api/delete-all: GET method, deletes all the users.
/api/delete-user: POST method, deletes the user matching the username and password.

```
payload = {
      username: string,
      passwordHash: string
}
```

/api/login: POST method, returns the user matching the username and password. The password must be not yet hashed.

```
payload = {
      username: string,
      passwordHash: string
}
```

/api/get-user: POST method, returns the user matching the username and password. The password must be already hashed.

```
payload = {
      username: string,
      passwordHash: string
}
```

/api/register: POST method, registers a new username if credentials are valid.

```
payload = {
      username: string,
      passwordHash: string,
```

```
      calendar: array[day],
      firstName: string,
      middleName: string,
      lastName: string,
      panicButtonPressed: int,
      journal: array[string]
}
```

/api/update-user: POST method, replaces the user matching the previous data with the data from the new user json.

```
payload = {
      user: {
            username: string,
            passwordHash: string,
            calendar: array[day],
            firstName: string,
            middleName: string,
            lastName: string,
            panicButtonPressed: int,
            journal: array[string]
            },
      prevUsername: string,
      prevPasswordHash: string,
      prevEmail: string
}
```

/api/query-journal: POST method, returns the journal entries between a given range for the user matching the credentials.

```
payload = {
    username: string,
    passwordHash: string,
    startingTimestamp: long,
    endingTimestamp: long
}
```