HW15: Final Project database

Anu Srikanth

Now URL: <https://final-proj-mlolscbxye.now.sh>

**Introduction**

The first two of my A-level Use cases have to do with the User or Account model. User and account refer to the same model. They are used to associate each of the users with their specific user account details, tickets they have purchased, ticket information, etc. This is the reason I have chosen to complete this model first.

|  |  |  |
| --- | --- | --- |
| **A-Level Use Cases** | | |
| **Use Case Name** | **Actor(s)** | **Description** |
| Create Account | User | The user can create an account with credentials |
| Login | User | The user should be able to login with previously created credentials |

Some of the modules/packages that I have added to my project are as follows. These were not included in the initial project proposal.

|  |  |  |
| --- | --- | --- |
| **Modules and Packages** | | |
| **Package/Module Name** | **Component** | **Description** |
| mongoose | Database | This allows for easy schema creation for models that give them a more definite structure and makes it easier for testing and validations. |
| Mocha | Testing | JavaScript framework to make asynchronous testing easier |
| Chai | Testing | Assertions library |

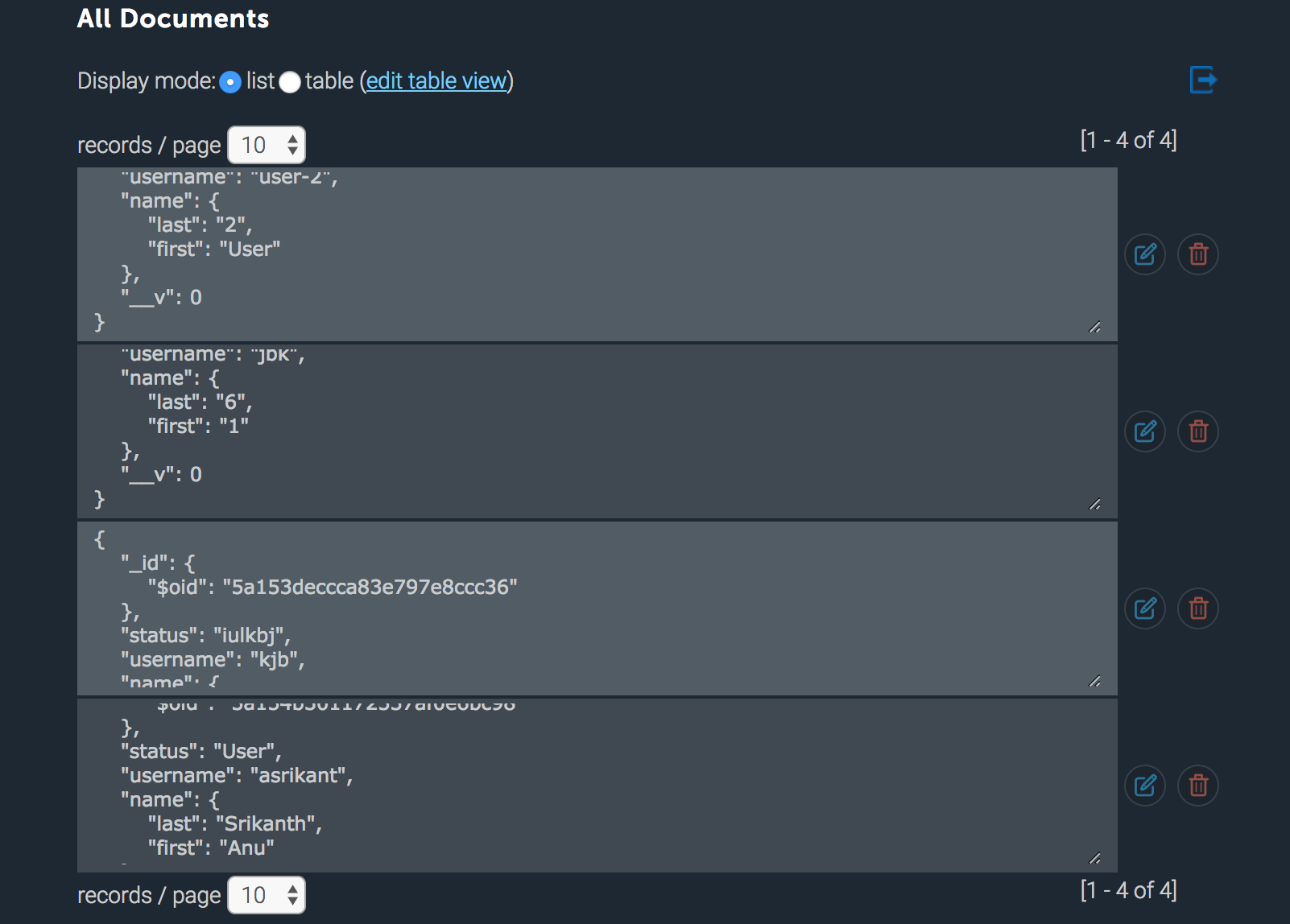
**Model functionality**

The following code sample is from my user model, the model that Ive chosen to store in the database. Here I am connecting to the mongodb database on mlab with credentials that I have previously provided. Then I set up my models using schema functionality provided by mongoose that helps keep models in a particular structure.





Documents updated/added in mlab’s database showing a successful connection



**Model Testing**

Below is some tests that I have written using the Mocha testing framework and a local database in order to not interfere with the data on the online one and also to speed up testing. They have passed.

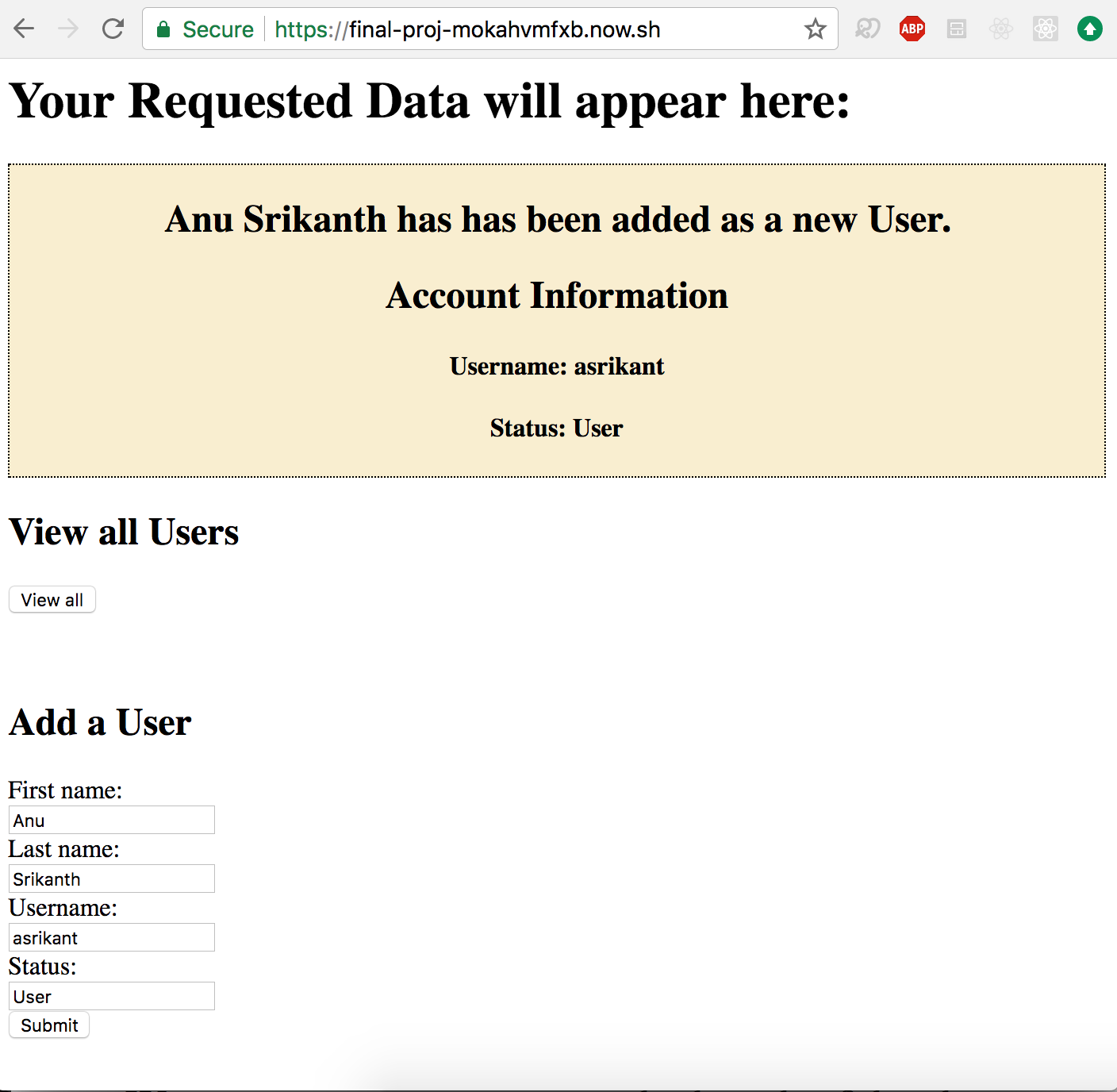
****

**CRUD functionality**

The following is how the process of executing the CRUD functions work on the backend. This will be followed up by working examples.

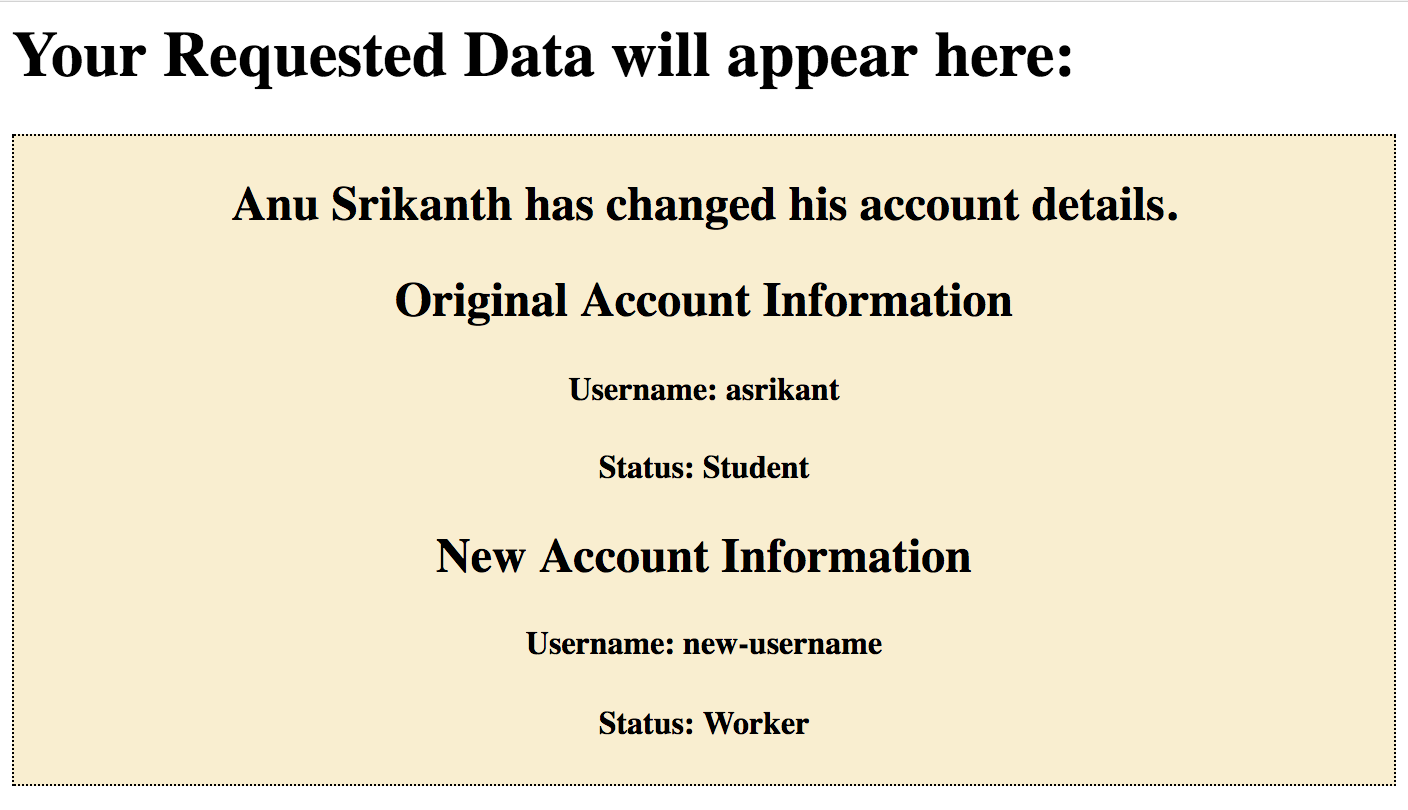
* 1. Form on public index page triggers an ajax request
  2. Ajax request sent to routes along with the associated method
  3. Routes file enumerates the different urls and paths that could be passed to it and does the relevant function when the ajax request url matches a route.
  4. This function updates/retrieved information in the database and follows up with a success or an error response
     1. If there is an error response, an error handler is called that logs the error and responds back to the client
     2. If there is a success response, the relevant view is populated with information and then displayed on the browser.

1. CREATE:



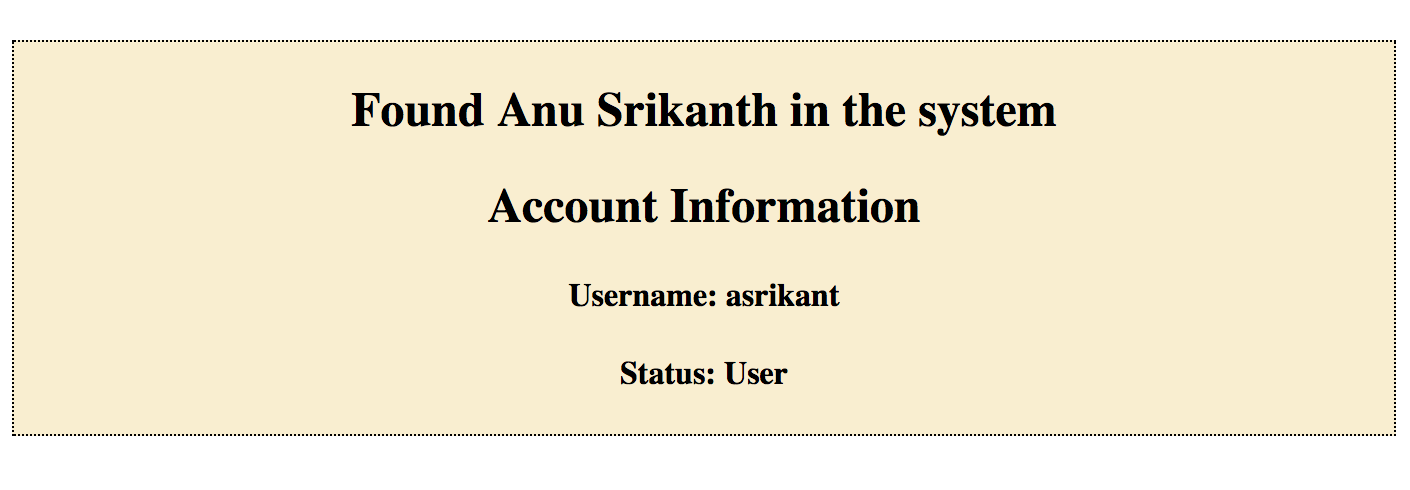


1. UPDATE





1. RETRIEVE





1. DELETE

