News Aggregator Single Page Application using MEAN Stack

Programmer's Manual

Folders/Files/Funtions

MongoDB Database

Articles Collection

```
{
  "_id": <ObjectId>,
  "title": <string>,
  "URL": <string>,
  "votes": <int>,
   “user”: <string>
  "comments": [<comment> [, ...]]
}
```

comments will be an embedded document

```
{
  "_id": <ObjectId>,
  "title": <string>,
  "URL": <string>,
  "votes": <int>,
   “user”: <string>
  "comments": [<comment> [, ...]]
}
```

Users Collection

```
{
    "id" : <ObjectId>,
    "name" : <string>,
    "email" : <string>,
    "username" : <string>,
    "password" : <string>
}
```

Folder: NewsAggregator (Generated from express-generator)

Folder: models

user.js

Connecting the User Schema through mongoose

CreateUser(newUser)

Takes in a user object and add it into the database

getUserByUsername(username)

Return a user object if the username parameter match in the database

getUserById(id)

Return a user object if the id parameter match in the database

comparePassword(candidatePassword, hash)

Compare user password in the database with the input password

Folder: public

Folder: javascript

newsaggregator.js

Routing, factory settings and controller functions are defined here

ngResource provides interaction support with RESTful services via the $resource service.

ngRoute provides routing and deeplinking services and directives for AngularJS apps.

ngStorage for local storing needs. User session is stored here.

Folder: partials

All the html pages are located in here

Folder: stylesheets

The .css file are located in here

Folder: routes

All the route modules are stored here. The endpoints for the application that deals with the article database are defined with router objects.

users.js

/users with get method:

A get request to this endpoint returns a matching user based on username and password in JSON format, allowing user to log in to the home page

/users/register with post method:

A post request to this endpoint adds a new user to the users collection

/users/logout with get method:

A get request to this endpoint will allow user to logout

articles.js

/api/articles with get method:

A get request to this endpoint returns list of articles in the JSON format

/api/articles with post method:

A post request to this endpoint adds a new article to the articles collection with the empty comments field.

/api/articles/:articleid with get method:

A get request to this endpoint returns the corresponding article from the database in JSON format.

/api/articles/:articleid with post method:

A post request to this endpoint adds a new comment to the corresponding article in the database

/api/articles/:articleid with delete method:

A delete request to this endpoint deletes an article from the database.

/api/articles/:articleid/:commentid with get method:

A get request to this endpoint returns the corresponding comment in an article from the database in JSON format.

/api/articles/:articleid/:commentid with delete method:

A delete request to this endpoint deletes a comment on an article from the database.

/api/articles/:articleid with put method:

A put request to this endpoint should retrieve the up/down-vote state of the article and updates the vote total in the database accordingly. It can also edit article titles.

/api/articles/:articleid/:commentid with put method:

Retreives the up/down-vote state of the comment from the user and updates the vote total of the comment accordingly. It can also edit comment contents.

Folder: view

The template engine, Jade, for the app is stored here.

File: app.js

The main body of the app, the server was built in this file.

File: package.json

Dependencies for the application are stored here