Theodore Kim
JinZhao Su
CS-UY 3083: Introduction to Databases
Project Part 3

Implemented Features:
1. Login
2. Public Posts

Planned Extra Features:
1. Adding meta data to content items (categorizing types of content items (i.e. photos, videos, links, etc.)) and categories
2. Commenting on Content Items
3. Defriending

SOURCE CODE

NOTE: 3rd Party Package's Source Code not shown:
>> express
>> jquery
>> mysql (mysql interface for nodejs)
>> jsonwebtoken
...etc...

NOTE: Pug (.pug files) is an HTML Preprocessor Scripting Language (to make HTML coding quicker)

FILE: ./index.js

```
1     // Import Express Server Framework
2     var express = require('express');
3
4     // Import query abstraction
5     const sql = require('./db');
6
7     // Import other utilizity modules
8     var path = require('path');
9     var debug = require('debug')('pricosha');
10    var http = require('http');
11    var cookieParser = require('cookie-parser');
12    var bodyParser = require('body-parser');
13
14    // Import API Routing
15    var site = require('./routes');
16    var api = require('./routes/api');
17
18    // Initialize server object
19    var app = express();
20    app.use(cookieParser());
21    app.use(bodyParser.json());
22    app.use(bodyParser.urlencoded({ extended: false }));
23
24
25    // Specify the utilized directories: views and public resources (i.e. JS and
      CSS)
26    app.set('views', path.join(__dirname, 'views'));
27    app.set('view engine', 'pug');
28    app.use(express.static(path.join(__dirname, 'public')));
29
30    // app routing
31    app.use('/', site);
32    app.use('/api', api);
33
34
35    // CODE TAKEN FROM DEFAULT EXPRESS APP BOILERPLATE:
36    // https://github.com/express/example
37    // catch 404 and forward to error handler
38    app.use(function(req, res, next) {
39      var err = new Error('Not Found');
40      err.status = 404;
41      next(err);
42    });
43
44    // error handler
45    app.use(function(err, req, res, next) {
46      // set locals, only providing error in development
```

```
47      res.locals.message = err.message;
48      res.locals.error = req.app.get('env') === 'development' ? err : {};
49
50      // render the error page
51      res.status(err.status || 500);
52      res.render('error');
53    });
54
55    var port = normalizePort(process.env.PORT || '3000');
56    app.set('port', port);
57
58    /**
59     * Create HTTP server.
60     */
61
62    var server = http.createServer(app);
63
64    server.listen(port);
65    server.on('error', onError);
66    server.on('listening', onListening);
67
68    /**
69     * Normalize a port into a number, string, or false.
70     */
71
72    function normalizePort(val) {
73      var port = parseInt(val, 10);
74
75      if (isNaN(port)) {
76        // named pipe
77        return val;
78      }
79
80      if (port >= 0) {
81        // port number
82        return port;
83      }
84
85      return false;
86    }
87
88    /**
89     * Event listener for HTTP server "error" event.
90     */
91
92    function onError(error) {
93      if (error.syscall !== 'listen') {
94        throw error;
95      }
96
97      var bind = typeof port === 'string'
98        ? 'Pipe ' + port
99        : 'Port ' + port;
100
101     // handle specific listen errors with friendly messages
102     switch (error.code) {
103       case 'EACCES':
104         console.error(bind + ' requires elevated privileges');
105         process.exit(1);
106         break;
107       case 'EADDRINUSE':
108         console.error(bind + ' is already in use');
```

```
109         process.exit(1);
110         break;
111       default:
112         throw error;
113     }
114   }
115
116   /**
117    * Event listener for HTTP server "listening" event.
118    */
119
120   function onListening() {
121     var addr = server.address();
122     var bind = typeof addr === 'string'
123       ? 'pipe ' + addr
124       : 'port ' + addr.port;
125     debug('Listening on ' + bind);
126   }
```

FILE: ./db.js

```
1     // Import mysql interface
2     const mysql = require("mysql");
3
4     // Create connection to MAMP server
5     const connection = mysql.createConnection({
6     host : "localhost",
7     port : "8889",
8     user: "root",
9     password : "root",
10    database : "pricosha"
11    });
12
13    // Define easy abstraction for making sql queries
14    const query = function (query, values) {
15    return new Promise((resolve, reject) => {
16    if (!values && query.includes("?"))
17          reject("Your query includes placeholders, but you don't provide binding
      values");
18    connection.query({
19          sql: query,
20          values: values
21    }, function (error, results, fields) {
22          if (error) reject(error);
23          else {
24                resolve(results, fields);
25          }
26    })
27    })
28    }
29
30    // export module
31    module.exports = query;
```

FILE: ./views/layout.pug

```
1     doctype html
2     html
3       head
4         title PriCoSha
5         link(rel='stylesheet', href='/css/style.css')
```

```
6         script(src="/js/jquery.min.js")
7         script(src="/js/jquery.cookie.js")
8       body
9         block content
```

FILE: ./views/error.pug

```
1     extends layout
2
3     block content
4       h1= message
5       h2= error.status
6       pre #{error.stack}
```

FILE: ./views/main.pug

```
1     extends layout
2
3     block content
4     h1 PriCoSha
5     h2 Pivate Content Sharing
6
7     - if (loggedin)
8          h3 Hello #{user.fname}!
9          h4 Public Posts
10      button#new-group Create a New Friend Group
11      button#show-post Post New Content
12      button#logout Logout
13       14    - else
14      h3 Recent Public Posts
15      div#public-posts
16      h3 Please Login to the System
17       18       div
18       input#email(type="email")
19      div
20       input#pass(type="password")
21      div
22       button#login-btn Login
23
24       25    br
25
26       27    hr
27       28    #footer
28      div PriCoSha
29      div An academic project by Theodore Kim and JinZhao Su
30      div Source: https://github.com/theo-kim/databases/project/pricosha
31      div Version 0.0.1
32
33       34    script(src="scripts/main.js")
```

FILE: ./routes/index.js

```
1     var express = require('express');
2     var router = express.Router();
3     var jwt = require('jsonwebtoken');
4     var jwtsecret = require('../server_config.json').serversecret;
5
6     router.get('/', (req, res, next) => {
7     var usertoken, user;
8     if (req.cookies.usertoken != null)
```

```
9     usertoken = req.cookies.usertoken;
10    else usertoken = null;
11
12    if (usertoken !== null) {
13    try {
14          user = jwt.verify(usertoken, jwtsecret);
15    }
16    catch (e) {
17          usertoken = null;
18          user = null;
19    }
20    }
21
22    var state = {
23    loggedin: usertoken !== null,
24    user: user
25    };
26    res.render('main', state);
27    });
28
29    module.exports = router;
```

FILE: ./routes/api/index.js

```
1     var express = require('express');
2     var router = express.Router();
3     var crypto = require('crypto');
4     var jwt = require('jsonwebtoken');
5
6     var salt = require('../../server_config.json').passhashsalt;
7     var jwtsecret = require('../../server_config.json').serversecret;
8     var query = require('../../db.js');
9
10    // Log user in
11    router.post('/login', (req, res, next) => {
12    var email = req.body.email;
13
14    // Hash password
15    var password = crypto.createHash('sha256')
16                         .update(req.body.password + salt)
17                         .digest('hex');
18
19    // Perform query
20    query("SELECT email, password, fname, lname FROM Person WHERE email=? AND
      password=?", [email, password])
21    .then((result) => {
22          if (result.length == 0) res.send('fail');
23          else {
24                // Return a session token with user information
25                res.send(jwt.sign({ fname: result[0].fname, lname: result[0].lname,
      email: result[0].email },
26                                  jwtsecret,
27                                  { expiresIn: '7d' }));
28
29          }
30    });
31    });
32
33    // See content items
34    router.get('/item', (req, res, next) => {
35    var token;
```

```
36
37    if (req.params.token == null) {
38    token = null;
39    }
40    else token = req.params.token;
41
42    // If not token, user is not authenticated, only so public posts
43    if (token === null) {
44    query("SELECT * FROM ContentItem WHERE is_pub=1 AND post_time >
      DATE_SUB(CURDATE(), INTERVAL 1 DAY)")
45          .then((result) => {
46                res.json(result);
47          });
48    }
49    // Else ONLY show posts that the user is allowed to see
50    else {
51    // Implement this later
52    }
53    });
54
55
56    // Create content items
57    router.post('/item', (req, res, next) => {
58    // Implement this later
59    });
60
61    // Create a FriendGroup
62    router.post('/group', (req, res, next) => {
63    // Implement this later
64    })
65
66    module.exports = router;
```

FILE: ./public/scripts/main.js

```
1     var mode = 0;
2
3     // Check mode
4     if($('#login-btn').length) {
5     mode = 0;
6     }
7     else {
8     mode = 1;
9     }
10
11
12    if (mode == 0) {
13    // View all items the user can see, in this case, only public posts b/c
14    // the user is not authenticated
15    $.get("/api/item", {}, function(items) {
16    for (var i = 0; i < items.length; ++i) {
17          $("#public-posts").append($(`<div class='contentitem'>
18                      <span class='title'>${items[i]["item_name"]}</span><br>
19                      <span class='item-id'>Item ID:
      ${items[i]["item_id"]}</span><br>
20                      <span class='post-time'>Posted:
      ${items[i]["post_time"]}</span><br>
21                      <a
      href='${items[i]["file_path"]}'>${items[i]["file_path"]}</a><br>
22                      <span class='post-email'>Posted By:
      ${items[i]["email_post"]}</span><br>
```

```
23                </div>`));
24    }
25    });
26
27    // Log user in
28    $("#login-btn").click(function() {
29    $.post("/api/login", {email: $("#email").val(), password: $("#pass").val()},
30          function(res) {
31                if (res === "fail") alert("Incorrect User Email and/or Password");
32                else {
33                      $.cookie("usertoken", res);
34                      location.reload();
35                }
36          })
37    });
38    }
39
40    else {
41    var token = $.cookie("usertoken");
42    // View all posts the user can see
43    $.get("/api/item", { token: token }, function(items) {
44    // Implement this later
45    });
46
47    // Create a new FriendGroup
48    $("#new-group").click(function() {
49    // Implement this later
50    });
51
52    // Show menu to create a new post
53    $("#show-post").click(function() {
54    // Implement this later
55    });
56
57    // Log user out of PriCoSha
58    $("#logout").click(function() {
59    $.cookie("usertoken", 0);
60    location.reload();
61    });
62    }
```

FILE: ./public/css/style.js

```
1     body {
-       font-family: sans-serif;
2     padding: 0 !important;
3     margin: 0 !important;
4     }
5
6     h1, h2, h3, h4 {
7     padding: 5px;
8     }
9
10    button {
11    font-size :18px;
12    background: #ddd;
13    border: 0;
14    padding: 5px 10px;
15    margin: 20px;
16    }
17
```

```
#footer {
background: #ddd;
padding: 10px;
font-size: 12px;
margin: 20px;
}

.contentitem {
padding: 10px;
display: inline-block;
background: #eee;
margin: 10px;
font-size: 18px;
}

.title {
font-weight: bold;
font-size: 24px;
padding-bottom: 5px;
}

.item-id {
color: #888;
font-size: 14px;
}

.post-time {
color: #333;
font-size: 16px;
padding-bottom: 10px;
}
```