**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. *// Import query abstraction*
4. **const** sql = require('./db');
5. *// Import other utilizity modules*
6. **var** path = require('path');
7. **var** debug = require('debug')('pricosha');
8. **var** http = require('http');
9. **var** cookieParser = require('cookie-parser');
10. **var** bodyParser = require('body-parser');
11. *// Import API Routing*
12. **var** site = require('./routes');
13. **var** api = require('./routes/api');
14. *// Initialize server object*
15. **var** app = express();
16. app.use(cookieParser());
17. app.use(bodyParser.json());
18. app.use(bodyParser.urlencoded({ extended: **false** }));
19. *// Specify the utilized directories: views and public resources (i.e. JS and CSS)*
20. app.set('views', path.join(\_\_dirname, 'views'));
21. app.set('view engine', 'pug');
22. app.use(express.**static**(path.join(\_\_dirname, 'public')));
23. *// app routing*
24. app.use('/', site);
25. app.use('/api', api);
26. *// CODE TAKEN FROM DEFAULT EXPRESS APP BOILERPLATE:*
27. *// https://github.com/express/example*
28. *// catch 404 and forward to error handler*
29. app.use(**function**(req, res, next) {
30. **var** err = **new** Error('Not Found');
31. err.status = 404;
32. next(err);
33. });
34. *// error handler*
35. app.use(**function**(err, req, res, next) {
36. *// set locals, only providing error in development*
37. res.locals.message = err.message;
38. res.locals.error = req.app.get('env') === 'development' ? err : {};
39. *// render the error page*
40. res.status(err.status || 500);
41. res.render('error');
42. });
43. **var** port = normalizePort(process.env.PORT || '3000');
44. app.set('port', port);
45. */\*\**
46. *\* Create HTTP server.*
47. *\*/*
48. **var** server = http.createServer(app);
49. server.listen(port);
50. server.on('error', onError);
51. server.on('listening', onListening);
52. */\*\**
53. *\* Normalize a port into a number, string, or false.*
54. *\*/*
55. **function** normalizePort(val) {
56. **var** port = parseInt(val, 10);
57. **if** (isNaN(port)) {
58. *// named pipe*
59. **return** val;
60. }
61. **if** (port >= 0) {
62. *// port number*
63. **return** port;
64. }
65. **return** **false**;
66. }
67. */\*\**
68. *\* Event listener for HTTP server "error" event.*
69. *\*/*
70. **function** onError(error) {
71. **if** (error.syscall !== 'listen') {
72. **throw** error;
73. }
74. **var** bind = **typeof** port === 'string'
75. ? 'Pipe ' + port
76. : 'Port ' + port;
77. *// handle specific listen errors with friendly messages*
78. **switch** (error.code) {
79. **case** 'EACCES':
80. console.error(bind + ' requires elevated privileges');
81. process.exit(1);
82. **break**;
83. **case** 'EADDRINUSE':
84. console.error(bind + ' is already in use');
85. process.exit(1);
86. **break**;
87. **default**:
88. **throw** error;
89. }
90. }
91. */\*\**
92. *\* Event listener for HTTP server "listening" event.*
93. *\*/*
94. **function** onListening() {
95. **var** addr = server.address();
96. **var** bind = **typeof** addr === 'string'
97. ? 'pipe ' + addr
98. : 'port ' + addr.port;
99. debug('Listening on ' + bind);
100. }

**FILE: ./db.js**

1. *// Import mysql interface*
2. **const** mysql = require("mysql");
3. *// Create connection to MAMP server*
4. **const** connection = mysql.createConnection({
5. host : "localhost",
6. port : "8889",
7. user: "root",
8. password : "root",
9. database : "pricosha"
10. });
11. *// Define easy abstraction for making sql queries*
12. **const** query = **function** (query, values) {
13. **return** **new** Promise((resolve, reject) => {
14. **if** (!values && query.includes("?"))
15. reject("Your query includes placeholders, but you don't provide binding values");
16. connection.query({
17. sql: query,
18. values: values
19. }, **function** (error, results, fields) {
20. **if** (error) reject(error);
21. **else** {
22. resolve(results, fields);
23. }
24. })
25. })
26. }
27. *// export module*
28. 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. block content
3. h1= message
4. h2= error.status
5. pre #{error.stack}

**FILE: ./views/main.pug**

1. extends layout
2. block content
3. h1 PriCoSha
4. h2 Pivate Content Sharing
5. - if (loggedin)
6. h3 Hello #{user.fname}!
7. h4 Public Posts
8. button#new-group Create a New Friend Group
9. button#show-post Post New Content
10. button#logout Logout
11. 14 - else
12. h3 Recent Public Posts
13. div#public-posts
14. h3 Please Login to the System
15. 18 div
16. input#email(type="email")
17. div
18. input#pass(type="password")
19. div
20. button#login-btn Login
21. 25 br
22. 27 hr
23. 28 #footer
24. div PriCoSha
25. div An academic project by Theodore Kim and JinZhao Su
26. div Source: https://github.com/theo-kim/databases/project/pricosha
27. div Version 0.0.1
28. 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. router.get('/', (req, res, next) => {
6. **var** usertoken, user;
7. **if** (req.cookies.usertoken != **null**)
8. usertoken = req.cookies.usertoken;
9. **else** usertoken = **null**;
10. **if** (usertoken !== **null**) {
11. **try** {
12. user = jwt.verify(usertoken, jwtsecret);
13. }
14. **catch** (e) {
15. usertoken = **null**;
16. user = **null**;
17. }
18. }
19. **var** state = {
20. loggedin: usertoken !== **null**,
21. user: user
22. };
23. res.render('main', state);
24. });
25. 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. **var** salt = require('../../server\_config.json').passhashsalt;
6. **var** jwtsecret = require('../../server\_config.json').serversecret;
7. **var** query = require('../../db.js');
8. *// Log user in*
9. router.post('/login', (req, res, next) => {
10. **var** email = req.body.email;
11. *// Hash password*
12. **var** password = crypto.createHash('sha256')
13. .update(req.body.password + salt)
14. .digest('hex');
15. *// Perform query*
16. query("SELECT email, password, fname, lname FROM Person WHERE email=? AND password=?", [email, password])
17. .then((result) => {
18. **if** (result.length == 0) res.send('fail');
19. **else** {
20. *// Return a session token with user information*
21. res.send(jwt.sign({ fname: result[0].fname, lname: result[0].lname, email: result[0].email },
22. jwtsecret,
23. { expiresIn: '7d' }));
24. }
25. });
26. });
27. *// See content items*
28. router.get('/item', (req, res, next) => {
29. **var** token;
30. **if** (req.params.token == **null**) {
31. token = **null**;
32. }
33. **else** token = req.params.token;
34. *// If not token, user is not authenticated, only so public posts*
35. **if** (token === **null**) {
36. query("SELECT \* FROM ContentItem WHERE is\_pub=1 AND post\_time > DATE\_SUB(CURDATE(), INTERVAL 1 DAY)")
37. .then((result) => {
38. res.json(result);
39. });
40. }
41. *// Else ONLY show posts that the user is allowed to see*
42. **else** {
43. *// Implement this later*
44. }
45. });
46. *// Create content items*
47. router.post('/item', (req, res, next) => {
48. *// Implement this later*
49. });
50. *// Create a FriendGroup*
51. router.post('/group', (req, res, next) => {
52. *// Implement this later*
53. })
54. module.exports = router;

**FILE: ./public/scripts/main.js**

1. **var** mode = 0;
2. *// Check mode*
3. **if**($('#login-btn').length) {
4. mode = 0;
5. }
6. **else** {
7. mode = 1;
8. }
9. **if** (mode == 0) {
10. *// View all items the user can see, in this case, only public posts b/c*
11. *// the user is not authenticated*
12. $.get("/api/item", {}, **function**(items) {
13. **for** (**var** i = 0; i < items.length; ++i) {
14. $("#public-posts").append($(`<div **class**='contentitem'>
15. <span **class**='title'>${items[i]["item\_name"]}</span><br>
16. <span **class**='item-id'>Item ID: ${items[i]["item\_id"]}</span><br>
17. <span **class**='post-time'>Posted: ${items[i]["post\_time"]}</span><br>
18. <a href='${items[i]["file\_path"]}'>${items[i]["file\_path"]}</a><br>
19. <span **class**='post-email'>Posted By: ${items[i]["email\_post"]}</span><br>
20. </div>`));
21. }
22. });
23. *// Log user in*
24. $("#login-btn").click(**function**() {
25. $.post("/api/login", {email: $("#email").val(), password: $("#pass").val()},
26. **function**(res) {
27. **if** (res === "fail") alert("Incorrect User Email and/or Password");
28. **else** {
29. $.cookie("usertoken", res);
30. location.reload();
31. }
32. })
33. });
34. }
35. **else** {
36. **var** token = $.cookie("usertoken");
37. *// View all posts the user can see*
38. $.get("/api/item", { token: token }, **function**(items) {
39. *// Implement this later*
40. });
41. *// Create a new FriendGroup*
42. $("#new-group").click(**function**() {
43. *// Implement this later*
44. });
45. *// Show menu to create a new post*
46. $("#show-post").click(**function**() {
47. *// Implement this later*
48. });
49. *// Log user out of PriCoSha*
50. $("#logout").click(**function**() {
51. $.cookie("usertoken", 0);
52. location.reload();
53. });
54. }

**FILE: ./public/css/style.js**

1. **body** {
   * **font-family**: **sans-serif**;
2. **padding**: 0 !important;
3. **margin**: 0 !important;
4. }
5. **h1**, **h2**, **h3**, **h4** {
6. **padding**: 5px;
7. }
8. **button** {
9. **font-size** :18px;
10. **background**: #ddd;
11. **border**: 0;
12. **padding**: 5px 10px;
13. **margin**: 20px;
14. }
15. #footer {
16. **background**: #ddd;
17. **padding**: 10px;
18. **font-size**: 12px;
19. **margin**: 20px;
20. }
21. **.contentitem** {
22. **padding**: 10px;
23. **display**: **inline**-**block**;
24. **background**: #eee;
25. **margin**: 10px;
26. **font-size**: 18px;
27. }
28. **.title** {
29. **font-weight**: **bold**;
30. **font-size**: 24px;
31. **padding-bottom**: 5px;
32. }
33. **.item-id** {
34. **color**: #888;
35. **font-size**: 14px;
36. }
37. **.post-time** {
38. **color**: #333;
39. **font-size**: 16px;
40. **padding-bottom**: 10px;
41. }