

index.js

```
1  const express = require('express');
2  // Start
3  const session = require('express-session');
4  const helmet = require('helmet');
5  // End
6  const bodyParser = require('body-parser');
7  const compression = require('compression');
8  // Start
9  const path = require('path');
10 const dotenv = require('dotenv');
11 // End
12 const sqlite3 = require('sqlite3').verbose();
13 // Start
14 dotenv.config();
15 // End
16 const app = express();
17 const port = 3000;
18
19 // Set up SQLite
20 global.db = new sqlite3.Database('./database.db', function(err) {
21   if (err) {
22     console.error(err);
23     process.exit(1); // bail out we can't connect to the DB
24   } else {
25     console.log("Database connected");
26     global.db.run("PRAGMA foreign_keys=ON"); // tell SQLite to pay attention to foreign
key constraints
27   }
28 });
29
30 // Start
31 // Security Best Practices
32 app.use(helmet());
33 app.disable('x-powered-by');
34
35 // Middleware setup
36 app.use(compression()); // Compress responses
37 app.use(bodyParser.urlencoded({ extended: true })); // Parse URL-encoded bodies
38
39 app.set('view engine', 'ejs');
40
41 // Static file caching
42 app.use(express.static(path.join(__dirname, 'public'), { maxAge: '1y' }));
43
44 // Session management with secure cookies
45 app.use(session({
46   secret: process.env.SESSION_SECRET,
47   resave: false,
48   saveUninitialized: true,
49   cookie: {
50     httpOnly: true,
```

```
51     secure: process.env.NODE_ENV === 'production', // Ensure cookies are secure in
production
52     sameSite: 'lax',
53     maxAge: 60 * 60 * 1000 // 1 hour
54   }
55   }));
56
57   // Prevent caching for dynamic responses
58   app.use((req, res, next) => {
59     res.setHeader('Cache-Control', 'no-store');
60     next();
61   });
62
63   // Middleware and Routes setup
64   const authMiddleware = require('./middleware/auth');
65   const setUserMiddleware = require('./middleware/setUser');
66
67   const authRoutes = require('./routes/auth');
68   const authorRoutes = require('./routes/author');
69   const readerRoutes = require('./routes/reader');
70   const settingsRoutes = require('./routes/settings');
71
72   app.use(setUserMiddleware); // Ensure setUserMiddleware runs before routes
73
74   app.use('/auth', authRoutes); // Public route
75   app.use('/author', authMiddleware, authorRoutes); // Protected route
76   app.use('/reader', readerRoutes); // Public route
77   app.use('/settings', authMiddleware, settingsRoutes); // Protected route
78
79   // Route for getting user data
80   app.get('/api/user', (req, res) => {
81     if (req.session.isAuthenticated) {
82       res.json({ user: req.session.user });
83     } else {
84       res.status(401).json({ error: 'Unauthorized' });
85     }
86     res.setHeader('Cache-Control', 'no-store');
87   });
88
89   // Route for serving the main home page
90   app.get('/', (req, res) => {
91     res.render('mainHome');
92   });
93
94   // Start the server
95   app.listen(port, () => {
96     console.log(`Blogging tool listening at http://localhost:${port}`);
97   });
98   // End
```

db_schema.sql

```
1 PRAGMA foreign_keys=ON;
2
3 -- Start
4 CREATE TABLE IF NOT EXISTS users (
5     user_id INTEGER PRIMARY KEY AUTOINCREMENT,
6     user_name TEXT NOT NULL,
7     email TEXT UNIQUE NOT NULL,
8     password TEXT
9 );
10
11 CREATE TABLE IF NOT EXISTS articles (
12     article_id INTEGER PRIMARY KEY AUTOINCREMENT,
13     title TEXT NOT NULL,
14     content TEXT NOT NULL,
15     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
16     updated_at DATETIME DEFAULT CURRENT_TIMESTAMP,
17     published_at DATETIME,
18     author_id INTEGER,
19     views INTEGER DEFAULT 0,
20     FOREIGN KEY (author_id) REFERENCES users(user_id) ON DELETE CASCADE
21 );
22
23 CREATE TABLE IF NOT EXISTS comments (
24     comment_id INTEGER PRIMARY KEY AUTOINCREMENT,
25     article_id INTEGER,
26     commenter_name TEXT NOT NULL,
27     comment TEXT NOT NULL,
28     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
29     FOREIGN KEY (article_id) REFERENCES articles(article_id) ON DELETE CASCADE
30 );
31
32 CREATE TABLE IF NOT EXISTS likes (
33     like_id INTEGER PRIMARY KEY AUTOINCREMENT,
34     article_id INTEGER,
35     user_id INTEGER,
36     FOREIGN KEY (article_id) REFERENCES articles(article_id) ON DELETE CASCADE,
37     FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
38     UNIQUE(article_id, user_id)
39 );
40
41 CREATE TABLE IF NOT EXISTS settings (
42     setting_id INTEGER PRIMARY KEY AUTOINCREMENT,
43     blog_title TEXT NOT NULL,
44     author_name TEXT NOT NULL,
45     author_id INTEGER NOT NULL UNIQUE,
46     FOREIGN KEY (author_id) REFERENCES users(user_id) ON DELETE CASCADE
47 );
48
49 CREATE TABLE IF NOT EXISTS views (
50     view_id INTEGER PRIMARY KEY AUTOINCREMENT,
51     article_id INTEGER,
```

```
52     user_id INTEGER,  
53     created_at DATETIME DEFAULT CURRENT_TIMESTAMP,  
54     FOREIGN KEY (article_id) REFERENCES articles(article_id) ON DELETE CASCADE,  
55     FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE  
56 );  
57  
58 -- Insert a default user if not exists  
59 INSERT INTO users (user_name, email, password)  
60 SELECT 'Default Author', 'default@example.com', NULL  
61 WHERE NOT EXISTS (SELECT 1 FROM users);  
62  
63 -- Insert initial settings if not exists  
64 INSERT INTO settings (blog_title, author_name, author_id)  
65 SELECT 'My Blog', 'Author Name', user_id  
66 FROM users  
67 WHERE NOT EXISTS (SELECT 1 FROM settings);  
68 -- End
```

.env

```
1 | # Start
2 | SESSION_SECRET=
3 |
4 | GOOGLE_CLIENT_ID=
5 | GOOGLE_CLIENT_SECRET=
6 | # End
```

views\settingsPage.ejs

```
1 <!-- Start -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <title>Settings</title>
7
8   <!-- Add a favicon for the website to avoid 404 error message -->
9   <link rel="shortcut icon" href="#">
10
11   <!-- Loading CSS -->
12   <link rel="stylesheet" type="text/css" href="/main.css">
13 </head>
14 <body>
15   <!-- -----navbar----- -->
16   <nav class="navbar">
17     <h1>Settings Page</h1>
18     <!-- -----navbar Links----- -->
19     <ul>
20       <li><a href="/author">Back to Author Home</a></li>
21       <li><a href="/auth/logout">Logout</a></li>
22     </ul>
23   </nav>
24   <!-- -----Blog----- -->
25   <div class="blog-section">
26     <form action="/settings" method="post">
27       <label>Blog Title:</label>
28       <input type="text" name="blog_title" value="<%= settings.blog_title %>">
29       <label>Author Name:</label>
30       <input type="text" name="author_name" value="<%= settings.author_name %>">
31       <button type="submit">Save Settings</button>
32     </form>
33   </div>
34 </body>
35 </html>
36 <!-- End -->
```

views\register.ejs

```

1  <!-- Start -->
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <title>Register</title>
7
8      <!-- Add a favicon for the website to avoid 404 error message -->
9      <link rel="shortcut icon" href="#">
10
11     <!-- Loading CSS -->
12     <link rel="stylesheet" type="text/css" href="/main.css" />
13 </head>
14 <body>
15     <!-- -----navbar----- -->
16     <nav class="navbar">
17         <h1>Register</h1>
18     </nav>
19     <!-- -----register form----- -->
20     <div class="register-form">
21         <h2>Register and then login</h2>
22         <form action="/auth/register" method="post">
23             <label>User Name:</label>
24             <input type="text" name="user_name" required>
25             <label>Email:</label>
26             <input type="email" name="email" required>
27             <label>Password:</label>
28             <input type="password" name="password" required>
29             <button type="submit">Register</button>
30         </form>
31         <!-- -----Register error----- -->
32         <% if (errors && errors.length > 0) { %>
33             <ul id="registerError-text">
34                 <% errors.forEach(error => { %>
35                     <li><%= error.msg %></li>
36                 <% }) %>
37             </ul>
38         <% } %>
39         <!-- -----Link to back to login----- -->
40         <p>Back to <a href="/auth/login">Login</a></p>
41     </div>
42 </body>
43 </html>
44 <!-- End -->

```

views\readerHome.ejs

```
1 <!-- Start -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5     <meta charset="UTF-8">
6     <title><%= blogTitle %> - Reader Home</title>
7
8     <!-- Add a favicon for the website to avoid 404 error message -->
9     <link rel="shortcut icon" href="#">
10
11     <!-- Loading CSS -->
12     <link rel="stylesheet" type="text/css" href="/main.css">
13 </head>
14 <body>
15     <!-- -----navbar----- -
16     ->
17     <nav class="navbar">
18         <h1>Reader Home Page</h1>
19         <!-- -----navbar Links----- -->
20         <ul>
21             <li><a href="/">Main Home</a></li>
22             <li class="active">Reader Home</li>
23         </ul>
24     </nav>
25     <!-- -----Blog----- -->
26     <% if (articles.length > 0) { %>
27         <% articles.forEach(article => { %>
28             <div class="article-item">
29                 <h3><a href="/reader/article/<%= article.article_id %>"><%= article.title
30 %></a></h3>
31                 <p class="publication-date">
32                     Article by: <%= article.author_name %>
33                     Published on: <%= article.published_at %> PKT
34                 </p>
35             </div>
36             <% }) %>
37         <% } else { %>
38             <p class="no-article-item">No articles to display.</p>
39         <% } %>
40 </body>
41 </html>
42 <!-- End -->
```


views\mainHome.ejs

```
1 <!-- Start -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <title>Main Home</title>
7
8   <!-- Add a favicon for the website to avoid 404 error message -->
9   <link rel="shortcut icon" href="#">
10
11   <!-- Loading CSS -->
12   <link rel="stylesheet" type="text/css" href="/main.css">
13 </head>
14 <body>
15   <!-- -----navbar----- -->
16   <nav class="navbar">
17     <h1>Blogging Tool</h1>
18     <!-- -----navbar Links----- -->
19     <ul>
20       <li class="active">Home</li>
21       <li><a href="/author">Author Home Page</a></li>
22       <li><a href="/reader">Reader Home Page</a></li>
23     </ul>
24     <!-- -----Blog----- -->
25     <% if (user && user.user_name !== 'Guest') { %>
26       <div class="welcome-message">
27         Welcome, <%= user.user_name %>! <a href="/auth/logout">Logout</a>
28       </div>
29     <% } else { %>
30       <div class="welcome-message">
31         Welcome, Guest! <a href="/auth/login">Login</a>
32       </div>
33     <% } %>
34   </nav>
35   <div class="main-blog-articles">
36     <div class="blog-articles">
37       <h4>Welcome</h4>
38       <p>to the Blogging Tool</p>
39     </div>
40   </div>
41 </body>
42 </html>
43 <!-- End -->
```

views\login.ejs

```

1  <!-- Start -->
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <title>Login</title>
7
8      <!-- Add a favicon for the website to avoid 404 error message -->
9      <link rel="shortcut icon" href="#">
10
11      <!-- Loading CSS -->
12      <link rel="stylesheet" type="text/css" href="/main.css" />
13 </head>
14 <body>
15     <!-- -----navbar----- -->
16     <nav class="navbar">
17         <h1>Login</h1>
18         <!-- -----navbar Links----- -->
19         <ul>
20             <li><a href="/">Main Home</a></li>
21             <li class="active">Author Home Page</li>
22         </ul>
23     </nav>
24     <!-- -----login form----- -->
25     <div class="login-form">
26         <form action="/auth/login" method="post">
27             <label>Email:</label>
28             <input type="email" name="email" required>
29             <label>Password:</label>
30             <input type="password" name="password" required>
31             <button type="submit">Login</button>
32         </form>
33         <!-- -----Login error----- -->
34         <% if (errors && errors.length > 0) { %>
35             <ul id="loginError-text">
36                 <% errors.forEach(error => { %>
37                     <li><%= error.msg %></li>
38                 <% }) %>
39             </ul>
40         <% } %>
41         <!-- -----Other Login options----- -->
42         <p>Don't have an account? <a href="/auth/register">Register here</a> </p>
43         <p>Or</p>
44         <a href="/auth/google">Login with Google</a>
45     </div>
46 </body>
47 </html>
48 <!-- End -->

```

views\editArticle.ejs

```

1 <!-- Start -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <title>Edit Article</title>
7
8   <!-- Add a favicon for the website to avoid 404 error message -->
9   <link rel="shortcut icon" href="#">
10
11   <!-- Loading CSS -->
12   <link rel="stylesheet" type="text/css" href="/main.css">
13 </head>
14 <body>
15   <!-- -----navbar----- -->
16   <nav class="navbar">
17     <h1>Edit Article</h1>
18     <!-- -----navbar Links----- -->
19     <ul>
20       <li><a href="/author">Back to Author Home</a></li>
21     </ul>
22   </nav>
23   <!-- -----Blog----- -->
24   <div class="blog-section">
25     <div class="article-container">
26       <h5>Edit Article</h5>
27       <form class="editArticle-field" action="/author/edit/<%= article.article_id %>"
method="post">
28         <label>Title:</label>
29         <input type="text" name="title" value="<%= article.title %>" required>
30         <label>Content:</label>
31         <textarea name="content" required><%= article.content %></textarea>
32         <button type="submit">Save Changes</button>
33       </form>
34     </div>
35   </div>
36 </body>
37 </html>
38 <!-- End -->

```

views\authorHome.ejs

```
1 <!-- Start -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <title>Author Home</title>
7
8   <!-- Add a favicon for the website to avoid 404 error message -->
9   <link rel="shortcut icon" href="#">
10
11   <!-- Loading CSS -->
12   <link rel="stylesheet" type="text/css" href="/main.css">
13   <!-- Loading Javascript -->
14   <script src="/confirm.js" defer></script>
15 </head>
16 <body>
17   <!-- -----navbar----- -->
18   <nav class="navbar">
19     <h1><%= blogTitle %></h1>
20     <!-- -----navbar Links----- -->
21     <ul>
22       <li><a href="/">Main Home</a></li>
23       <li class="active">Author Home</li>
24       <li><a href="/settings">Settings</a></li>
25       <li><a href="/auth/logout">Logout</a></li>
26     </ul>
27   </nav>
28   <!-- -----Blog----- -->
29   <div class="main-blog-articles">
30     <div class="blog-articles">
31       <h4>Published Articles</h4>
32       <ul>
33         <% publishedArticles.forEach(article => { %>
34           <li class="published-articles">
35             <a href="/reader/article/<%= article.article_id %>">View <%=
36 article.title %></a> by <%= authorName %>
37             <p>Created at: <%=
38 moment.utc(article.created_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss') %> PKT</p>
39             <p>Last modified at: <%=
40 moment.utc(article.updated_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss') %> PKT</p>
41             <p>Published at: <%=
42 moment.utc(article.published_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss') %> PKT</p>
43             <form id="delete-published-<%= article.article_id %>"
44 action="/author/delete/<%= article.article_id %>" method="post">
45               <button type="button" class="delete-published-button" data-form-
46 id="delete-published-<%= article.article_id %>">Delete</button>
47             </form>
48           </li>
49         <% }) %>
50       </ul>
```

```

45     </div>
46     <div class="blog-articles line-between-publish-and-draft">
47         <h4>Draft Articles</h4>
48         <ul>
49             <% draftArticles.forEach(article => { %>
50                 <li class="draft-articles">
51                     <a href="/author/edit/<%= article.article_id %">Click to Edit <%=
article.title %></a> by <%= authorName %>
52                     <p>Created at: <%=
moment.utc(article.created_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss') %> PKT</p>
53                     <p>Last modified at: <%=
moment.utc(article.updated_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss') %> PKT</p>
54                     <form id="publish-draft-<%= article.article_id %">
action="/author/publish/<%= article.article_id %>" method="post">
55                         <button type="button" class="publish-button" data-form-id="publish-
draft-<%= article.article_id %">Publish</button>
56                     </form>
57                     <% if (publishMessage) { %>
58                         <p class="publish-message"><%= publishMessage %></p>
59                     <% }; %>
60                     <form id="delete-draft-<%= article.article_id %">
action="/author/delete/<%= article.article_id %>" method="post">
61                         <button type="button" class="delete-draft-button" data-form-
id="delete-draft-<%= article.article_id %">Delete</button>
62                     </form>
63                 </li>
64             <% }) %>
65         </ul>
66         <form action="/author/create-draft" method="post">
67             <button type="submit">Create new draft</button>
68         </form>
69     </div>
70 </div>
71
72 <!-- Modal for confirmation dialog -->
73 <div id="confirmationModal" class="modal">
74     <div class="modal-content">
75         <p id="confirmationMessage"></p>
76         <button id="confirmYes">Yes</button>
77         <button id="confirmNo">No</button>
78     </div>
79 </div>
80 <style>
81     .modal /* Styles for the modal container */
82     {
83         display: none; /* Initially hide the modal */
84         position: fixed; /* Fixed position so it stays in place regardless of scrolling
*/
85         z-index: 1; /* Ensure modal is on top of other content */
86         left: 0;
87         top: 0;
88         width: 100%; /* Full width of the viewport */
89         height: 100%; /* Full height of the viewport */
90         overflow: auto; /* Enable scrolling if content exceeds viewport */
91         background-color: rgb(0,0,0); /* Fallback color for older browsers */

```

```
92         background-color: rgba(0,0,0,0.4); /* Transparent black overlay */
93     }
94
95     .modal-content /* Styles for the modal content */
96     {
97         background-color: #fefefe; /* White background for the modal content */
98         margin: 20% auto; /* Center the modal vertically & horizontally */
99         padding: 30px; /* Padding inside the modal content */
100        border: 1px solid #4d388b;
101        width: 30%;
102        text-align: center;
103    }
104
105    #confirmYes, #confirmNo /* Styles for 'Yes' and 'No' buttons */
106    {
107        margin-top: 10px; /* Add some space above the buttons */
108    }
109    </style>
110 </body>
111 </html>
112 <!-- End -->
```

views\articlePage.ejs

```

1 <!-- Start -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <title><%= article.title %></title>
7
8   <!-- Add a favicon for the website to avoid 404 error message -->
9   <link rel="shortcut icon" href="#">
10
11   <!-- Loading CSS -->
12   <link rel="stylesheet" type="text/css" href="/main.css">
13 </head>
14 <body>
15   <!-- -----navbar----- -->
16   <nav class="navbar">
17     <h1><%= article.title %></h1>
18     <!-- -----navbar Links----- -->
19     <ul>
20       <li><a href="/reader">Back to Reader Home</a></li>
21       <li><a href="/author">Back to Author Home</a></li>
22     </ul>
23   </nav>
24   <!-- -----Blog----- -->
25   <div class="blog-section">
26     <div class="article-container">
27       <h5><%= article.title %></h5>
28       <% if (article.showAuthorName) { %>
29         <p>By <%= article.author %></p>
30       <% } %>
31       <p class="p-article"><%= article.content %></p>
32       <hr>
33       <p class="article-info">Published on: <%=
moment.utc(article.published_at).tz('Asia/Karachi').format('YYYY-MM-DD HH:mm:ss') %> PKT</p>
34       <p class="article-info">Views: <%= article.views %> | Likes: <%= likeCount %></p>
35       <% if (likeMessage) { %>
36         <p class="like-message"><%= likeMessage %></p>
37       <% } %>
38       <form action="/reader/article/<%= article.article_id %>/like" method="post">
39         <button type="submit">Like 👍</button>
40       </form>
41       <form action="/reader/article/<%= article.article_id %>/comment" method="post">
42         <label>Name:</label>
43         <input type="text" name="commenter_name" required>
44         <label>Comment:</label>
45         <textarea name="comment" required></textarea>
46         <button type="submit">Add Comment</button>
47       </form>
48       <ul>
49         <% comments.forEach(comment => { %>
50           <li>

```

```
51         <strong><%= comment.commenter_name %>:</strong> <%= comment.comment %> - <%=  
moment.utc(comment.created_at).tz('Asia/Karachi').format('YYYY-MM-DD HH:mm:ss') %> PKT  
52         </li>  
53     <% }>; %>  
54 </ul>  
55 </div>  
56 </div>  
57 </body>  
58 </html>  
59 <!-- End -->
```


routes\settings.js

```
1 // Start
2 const express = require('express');
3
4 const router = express.Router();
5
6 /**
7  * Route serving the settings page.
8  * @name get/settings
9  * @function
10  * @memberof module:routers/settings~settingsRouter
11  * @inner
12  * @param {string} path - Express path
13  */
14 router.get('/', (req, res) => {
15     const query = "SELECT * FROM settings WHERE author_id = ?";
16     // Database interaction: Fetching the settings of the current user
17     // Inputs: Author ID from the session
18     // Outputs: Settings of the current user if found, else null
19     global.db.get(query, [req.session.user.user_id], (err, settings) => {
20         if (err)
21         {
22             console.error("Error retrieving settings:", err);
23             res.status(500).send("Error retrieving settings");
24             return;
25         }
26         if (!settings)
27         {
28             settings = { blog_title: 'Default Blog Title', author_name: 'Set your name' };
29         }
30         res.render('settingsPage', { settings: settings });
31     });
32 });
33
34 /**
35  * Route for updating the settings.
36  * @name post/settings
37  * @function
38  * @memberof module:routers/settings~settingsRouter
39  * @inner
40  * @param {string} path - Express path
41  */
42 router.post('/', (req, res) => {
43     const blogTitle = req.body.blog_title;
44     const authorName = req.body.author_name;
45     const authorId = req.session.user.user_id;
46     console.log("Received data:", { blogTitle, authorName, authorId });
47     if (!blogTitle || !authorName || !authorId)
48     {
49         console.error("Missing required fields");
50         return res.status(400).send("Missing required fields");
51     }
52 });
```

```
52     const query = `
53         INSERT INTO settings (blog_title, author_name, author_id)
54         VALUES (?, ?, ?)
55         ON CONFLICT(author_id) DO UPDATE SET
56         blog_title = excluded.blog_title,
57         author_name = excluded.author_name`;
58     const params = [blogTitle, authorName, authorId];
59     // Database interaction: Inserting or updating the settings in the database
60     // Inputs: Blog title, author name, and author ID from the request body and session
61     // Outputs: None
62     global.db.run(query, params, function(err)
63     {
64         if (err)
65         {
66             console.error("Error updating settings:", err);
67             res.status(500).send("Error updating settings");
68             return;
69         }
70         console.log("Settings updated successfully");
71         res.redirect('/author');
72     });
73 });
74
75 module.exports = router;
76 // End
```

routes\reader.js

```
1 // Start
2 const express = require('express');
3 const moment = require('moment-timezone');
4 const TIMEZONE = 'Asia/Karachi';
5
6 const router = express.Router();
7
8 /**
9  * Route serving the reader's home page.
10  * @name get/reader
11  * @function
12  * @memberof module:routers/reader~readerRouter
13  * @inner
14  * @param {string} path - Express path
15  */
16 router.get('/', (req, res) => {
17   const articlesQuery = `
18     SELECT articles.*, IFNULL(settings.author_name, users.user_name) AS author_name
19     FROM articles
20     LEFT JOIN settings ON articles.author_id = settings.author_id
21     LEFT JOIN users ON articles.author_id = users.user_id
22     WHERE published_at IS NOT NULL
23     ORDER BY published_at DESC`;
24   // Database interaction: Fetching all published articles
25   // Inputs: None
26   // Outputs: All published articles
27   global.db.all(articlesQuery, [], (err, articles) => {
28     if (err)
29     {
30       console.error(err);
31       return res.status(500).send("Error retrieving articles");
32     }
33     articles.forEach(article => {
34       article.published_at =
35       moment.utc(article.published_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss');
36       res.render('readerHome', {
37         blogTitle: 'Blog',
38         articles: articles,
39         moment: moment
40       });
41     });
42   });
43
44 /**
45  * Route for viewing an article.
46  * @name get/reader/article/:id
47  * @function
48  * @memberof module:routers/reader~readerRouter
49  * @inner
50  * @param {string} path - Express path
51  */
```

```
52 router.get('/article/:id', (req, res) => {
53   const articleQuery = "SELECT articles.*, users.user_name FROM articles JOIN users ON
articles.author_id = users.user_id WHERE article_id = ?";
54   const commentsQuery = "SELECT * FROM comments WHERE article_id = ? ORDER BY created_at
DESC";
55   const likesQuery = "SELECT COUNT(*) AS like_count FROM likes WHERE article_id = ?";
56   const viewsInsertQuery = "INSERT INTO views (article_id, user_id) VALUES (?, ?)";
57   const viewsUpdateQuery = "UPDATE articles SET views = views + 1 WHERE article_id = ?";
58   // Database interaction: Fetching the article
59   // Inputs: Article ID from the URL parameters
60   // Outputs: Article data if found, else null
61   global.db.get(articleQuery, [req.params.id], (err, article) => {
62     if (err)
63     {
64       console.error(err);
65       res.status(500).send("Error retrieving article");
66       return;
67     }
68     if (!req.session.user)
69     {
70       req.session.user = { user_id: null };
71     }
72     if (!req.session.viewedArticles)
73     {
74       req.session.viewedArticles = [];
75     }
76     if (!req.session.viewedArticles.includes(req.params.id))
77     {
78       // Database interaction: Inserting a view into the database
79       // Inputs: Article ID from the URL parameters and user ID from the session
80       // Outputs: None
81       global.db.run(viewsInsertQuery, [req.params.id, req.session.user ?
req.session.user.user_id : null], (err) => {
82         if (err)
83         {
84           console.error(err);
85         }
86       });
87       // Database interaction: Updating the views count of the article in the database
88       // Inputs: Article ID from the URL parameters
89       // Outputs: None
90       global.db.run(viewsUpdateQuery, [req.params.id], (err) => {
91         if (err)
92         {
93           console.error(err);
94         }
95       });
96       req.session.viewedArticles.push(req.params.id);
97     }
98     // Database interaction: Fetching the comments of the article
99     // Inputs: Article ID from the URL parameters
100    // Outputs: Comments of the article if found, else null
101    global.db.all(commentsQuery, [req.params.id], (err, comments) => {
102      if (err)
```

```

103     {
104         console.error(err);
105         res.status(500).send("Error retrieving comments");
106         return;
107     }
108     // Database interaction: Fetching the likes count of the article
109     // Inputs: Article ID from the URL parameters
110     // Outputs: Likes count of the article
111     global.db.get(likesQuery, [req.params.id], (err, likeData) => {
112         if (err)
113         {
114             console.error(err);
115             res.status(500).send("Error retrieving likes");
116             return;
117         }
118         article.published_at =
moment(article.published_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss');
119         comments.forEach(comment => {
120             comment.created_at =
moment(comment.created_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss');
121         });
122         const likeMessage = req.session.likeMessage;
123         req.session.likeMessage = null;
124         res.render('articlePage', {
125             article: article,
126             comments: comments,
127             likeCount: likeData.like_count,
128             moment: moment,
129             likeMessage: likeMessage
130         });
131     });
132 });
133 });
134 });
135
136 /**
137  * Route for liking an article.
138  * @name post/reader/article/:id/like
139  * @function
140  * @memberof module:routers/reader~readerRouter
141  * @inner
142  * @param {string} path - Express path
143  */
144 router.post('/article/:id/like', (req, res) => {
145     const userId = req.session.user ? req.session.user.user_id : null;
146     if (!userId)
147     {
148         req.session.likeMessage = 'Please log in to like articles.';
149         return res.redirect(`/reader/article/${req.params.id}`);
150     }
151     const checkLikeQuery = "SELECT COUNT(*) AS count FROM likes WHERE article_id = ? AND
user_id = ?";
152     const insertLikeQuery = "INSERT INTO likes (article_id, user_id) VALUES (?, ?)";

```

```
153 // Database interaction: Checking if the user has already liked the article and
inserting a like if not
154 // Inputs: Article ID from the URL parameters and user ID from the session
155 // Outputs: None
156 global.db.get(checkLikeQuery, [req.params.id, userId], (err, row) => {
157     if (err)
158     {
159         console.error(err);
160         res.status(500).send("Error checking like status");
161         return;
162     }
163     if (row.count > 0)
164     {
165         return res.redirect(`/reader/article/${req.params.id}`);
166     }
167     // Database interaction: Inserting a like into the database
168     // Inputs: Article ID from the URL parameters and user ID from the session
169     // Outputs: None
170     global.db.run(insertLikeQuery, [req.params.id, userId], (err) => {
171         if (err)
172         {
173             console.error(err);
174             res.status(500).send("Error liking article");
175             return;
176         }
177         res.redirect(`/reader/article/${req.params.id}`);
178     });
179 });
180 });
181
182 /**
183  * Route for commenting on an article.
184  * @name post/reader/article/:id/comment
185  * @function
186  * @memberof module:routers/reader~readerRouter
187  * @inner
188  * @param {string} path - Express path
189  */
190 router.post('/article/:id/comment', (req, res) => {
191     const query = "INSERT INTO comments (article_id, commenter_name, comment) VALUES (?, ?, ?)";
192     const params = [req.params.id, req.body.commenter_name, req.body.comment];
193     // Database interaction: Inserting a new comment into the database
194     // Inputs: Article ID from the URL parameters, commenter name, and comment from the
request body
195     // Outputs: None
196     global.db.run(query, params, (err) => {
197         if (err)
198         {
199             console.error(err);
200             res.status(500).send("Error adding comment");
201             return;
202         }
203         res.redirect(`/reader/article/${req.params.id}`);
```

```
204     });  
205   });  
206  
207   module.exports = router;  
208   // End
```

routes\author.js

```
1 // Start
2 const express = require('express');
3 const moment = require('moment-timezone');
4 const { body, validationResult } = require('express-validator');
5 const TIMEZONE = 'Asia/Karachi';
6
7 const router = express.Router();
8
9 /**
10  * Route serving the author's home page.
11  * @name get/author
12  * @function
13  * @memberof module:routers/author~authorRouter
14  * @inner
15  * @param {string} path - Express path
16  */
17 router.get('/', (req, res) => {
18   const queryPublished = `
19     SELECT articles.*, users.user_name
20     FROM articles
21     JOIN users ON articles.author_id = users.user_id
22     WHERE published_at IS NOT NULL AND articles.author_id = ?`;
23   const queryDrafts = `
24     SELECT articles.*, users.user_name
25     FROM articles
26     JOIN users ON articles.author_id = users.user_id
27     WHERE published_at IS NULL AND articles.author_id = ?`;
28   const querySettings = "SELECT * FROM settings WHERE author_id = ?";
29   // Database interaction: Fetching published articles, drafts & any settings of the
   current user
30   // Inputs: User ID from the session
31   // Outputs: Published articles, drafts & any setting of the current user
32   global.db.all(queryPublished, [req.session.user.user_id], (err, publishedArticles) => {
33     if (err) throw err;
34     global.db.all(queryDrafts, [req.session.user.user_id], (err, draftArticles) => {
35       if (err) throw err;
36       global.db.get(querySettings, [req.session.user.user_id], (err, settings) => {
37         if (err) throw err;
38         publishedArticles.forEach(article => {
39           article.created_at =
40             moment(article.created_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss');
41           article.updated_at =
42             moment(article.updated_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss');
43           article.published_at =
44             moment(article.published_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss');
45         });
46         draftArticles.forEach(article => {
47           article.created_at =
48             moment(article.created_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss');
49           article.updated_at =
50             moment(article.updated_at).tz(TIMEZONE).format('YYYY-MM-DD HH:mm:ss');
```



```

47         res.render('authorHome', {
48             publishedArticles: publishedArticles,
49             draftArticles: draftArticles,
50             TIMEZONE: TIMEZONE,
51             moment: moment,
52             blogTitle: settings ? settings.blog_title : 'Set your Blog Title in
Settings',
53             authorName: settings ? settings.author_name : 'Default Author Name',
54             publishMessage: res.locals.publishMessage
55         });
56     });
57 });
58 });
59 });
60
61 /**
62  * Route for creating a new draft.
63  * @name post/author/create-draft
64  * @function
65  * @memberof module:routers/author~authorRouter
66  * @inner
67  * @param {string} path - Express path
68  */
69 router.post('/create-draft', (req, res) => {
70     if (!req.session.user || !req.session.user.user_id)
71     {
72         res.status(401).send("Unauthorized");
73         return;
74     }
75     const query = "INSERT INTO articles (title, content, author_id) VALUES (?, ?, ?)";
76     const params = ['New Draft', '', req.session.user.user_id];
77     // Database interaction: Inserting a new draft into the database
78     // Inputs: Title, content, and author ID
79     // Outputs: None
80     global.db.run(query, params, function(err)
81     {
82         if (err)
83         {
84             console.error(err);
85             res.status(500).send("Error creating draft");
86             return;
87         }
88         res.redirect(`/author/edit/${this.lastID}`);
89     });
90 });
91
92 /**
93  * Route for editing an article.
94  * @name get/author/edit/:id
95  * @function
96  * @memberof module:routers/author~authorRouter
97  * @inner
98  * @param {string} path - Express path
99  */

```

```

100 router.get('/edit/:id', (req, res) => {
101     const query = "SELECT * FROM articles WHERE article_id = ? AND author_id = ?";
102     // Database interaction: Fetching the article to be edited
103     // Inputs: Article ID from the URL parameters and author ID from the session
104     // Outputs: Article data if found, else null
105     global.db.get(query, [req.params.id, req.session.user.user_id], (err, article) => {
106         if (err)
107         {
108             console.error(err);
109             res.status(500).send("Error retrieving article");
110             return;
111         }
112         res.render('editArticle', { article: article, errors: [] });
113     });
114 });
115
116 /**
117  * Route for updating an article.
118  * @name post/author/edit/:id
119  * @function
120  * @memberof module:routers/author~authorRouter
121  * @inner
122  * @param {string} path - Express path
123  * @param {function} middleware - Express middleware
124  */
125 router.post('/edit/:id', [
126     body('title').notEmpty().withMessage('Title is required'),
127     body('content').notEmpty().withMessage('Content is required')
128 ],
129 (req, res) => {
130     const errors = validationResult(req);
131     if (!errors.isEmpty())
132     {
133         const query = "SELECT * FROM articles WHERE article_id = ? AND author_id = ?";
134         // Database interaction: Fetching the article to be updated
135         // Inputs: Article ID from the URL parameters and author ID from the session
136         // Outputs: Article data if found, else null
137         global.db.get(query, [req.params.id, req.session.user.user_id], (err, article)
=> {
138             if (err)
139             {
140                 console.error(err);
141                 res.status(500).send("Error retrieving article");
142                 return;
143             }
144             res.render('editArticle', {
145                 article: article,
146                 errors: errors.array()
147             });
148         });
149         return;
150     }
151     const query = "UPDATE articles SET title = ?, content = ?, updated_at =
CURRENT_TIMESTAMP WHERE article_id = ? AND author_id = ?";

```

```
152     const params = [req.body.title, req.body.content, req.params.id,  
req.session.user.user_id];  
153     // Database interaction: Updating the article in the database  
154     // Inputs: Title, content, article ID from the URL parameters, and author ID from  
the session  
155     // Outputs: None  
156     global.db.run(query, params, (err) => {  
157         if (err)  
158         {  
159             console.error(err);  
160             res.status(500).send("Error updating article");  
161             return;  
162         }  
163         res.redirect('/author');  
164     });  
165 });  
166  
167 /**  
168  * Route for publishing an article.  
169  * @name post/author/publish/:id  
170  * @function  
171  * @memberof module:routers/author~authorRouter  
172  * @inner  
173  * @param {string} path - Express path  
174  */  
175 router.post('/publish/:id', (req, res) => {  
176     const query = "SELECT title, content FROM articles WHERE article_id = ? AND author_id =  
?";  
177     // Database interaction: Fetching the title and content of the article to be published  
178     // Inputs: Article ID from the URL parameters and author ID from the session  
179     // Outputs: Title and content of the article if found, else null  
180     global.db.get(query, [req.params.id, req.session.user.user_id], (err, article) => {  
181         if (err)  
182         {  
183             console.error(err);  
184             res.status(500).send("Error retrieving article");  
185             return;  
186         }  
187         if (!article.title || !article.content) {  
188             req.session.publishMessage = 'Title and content are required to publish the  
article.';  
189             res.redirect('/author');  
190             return;  
191         }  
192         const updateQuery = "UPDATE articles SET published_at = CURRENT_TIMESTAMP WHERE  
article_id = ? AND author_id = ?";  
193         // Database interaction: Updating the published_at timestamp of the article in the  
database  
194         // Inputs: Article ID from the URL parameters and author ID from the session  
195         // Outputs: None  
196         global.db.run(updateQuery, [req.params.id, req.session.user.user_id], (err) => {  
197             if (err)  
198             {  
199                 console.error(err);
```

```
200         res.status(500).send("Error publishing article");
201         return;
202     }
203     res.redirect('/author');
204 });
205 });
206 });
207
208 /**
209  * Route for deleting an article.
210  * @name post/author/delete/:id
211  * @function
212  * @memberof module:routers/author~authorRouter
213  * @inner
214  * @param {string} path - Express path
215  */
216 router.post('/delete/:id', (req, res) => {
217     const query = "DELETE FROM articles WHERE article_id = ? AND author_id = ?";
218     // Database interaction: Deleting the article from the database
219     // Inputs: Article ID from the URL parameters and author ID from the session
220     // Outputs: None
221     global.db.run(query, [req.params.id, req.session.user.user_id], (err) => {
222         if (err)
223         {
224             console.error(err);
225             res.status(500).send("Error deleting article");
226             return;
227         }
228         res.redirect('/author');
229     });
230 });
231
232 module.exports = router;
233 // End
```

routes\auth.js

```
1 // Start
2 const express = require('express');
3 const bcrypt = require('bcrypt');
4 const passport = require('passport');
5 const { body, validationResult } = require('express-validator');
6 require('dotenv').config();
7
8 const router = express.Router();
9
10 /**
11  * Route serving login form.
12  * @name get/auth/login
13  * @function
14  * @memberof module:routers/auth~authRouter
15  * @inner
16  * @param {string} path - Express path
17  */
18 router.get('/login', (req, res) => {
19   res.render('login', { errors: [] });
20 });
21
22 /**
23  * Route processing login form.
24  * @name post/auth/login
25  * @function
26  * @memberof module:routers/auth~authRouter
27  * @inner
28  * @param {string} path - Express path
29  * @param {function} middleware - Express middleware
30  */
31 router.post('/login',
32   body('email').isEmail().withMessage('Enter a valid email').normalizeEmail(),
33   body('password').notEmpty().withMessage('Password is required').trim().escape(),
34   (req, res) => {
35     const errors = validationResult(req);
36     if (!errors.isEmpty())
37     {
38       return res.status(400).render('login', { errors: errors.array() });
39     }
40     const { email, password } = req.body;
41     const query = "SELECT * FROM users WHERE email = ?";
42     // Database interaction: Fetching user with the provided email
43     // Inputs: Email provided by the user
44     // Outputs: User data if found, else null
45     global.db.get(query, [email], async (err, user) => {
46       if (err || !user) {
47         return res.status(401).render('login', { errors: [{ msg: 'Invalid credentials' }] });
48       }
49       // Compare hashed password for regular users
50       const passwordMatch = await bcrypt.compare(password, user.password);
51       if (passwordMatch)
```

```
52     {
53         req.session.isAuthenticated = true;
54         req.session.user = user;
55         return res.redirect('/author');
56     }
57     else
58     {
59         return res.status(401).render('login', { errors: [{ msg: 'Invalid
credentials' }] });
60     }
61 });
62 });
63
64 /**
65  * Route serving register form.
66  * @name get/auth/register
67  * @function
68  * @memberof module:routers/auth~authRouter
69  * @inner
70  * @param {string} path - Express path
71  */
72 router.get('/register', (req, res) => {
73     res.render('register', { errors: [] });
74 });
75
76 /**
77  * Route processing register form.
78  * @name post/auth/register
79  * @function
80  * @memberof module:routers/auth~authRouter
81  * @inner
82  * @param {string} path - Express path
83  * @param {function} middleware - Express middleware
84  */
85 router.post('/register',
86     body('user_name').notEmpty().withMessage('User name is required').trim().escape(),
87     body('email').isEmail().withMessage('Enter a valid email').normalizeEmail(),
88     body('password').isLength({ min: 6 }).withMessage('Password must be at least 6
characters long').trim().escape(),
89     async (req, res) => {
90         const errors = validationResult(req);
91         if (!errors.isEmpty())
92         {
93             return res.status(400).render('register', { errors: errors.array() });
94         }
95         const { user_name, email, password } = req.body;
96         const hashedPassword = await bcrypt.hash(password, 10);
97         const query = "INSERT INTO users (user_name, email, password) VALUES (?, ?, ?)";
98         // Database interaction: Inserting new user into the database
99         // Inputs: User name, email, and hashed password provided by the user
100        // Outputs: None
101        global.db.run(query, [user_name, email, hashedPassword], function(err)
102        {
103            if (err)
```

```

104         {
105             return res.status(500).render('register', { errors: [{ msg: 'Error
registering user' }] });
106         }
107         res.redirect('/auth/login');
108     });
109 });
110
111 const GoogleStrategy = require('passport-google-oauth20').Strategy;
112 passport.use(new GoogleStrategy({
113     clientID: process.env.GOOGLE_CLIENT_ID,
114     clientSecret: process.env.GOOGLE_CLIENT_SECRET,
115     callbackURL: "http://localhost:3000/auth/google/callback" },
116     (accessToken, refreshToken, profile, done) => {
117         const email = profile.emails[0].value;
118         const query = "SELECT * FROM users WHERE email = ?";
119         // Database interaction: Fetching user with the provided email
120         // Inputs: Email provided by Google OAuth
121         // Outputs: User data if found, else null
122         global.db.get(query, [email], (err, user) => {
123             if (user)
124             {
125                 return done(null, user);
126             }
127             else
128             {
129                 const newUser = {
130                     user_name: profile.displayName,
131                     email: email,
132                     password: null
133                 };
134                 const insertQuery = "INSERT INTO users (user_name, email, password) VALUES (?,
?, ?)";
135                 // Database interaction: Inserting new user into the database
136                 // Inputs: User name and email provided by Google OAuth
137                 // Outputs: None
138                 global.db.run(insertQuery, [newUser.user_name, newUser.email, newUser.password],
function(err)
139                 {
140                     if (err)
141                     {
142                         return done(err, null);
143                     }
144                     newUser.user_id = this.lastID;
145                     return done(null, newUser);
146                 });
147             }
148         });
149     });
150
151 /**
152  * Route for Google OAuth.
153  * @name get/auth/google
154  * @function

```




```
155 * @memberof module:routers/auth~authRouter
156 * @inner
157 * @param {string} path - Express path
158 * @param {function} middleware - Express middleware
159 */
160 router.get('/google',
161   passport.authenticate('google', { scope: ['profile', 'email'] }));
162
163 /**
164 * Route for Google OAuth callback.
165 * @name get/auth/google/callback
166 * @function
167 * @memberof module:routers/auth~authRouter
168 * @inner
169 * @param {string} path - Express path
170 * @param {function} middleware - Express middleware
171 */
172 router.get('/google/callback',
173   passport.authenticate('google', { failureRedirect: '/auth/login' }),
174   (req, res) => {
175     req.session.isAuthenticated = true;
176     req.session.user = req.user;
177     res.redirect('/author');
178   });
179
180 passport.serializeUser((user, done) => {
181   done(null, user.user_id);
182 });
183
184 passport.deserializeUser((id, done) => {
185   const query = "SELECT * FROM users WHERE user_id = ?";
186   // Database interaction: Fetching user with the provided user ID
187   // Inputs: User ID provided by Passport.js
188   // Outputs: User data if found, else null
189   global.db.get(query, [id], (err, user) => {
190     done(err, user);
191   });
192 });
193
194 /**
195 * Route for logout.
196 * @name get/auth/logout
197 * @function
198 * @memberof module:routers/auth~authRouter
199 * @inner
200 * @param {string} path - Express path
201 */
202 router.get('/logout', (req, res) => {
203   req.session.destroy((err) => {
204     if (err)
205     {
206       console.log(err);
207     }
208     else
```










```
209      {
210          res.clearCookie('connect.sid'); // Clear the session cookie
211          res.redirect('/auth/login');
212      }
213  });
214 });
215
216 module.exports = router;
217 // End
```




public\main.css

```

1 // Start
2 /*
3     Note: for styling, I have reused most contents from my Final Assignment of previous module
4     named CM1040 Web Development
5     Another thing to note is <h1>, <h2> and so on might look disordered in ejs templates & it
6     is
7     because I have used them in order in CSS instead
8 */
9 /* =====
10    - Global (like body, navbar, default settings etc) CSS style for all pages
11    ===== */
12
13 body, html
14 {
15     font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
16     margin: 0; /* Remove default margin */
17     padding: 0; /* Remove default padding */
18     overflow-x: hidden; /*to prevent any possible horizontal scroll bar*/
19     background-color:  lightcyan
20     position: relative; /*stay Relative to its positon*/
21 }
22
23 /* Making & Adjusting default settings-----
24 */
25 /* Paragraphs */
26 p
27 {
28     padding: 0px; /* Remove default padding & margin */
29     margin: 0;
30 }
31
32 /* Links */
33 li
34 {
35     list-style: none; /* Removes default dot on links*/
36 }
37
38 li a:hover /*when mouse is dragged on the link*/
39 {
40     color:  red
41     text-decoration: underline;
42 }
43
44 a
45 {
46     text-decoration: none; /* Removes default underline on links*/
47     color:  blue
48 }
49
50 a:hover /*when mouse is dragged on the link*/

```

```
51 {
52   color:  red
53   text-decoration: underline;
54 }
55
56 /* Button */
57 button
58 {
59   font-weight: bolder; /*weight of texts on buttons*/
60   background-color:  white /*BG color of buttons*/
61   color:  #3db8c0; /*color of texts on buttons*/
62   border: 1px solid #4d388b; /*border size & color*/
63   cursor: pointer; /*To make buttons look clickable*/
64 }
65
66 button:hover /*when mouse is dragged on the button*/
67 {
68   background-color: rgb(221, 221, 221); /*BG color of buttons*/
69 }
70
71 button:active /*when mouse is clicked on the button*/
72 {
73   background-color:  #ffe7c2; /*BG color of buttons*/
74 }
75
76 /* NAVBAR-----*/
77 .navbar
78 {
79   position: fixed; /*sticky navbar*/
80   z-index: 9999; /*To make the navbar always appear first*/
81   top: 0;
82   width: 100%;
83   background-color:  #3db8c0;
84   padding: 30px;
85   text-align: center;
86   box-shadow: 5px 4px 20px grey; /* shadow effect from below*/
87 }
88
89 h1
90 {
91   margin: 0;
92   font-size: 36px;
93   color:  white
94   text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5); /* shadow effect to the text */
95   transition: color 0.3s ease; /* smooth transition effect to the text color on mouse drag*/
96 }
97
98 h1:hover
99 {
100   color:  #ff9900; /* Changes the text color when mouse is dragged over h1*/
101 }
102
103 /* NAVBAR links */
104 nav ul li
```

```
105 {
106     display: inline; /*Horizontal display of text*/
107     margin-right: 20px; /*Space between links*/
108 }
109
110 nav ul li a
111 {
112     color:  white /*Non-active links color*/
113     font-size: 20px; /*Non-active links size*/
114     transition: color 0.3s ease;
115     display: inline-block; /* Ensure links are displayed */
116 }
117
118 nav ul li a:hover /*when mouse is dragged over links*/
119 {
120     color:  #ff9900; /* Changes the text color of the navbar links from blue */
121     text-decoration: none; /*remove default underline decoration*/
122     font-weight: bolder; /*Make text bolder*/
123 }
124
125 nav ul li.active /*active links size & weight*/
126 {
127     color:  white
128     font-size: 25px;
129     font-weight: bolder;
130 }
131
132 /* =====
133 - articlePage.ejs CSS style
134 ===== */
135
136 .p-article /*article blog content*/
137 {
138     font-weight: bolder;
139     font-size: 20px;
140     color: rgb(43, 0, 160);
141     line-height: 1.4;
142 }
143
144 .article-info /*when was article published, how many likes & views it have*/
145 {
146     font-size: 20px;
147     font-weight: bolder;
148     color:  #000000;
149     line-height: 1.5;
150 }
151
152 /* =====
153 - authorHome.ejs CSS style
154 ===== */
155
156 .published-articles, .draft-articles /* 2 contents (published, draft)*/
157 {
158     border: 1px solid #4d388b; /*border size & color*/
```

```
159     border-radius: 10px; /*rounded corners of border*/
160     margin: 3% 5% auto; /*3% sets the top margin, 5% sets the right & left margins, & auto
sets the bottom margin*/
161     padding: 10px; /* Optional: Add some spacing between articles */
162 }
163
164 .line-between-publish-and-draft /*vertical line separating 2 articles content*/
165 {
166     border-left: 1px solid #007e92;
167 }
168
169 /*Display a message if publish button is clicked with empty title and/or content*/
170 .blog-articles .publish-message
171 {
172     font-weight: normal;
173     color: ■ #000;
174     margin-right: 25%;
175     font-size: 15px;
176 }
177
178 /* =====
179 - editArticle.ejs CSS style
180 ===== */
181
182 .editArticle-field /*text fields & save button*/
183 {
184     position: relative;
185     width: 10%;
186     display: grid;
187     grid-gap: 5px; /*Vertical gap*/
188     text-align: left;
189     color: ■ #000;
190 }
191
192 /* =====
193 - login.ejs CSS style
194 - register.ejs CSS style
195 ===== */
196
197 .login-form
198 {
199     position: relative;
200     margin: 13% 30% 1% 41%; /* To adjust it in center */
201     width: 10%;
202     display: grid;
203     grid-gap: 10px; /*Vertical gap*/
204     text-align: center;
205     padding: 70px;
206     color: ■ #000;
207     border: 1px solid #4d388b; /*border size & color*/
208     border-radius: 8px; /*rounded corners*/
209 }
210
211 .login-form button, .register-form button
```

```
212 {
213   font-weight: bolder;
214   background-color: #3db8c0;
215   color: white
216   border: none;
217   cursor: pointer;
218   width: 80%; /*width size*/
219   font-size: 100%;
220   padding: 10px 10px; /*overall size*/
221   margin: 8px 0; /*a top & bottom margin of 8px, a left & right margin of 0 */
222   border: none; /*remove default border*/
223   border-radius: 10px; /*corner roundness*/
224 }
225
226 /* Styles for the submit button when mouse is dragged over it*/
227 .login-form button:hover, .register-form button:hover
228 {
229   background-color: rgb(1, 98, 113); /*buttons BG color*/
230 }
231
232 /* Styles for the submit button when clicked */
233 .login-form button:active, .register-form button:active
234 {
235   background-color: #ff9900; /*buttons BG color*/
236 }
237
238 h2 /*register title*/
239 {
240   color: #007e92;
241   font-size: 20px;
242   margin-top: 0;
243 }
244
245 .register-form
246 {
247   position: relative;
248   margin: 11% 30% 1% 43%; /* To adjust it in center */
249   width: 10%;
250   display: grid;
251   grid-gap: 10px; /*Vertical gap*/
252   text-align: center;
253   padding: 50px;
254   color: #000;
255   border: 1px solid #4d388b; /*border size & color*/
256   border-radius: 8px; /*rounded corners*/
257 }
258
259 #loginError-text, #registerError-text
260 {
261   width: 80%; /*border width*/
262   margin: 20px auto 0; /*a top margin of 20px, horizontally center it & has a bottom margin
of 0 */
263   font-weight: bolder; /*text weight & color*/
264   color: red
```

```
265     border: 1px solid #4d388b; /*border size & color*/
266     border-radius: 10px; /*rounded corners of border*/
267     padding: 10px 10px; /*overall space added for text*/
268 }
269
270 /* =====
271     - readerHome.ejs CSS style
272     ===== */
273
274 h3 /*titles of every published articles in article-item*/
275 {
276     font-size: 20px;
277     margin-bottom: 10px;
278 }
279
280 .article-item /*Display list of articles if published*/
281 {
282     display: inline-grid;
283     grid-template-rows: 1fr;
284     grid-auto-flow: column;
285     grid-gap: 10px;
286     position: relative;
287     margin: 13% auto auto 0.3%; /*13% sets the top margin, auto sets the right & bottom
margins & 0.3% sets the left margin.*/
288     border: 1px solid #4d388b; /*border size & color*/
289     border-radius: 10px; /*rounded corners of border*/
290     padding: 5px;
291 }
292
293 .no-article-item /*otherwise display a message saying No articles to display */
294 {
295     position: relative;
296     top: 100%; /* position 100% below its original position */
297     margin: 13% auto auto 3%; /*13% sets the top margin, auto sets the right & bottom margins,
& 3% sets the left margin*/
298 }
299
300 /* =====
301     - authorHome.ejs CSS style
302     - mainHome.ejs CSS style
303     ===== */
304
305 /*display welcome message in mainHome & article (published, draft) contents in authorHome in
columns*/
306 .main-blog-articles
307 {
308     display: grid;
309     grid-auto-flow: column;
310     margin-top: 200px;
311 }
312
313 /*Overall space for welcome message in mainHome & article (published, draft) contents in
authorHome */
314 .blog-articles
```

```
315 {
316     padding: 8px 32px 8px 32px; /*to adjust it in center*/
317
318 }
319
320 h4 /*title for welcome message in mainHome & article (published, draft) in authorHome */
321 {
322     font-size: 32px;
323     text-align: center;
324     font-weight: bolder;
325     color: rgb(127, 85, 243);
326     margin: 0px;
327 }
328
329 .blog-articles p /*p below welcome message in mainHome &, in publish & draft article
330 contents in authorHome*/
331 {
332     font-weight: bolder;
333     font-size: 20px;
334     color: rgb(127, 85, 243);
335     margin: 0;
336     text-align: center;
337 }
338
339 /* =====
340 - articlePage.ejs CSS style
341 - editArticle.ejs CSS style
342 ===== */
343
344 .article-container /*1 container in articlePage & other in editArticle*/
345 {
346     margin-bottom: 40px;
347     padding: 20px;
348     border: 1px solid #4d388b;
349     border-radius: 8px;
350     width: auto;
351 }
352
353 h5 /* 'Edit Article' title displayed in editArticle & display titles set by authors in
354 articlePage */
355 {
356     color: ☐ white
357     font-size: 30px;
358     font-weight: bolder;
359     text-align: center;
360     background-color: rgb(0,126,146);
361     padding: 5px;
362     margin: -20px -20px 15px;
363     border-radius: 8px 8px 0 0;
364 }
365
366 /* =====
367 - articlePage.ejs CSS style
368 - editArticle.ejs CSS style
```



```
367 | - settingPage.ejs CSS style
368 | ===== */
369 |
370 | .blog-section /*main container for blog article read, edit & settings*/
371 | {
372 |     position: relative;
373 |     margin: 15% 12% 0 12%;
374 | }
375 | // End
```

public\confirm.js

```
1 // Start
2 /*
3     Content-Security-Policy: The page's settings blocked an event handler (script-src-attr)
4     from being executed because it violates the following directive: "script-src-attr 'none'"
5     Source: confirmAction(event, 'Are you sure you w...
6
7     To avoid the above error messages, I will implement event listeners using JavaScript
8     instead of
9     using inline event handlers (such as onclick in the HTML)
10 */
11 // Wait for the DOM content to be fully loaded before executing the script
12 document.addEventListener('DOMContentLoaded', function ()
13 {
14     // Select all elements with class 'publish-button' and store them in publishButtons
15     array
16     const publishButtons = document.querySelectorAll('.publish-button');
17
18     // Select all elements with class 'delete-draft-button' and store them in
19     deleteDraftButtons array
20     const deleteDraftButtons = document.querySelectorAll('.delete-draft-button');
21
22     // Select all elements with class 'delete-published-button' and store them in
23     deletePublishedButtons array
24     const deletePublishedButtons = document.querySelectorAll('.delete-published-button');
25
26     // Get the confirmation modal element by its ID
27     const confirmationModal = document.getElementById('confirmationModal');
28
29     // Get the message element inside the confirmation modal
30     const confirmationMessage = document.getElementById('confirmationMessage');
31
32     // Get the 'Yes' button inside the confirmation modal
33     const confirmYes = document.getElementById('confirmYes');
34
35     // Get the 'No' button inside the confirmation modal
36     const confirmNo = document.getElementById('confirmNo');
37
38     // Function to show the confirmation modal with a specific message and a callback
39     function
40     function showConfirmationModal(message, callback)
41     {
42         // Set the text content of the confirmation message
43         confirmationMessage.textContent = message;
44
45         // Display the confirmation modal (make it visible)
46         confirmationModal.style.display = 'block';
47
48         // Event listener for the 'Yes' button click
49         confirmYes.onclick = function ()
50         {
51             // Invoke the callback with true when 'Yes' is clicked
```

```
47     callback(true);
48     // Hide the confirmation modal
49     confirmationModal.style.display = 'none';
50 };
51
52 // Event listener for the 'No' button click
53 confirmNo.onclick = function ()
54 {
55     // Invoke the callback with false when 'No' is clicked
56     callback(false);
57     // Hide the confirmation modal
58     confirmationModal.style.display = 'none';
59 };
60 }
61
62 // Loop through each publish button in the publishButtons array
63 publishButtons.forEach(button => {
64     // Add click event listener to each publish button
65     button.addEventListener('click', function (event)
66     {
67         // Prevent default form submission behavior
68         event.preventDefault();
69
70         // Get the form ID from the dataset attribute of the clicked button
71         const formId = event.target.dataset.formId;
72
73         // Display confirmation modal with message and define callback function
74         showConfirmationModal('Are you sure you want to publish this draft?', function
(confirmed)
75         {
76             // If user confirms (callback returns true), submit the form with the
corresponding ID
77             if (confirmed)
78             {
79                 document.getElementById(formId).submit();
80             }
81             });
82         });
83     });
84
85 // Similar logic for delete draft buttons
86 deleteDraftButtons.forEach(button => {
87     button.addEventListener('click', function (event)
88     {
89         event.preventDefault();
90         const formId = event.target.dataset.formId;
91         showConfirmationModal('Are you sure you want to delete this draft?', function
(confirmed)
92         {
93             if (confirmed)
94             {
95                 document.getElementById(formId).submit();
96             }
97             });
98     });
99 });
```

```
98     });
99   });
100
101   // Similar logic for delete published buttons
102   deletePublishedButtons.forEach(button => {
103     button.addEventListener('click', function (event)
104     {
105       event.preventDefault();
106       const formId = event.target.dataset.formId;
107       showConfirmationModal('Are you sure you want to delete this article?', function
(confirm)
108       {
109         if (confirm)
110         {
111           document.getElementById(formId).submit();
112         }
113       });
114     });
115   });
116 });
117 // End
```

middleware\setUser.js

```
1 // Start
2 /**
3  * Middleware to set the user object in the response locals.
4  * If the user is authenticated, it sets the user object in the response locals.
5  * If the user is not authenticated, it sets the user object in the response locals to null.
6  * If there is a publishMessage in the session, it sets the publishMessage in the response
  locals and clears it from the session.
7  *
8  * @function
9  * @param {Object} req - Express request object
10 * @param {Object} res - Express response object
11 * @param {function} next - Express next middleware function
12 * @property {boolean} req.session.isAuthenticated - Indicates if the user is authenticated
13 * @property {Object} req.session.user - The authenticated user's session data
14 * @property {string} req.session.publishMessage - The message to be published
15 * @property {Object} res.locals.user - The authenticated user's local data
16 * @property {string} res.locals.publishMessage - The message to be published
17 */
18 module.exports = function(req, res, next)
19 {
20     if (req.session.isAuthenticated)
21     {
22         // Purpose: To set the user's session data in the response locals
23         // Inputs: User's session data
24         // Outputs: User's local data
25         res.locals.user = req.session.user || { user_name: 'Guest' };
26     }
27     else
28     {
29         // Purpose: To clear the user's local data if not authenticated
30         // Inputs: None
31         // Outputs: User's local data set to null
32         res.locals.user = null;
33     }
34     if (req.session.publishMessage) {
35         // Purpose: To set the publish message in the response locals and clear it from the
  session
36         // Inputs: Publish message from the session
37         // Outputs: Publish message in the response locals
38         res.locals.publishMessage = req.session.publishMessage;
39         req.session.publishMessage = null;
40     }
41     next();
42 };
43 // End
```

middleware\auth.js

```
1 // Start
2 /**
3  * Middleware to check if the user is authenticated.
4  * If the user is authenticated, it sets the user object in the response locals and calls the
  next middleware.
5  * If the user is not authenticated, it redirects the user to the login page.
6  *
7  * @function
8  * @param {Object} req - Express request object
9  * @param {Object} res - Express response object
10 * @param {function} next - Express next middleware function
11 * @property {boolean} req.session.isAuthenticated - Indicates if the user is authenticated
12 * @property {Object} req.session.user - The authenticated user's session data
13 * @property {Object} res.locals.user - The authenticated user's local data
14 */
15 module.exports = function(req, res, next)
16 {
17     if (req.session.isAuthenticated)
18     {
19         // Purpose: To set the user's session data in the response locals
20         // Inputs: User's session data
21         // Outputs: User's local data
22         res.locals.user = req.session.user;
23         next();
24     }
25     else
26     {
27         // Purpose: To redirect unauthenticated users to the login page
28         // Inputs: None
29         // Outputs: Redirection to '/auth/login'
30         res.redirect('/auth/login');
31     }
32 };
33 // End
```