Main.css

```css
@import
url('https://fonts.googleapis.com/css2?family=Montserrat:wght@100;300;500;700&dis
play=swap');

body {
    font-family: 'Montserrat', sans-serif !important;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    padding: 0;
    background-color: #f8f9fa;
    color: #ffffffd7 !important;
    min-height: 100vh;
    background-image: linear-gradient( 111.4deg,  rgba(7,7,9,1) 6.5%,
rgba(27,24,113,1) 93.2% );
}

.underline {
    font-weight: 300;
}

.edit-article {
    min-width: 40vw;
}

hr {
    background-color: white;
    width: 90vw;
}

.card-header {
    background-color: #ffffff17 !important;
    color: white !important;
}

.card {
    width: 300px;
    margin: 10px;
    background-color: #00000022 !important;
    color: rgba(255, 255, 255, 0.598) !important;
}
```

```css
.card-container {
    display: flex;
    max-width: 90vw;
    flex-wrap: wrap;
}

.card p {
    font-size: 10px;
    padding-left: 20px;
}

.underline:hover {
    border-bottom: 2px solid rgb(255, 255, 255);
}

p, label {
    letter-spacing: 1px;
}

form {
    display: flex;
    flex-direction: column;
}

.container {
    max-width: 800px;
    width: 100%;
    padding: 20px;
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1, h2, h3 {
    color: white !important;
}

ul {
    list-style: none;
    padding: 0;
}

li {
    margin-bottom: 10px;
}
```

```
hr {
    border: 0;
    height: 1px;
    background-image: linear-gradient(to right, rgba(0, 0, 0, 0), rgba(0, 0, 0,
0.75), rgba(0, 0, 0, 0));
}

.comment {
    display: flex;
    align-items: center;
    width: 80vw;
    border-radius: 10px;
    background-color: #00000022;
}

.comment span {
    font-size: 10px;
}

.comment > div {
    border-radius: 5px;
    background-color: #0000004f;
    width: 10vw;
    text-align: center;
}

.comment > p {
    padding-left: 10px;
}
```

Authors.js

```
// authors.js

const express = require("express");
const router = express.Router();

/**
 * @desc Displays a page with a form for creating an author record
 */
```

```
router.get("/add-author", (req, res) => {
    res.render("add-author.ejs");
});

/**
 * @desc Add a new author to the database based on data from the submitted form
 */
router.post("/add-author", (req, res, next) => {
    // Define the query
    authorName = req.body.author_name;
    const query = "INSERT INTO authors (author_name, blog_title) VALUES (?, ?);";
    const queryParameters = [req.body.author_name, req.body.blog_title];

    // Validate that the author name is not empty
    if (!authorName) {
        return res.status(400).send("Author name is required.");
    }

    // Execute the query and send a confirmation message
    global.db.run(query, queryParameters,
        function (err) {
            if (err) {
                next(err); // send the error on to the error handler
            } else {
                res.redirect(`/authors/home/${this.lastID}`);
            }
        }
    );
});

/**
 * @desc Display the Author's Home Page
 */
router.get("/home/:authorId", (req, res, next) => {
    const authorId = req.params.authorId;

    const authorQuery = "SELECT * FROM authors WHERE author_id = ?";
    const publishedArticlesQuery = "SELECT * FROM articles WHERE author_id = ?
AND status = 'published'";
    const draftArticlesQuery = "SELECT * FROM articles WHERE author_id = ? AND
status = 'draft'";

    global.db.get(authorQuery, [authorId], function (err, authorData) {
        if (err) {
            next(err);
```

```javascript
        } else {
            global.db.all(publishedArticlesQuery, [authorId], function (err, publishedArticles) {
                if (err) {
                    next(err);
                } else {
                    global.db.all(draftArticlesQuery, [authorId], function (err, draftArticles) {
                        if (err) {
                            next(err);
                        } else {
                            const data = {
                                blogTitle: authorData.blog_title,
                                authorName: authorData.author_name,
                                articles: [], // You may keep this if needed for other use

                                authorId,
                                publishedArticles,
                                draftArticles,
                            };

                            res.render('author-home.ejs', data);
                        }
                    });
                }
            });
        }
    });
});

/**
 * @desc Display the Author's Settings Page
 */
router.get("/settings/:authorId", (req, res, next) => {
    const authorId = req.params.authorId;

    const query = "SELECT * FROM authors WHERE author_id = ?";

    global.db.get(query, [authorId], function (err, authorData) {
        if (err) {
            next(err);
        } else {
            const data = {
                authorId,
                blogTitle: authorData.blog_title,
                authorName: authorData.author_name,
            };

            res.render('author-settings.ejs', data);
        }
    });
});

/**
 * @desc Update author settings with new values and redirect to Author Home Page
 */
router.post("/update-settings/:authorId", (req, res, next) => {
    const authorId = req.params.authorId;

    // Validate form data
    const { blog_title, author_name } = req.body;
    if (!blog_title || !author_name) {
        // Handle validation error (you can customize this part based on your
needs)
        res.send("Blog title and author name are required.");
        return;
    }

    // Logic for updating author settings in the database
    const updateQuery = "UPDATE authors SET blog_title = ?, author_name = ? WHERE
author_id = ?";
    const updateParams = [blog_title, author_name, authorId];

    global.db.run(updateQuery, updateParams, function (err) {
        if (err) {
            next(err); // send the error on to the error handler
        } else {
            // Redirect to the Author's Home Page after updating settings
            res.redirect(`/authors/home/${authorId}`);
        }
    });
});

/**
 * @desc Display the Edit Article Page
 */
router.get("/edit-draft/:draftId", (req, res, next) => {
    const draftId = req.params.draftId;

    const query = "SELECT * FROM articles WHERE article_id = ?";
    global.db.get(query, [draftId], function (err, article) {
        if (err) {
            next(err);
        } else {
            console.log("Article:", article); // Add this line for debugging

            const data = {
                draftId,
                authorId: article ? article.author_id : null,
                article,
            };

            res.render('author-edit-article.ejs', data);
        }
    });
});

/**
 * @desc Create a new draft article and redirect to its edit page
 */
router.post("/create-draft", (req, res, next) => {
    const authorId = req.body.authorId;
    console.log(authorId)
    const createDraftQuery = "INSERT INTO articles (author_id, title, content,
status, created_date, modified_date, views, likes) VALUES (?, ?, ?, 'draft',
datetime('now'), datetime('now'), 0, 0)";

    global.db.run(createDraftQuery, [authorId, '', ''], function (err) {
        if (err) {
            next(err);
        } else {
            const draftId = this.lastID;
            res.redirect(`/authors/edit-draft/${draftId}`);
        }
    });
});

/**
 * @desc Update the draft article with new data
 */
router.post("/update-draft/:draftId", (req, res, next) => {
    const draftId = req.params.draftId;
    const title = req.body.title;
    const content = req.body.content;
    const updateDraftQuery = "UPDATE articles SET title = ?, content = ?,
modified_date = datetime('now') WHERE article_id = ?";

    global.db.run(updateDraftQuery, [title, content, draftId], function (err) {
        if (err) {
            next(err);
        } else {
            // Redirect to the edit page again or any other page as needed
            res.redirect(`/authors/home/${req.body.authorId}`);
        }
    });
});

/**
 * @desc Delete a draft article
 */
router.post("/delete-draft/:articleId", (req, res, next) => {
    const articleId = req.params.articleId;
    const deleteDraftQuery = "DELETE FROM articles WHERE article_id = ? AND
status = 'draft'";

    global.db.run(deleteDraftQuery, [articleId], function (err) {
        if (err) {
            next(err);
        } else {
            res.redirect(`/authors/home/${req.body.authorId}`);
        }
    });
});

/**
 * @desc Publish a draft article
 */
router.post("/publish-draft/:articleId", (req, res, next) => {
    const articleId = req.params.articleId;
    const publishDraftQuery = "UPDATE articles SET status = 'published',
published_date = datetime('now') WHERE article_id = ? AND status = 'draft'";

    global.db.run(publishDraftQuery, [articleId], function (err) {
        if (err) {
            next(err);
        } else {
            res.redirect(`/authors/home/${req.body.authorId}`);
        }
    });
});

/**
 * @desc Delete a draft article
 */
router.post("/delete-article/:articleId", (req, res, next) => {
    const articleId = req.params.articleId;
    const deleteDraftQuery = "DELETE FROM articles WHERE article_id = ? AND 
status = 'published'";

    global.db.run(deleteDraftQuery, [articleId], function (err) {
        if (err) {
            next(err);
        } else {
            res.redirect(`/authors/home/${req.body.authorId}`);
        }
    });
});

module.exports = router;
```

main.js

```
const express = require("express");
const router = express.Router();


/**
 * @desc Default home page with links to Author Home and Reader Home
 */
router.get('/', (req, res) => {
    res.render('main.ejs');
});

/**
 * @desc Display all the authors
 */
router.get("/list-authors", (req, res, next) => {
    const query = "SELECT * FROM authors";
    const createUserQuery = "INSERT INTO users (user_name) VALUES (?)";
    global.db.run(createUserQuery, [''], function (err) {
        if (err) {
            next(err);
        } else {
            const userId = this.lastID;
            global.db.all(query, function (err, authors) {
                if (err) {
                    next(err);
                } else {
                    res.render("list-authors.ejs", { authors, userId});
                }
            });
        }
    });
});

/**
 * @desc Handle form submissions to create a new author
 */
router.post('/authors/home', (req, res, next) => {
    const createAuthorQuery = "INSERT INTO authors (author_name, blog_title)
VALUES (?, ?)";
    global.db.run(createAuthorQuery, ['', ''], function (err) {
        if (err) {
            next(err);
        } else {
            const authorId = this.lastID;
            res.redirect(`/authors/home/${authorId}`);
        }
    });
});

router.post('/users/home', (req, res, next) => {
    const createUserQuery = "INSERT INTO users (user_name) VALUES (?)";
    global.db.run(createUserQuery, [''], function (err) {
        if (err) {
            next(err);
        } else {
            const userId = this.lastID;
            const selectedAuthorId = req.body.authorId;
            res.redirect(`/users/home/${userId}/${selectedAuthorId}`);
        }
    });
```

```
});

module.exports = router;
```

users.js

```
/**
 * users.js
 * These are example routes for user management
 * This shows how to correctly structure your routes for the project
 * and the suggested pattern for retrieving data by executing queries
 *
 * NB. it's better NOT to use arrow functions for callbacks with the SQLite
library
*
 */
const express = require("express");
const router = express.Router();

/**
 * @desc Display all the users
 */
router.get("/list-users", (req, res, next) => {
    // Define the query
    query = "SELECT * FROM users"

    // Execute the query and render the page with the results
    global.db.all(query,
        function (err, rows) {
            if (err) {
                next(err); //send the error on to the error handler
            } else {
                res.json(rows); // render page as simple json
            }
        }
    );
});

/**
 * @desc Displays a page with a form for creating a user record
 */
router.get("/add-user", (req, res) => {
    res.render("add-user.ejs");
```

```
});

/**
 * @desc Add a new user to the database based on data from the submitted form
 */
router.post("/add-user", (req, res, next) => {
    // Define the query
    query = "INSERT INTO users (user_name) VALUES( ? );"
    query_parameters = [req.body.user_name]

    // Execute the query and send a confirmation message
    global.db.run(query, query_parameters,
        function (err) {
            if (err) {
                next(err); //send the error on to the error handler
            } else {
                res.send(`New data inserted @ id ${this.lastID}!`);
                next();
            }
        }
    );
});

/**
 * @desc Display the Reader's Home Page
 */
router.get("/home/:userId/:authorId", (req, res, next) => {
    const userId = req.params.userId;
    const authorId = req.params.authorId;
    const blogInfoQuery = "SELECT blog_title, author_name FROM authors WHERE
author_id = ?";
    const articlesQuery = "SELECT * FROM articles WHERE status = 'published' AND
author_id = ? ORDER BY published_date DESC";

    global.db.get(blogInfoQuery, [authorId], function (err, blogInfo) {
        if (err) {
            next(err);
        } else {
            global.db.all(articlesQuery, [authorId], function (err,
publishedArticles) {
                if (err) {
                    next(err);
                } else {
                    const data = {
                        blogTitle: blogInfo.blog_title,
                        authorName: blogInfo.author_name,
                        publishedArticles,
                        userId: userId,
                        authorId: authorId,
                    };

                    res.render('reader-home.ejs', data);
                }
            });
        }
    });
});

/**
 * @desc Display the Reader - Article Page
 */
router.get("/article/:userId/:authorId/:articleId", (req, res, next) => {
    const userId = req.params.userId;
    const authorId = req.params.authorId;
    const articleId = req.params.articleId;

    const articleQuery = "SELECT * FROM articles WHERE article_id = ?";
    const commentsQuery = "SELECT * FROM comments WHERE article_id = ? ORDER BY
created_date DESC";
    const updateViewsQuery = "UPDATE articles SET views = views + 1 WHERE
article_id = ?";

    global.db.get(articleQuery, [articleId], function (err, article) {
        if (err) {
            next(err);
        } else {
            global.db.all(commentsQuery, [articleId], function (err, comments) {
                if (err) {
                    next(err);
                } else {
                    global.db.run(updateViewsQuery, [articleId], function (err) {
                        if(err) {
                            next(err);
                        } else {
                            const data = {
                                userId,
                                authorId,
                                article,
                                comments,
                            };
                            res.render('reader-article.ejs', data);
                        }
                    });
                }
            });
        }
    });
});

/**
 * @desc Add a comment to an article
 */
router.post("/add-comment/:userId/:authorId/:articleId", (req, res, next) => {
    const userId = req.params.userId;
    const authorId = req.params.authorId;
    const articleId = req.params.articleId;
    const commenterName = req.body.commenterName;
    const commentText = req.body.commentText;

    const addCommentQuery = "INSERT INTO comments (article_id, user_id,
commenter_name, comment_text, created_date) VALUES (?, ?, ?, ?,
datetime('now'))";

    global.db.run(addCommentQuery, [articleId, userId, commenterName,
commentText], function (err) {
        if (err) {
            next(err);
        } else {
            // Redirect back to the article page
            res.redirect(`/users/article/${userId}/${authorId}/${articleId}`);
        }
    });
});

/**
 * @desc Like an article
 */
router.post("/like-article/:userId/:authorId/:articleId", (req, res, next) => {
    const userId = req.params.userId;
    const authorId = req.params.authorId;
    const articleId = req.params.articleId;

    // Check if the user has already liked the article
    const checkLikeQuery = "SELECT COUNT(*) as count FROM likes WHERE article_id
= ? AND user_id = ?";
```

```
    global.db.get(checkLikeQuery, [articleId, userId], (err, result) => {
        if (err) {
            next(err);
        } else {
            const alreadyLiked = result.count > 0;

            if (!alreadyLiked) {
                // Update the likes and record the user's like
                const updateLikesQuery = "UPDATE articles SET likes = likes + 1
WHERE article_id = ?";
                const recordLikeQuery = "INSERT INTO likes (article_id, user_id)
VALUES (?, ?)";
                console.log()

                global.db.serialize(() => {
                    global.db.run(updateLikesQuery, [articleId]);
                    global.db.run(recordLikeQuery, [articleId, userId]);

                    // Redirect back to the article page
                    res.redirect(`/users/article/${userId}/${authorId}/${articleI
d}`);
                });
            } else {
                // User has already liked the article, redirect back to the
article page
                res.redirect(`/users/article/${userId}/${authorId}/${articleId}`)
;
            }
        }
    });
});

// Export the router object so index.js can access it
module.exports = router;
```

add-author.ejs

```
<!DOCTYPE html>
<html lang="en">
```

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="/main.css" />
    <title>Add Author</title>
</head>
<body>
    <h1>Add Author</h1>
    <form action="/authors/add-author" method="post">
        <p>Author Name: <input id="authorName" type="text" name="author_name"
/></p>
        <p>Blog Title: <input id="blogTitle" type="text" name="blog_title" /></p>
        <button type="submit">Add Author</button>
    </form>
</body>
</html>
```

Add-user.ejs

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"  type="text/css" href="/main.css" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <title>Add User</title>
</head>
<body>
    <h1>Add User</h1>
    <form action="add-user" method="post">
        <p>User: <input id="user" type="text" name="user_name" class="form-
control"/></p>
        <button type="submit">Create a new user record</button>
    </form>
</body>
</html>
```

Author-edit-article.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="/main.css" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <title>Edit Article</title>
</head>
<body>
    <h1>Edit Article</h1>

    <form class="edit-article" action="/authors/update-draft/<%= draftId %>"
method="post">
        <label for="articleTitle">Article Title:</label>
        <input class="form-control mb-2" type="text" id="articleTitle"
name="title" value="<%= article.title %>" required>
        <input type="hidden" name="authorId" value="<%= authorId %>">
        <label for="articleText">Article Text:</label>
        <textarea class="form-control mb-2" id="articleText" name="content"
rows="4" required><%= article.content %></textarea>

        <button class="btn btn-outline-info" type="submit">Submit
Changes</button>
    </form>

    <a class="link-info link-offset-2 link-underline-opacity-25 link-underline-
opacity-100-hover mt-2" href="/authors/home/<%= authorId %>" id="backToHome">Back
to Author Home</a>
</body>
</html>
```

Author-settings.ejs

```html
<!-- author-settings.ejs -->

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="/main.css" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <title>Author Settings</title>
</head>
<body>
    <h1 class="my-5">Author Settings Page</h1>

    <form action="/authors/update-settings/<%= authorId %>" method="post">
        <label for="blogTitle">Blog Title:</label>
        <input class="form-control mb-3" type="text" id="blogTitle"
name="blog_title" value="<%= blogTitle %>" required>

        <label for="authorName">Author Name:</label>
        <input class="form-control mb-2" type="text" id="authorName"
name="author_name" value="<%= authorName %>" required>

        <button class="btn btn-outline-info my-3" type="submit">Update
Settings</button>
    </form>

    <a class="link-info link-offset-2 link-underline-opacity-25 link-underline-
opacity-100-hover" href="/authors/home/<%= authorId %>">Back to Author Home
Page</a>
</body>
</html>
```

List-authors.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"  type="text/css" href="/main.css" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <title>List Authors</title>
</head>
<body>
    <h1>List of Authors</h1>
    <ul>
        <% authors.forEach(author => { %>
            <form action="/users/home/<%= userId%>/<%= author.author_id%>",
method="get" >
                <li>
                    <button class="btn mb-3 btn-outline-info" type="submit">
                        <%= author.author_name %>
                    </button>
                </li>
            </form>
        <% }); %>
    </ul>
</body>
</html>
```

List-users.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="/main.css" />
    <title>List Users</title>
```

```
</head>
<body>
    <h1>List Users</h1>
    <ul>
        <% users.forEach(user => { %>
            <li><%= user.user_name %></li>
        <% }); %>
    </ul>
</body>
</html>
```

Main.ejs

```
<!-- main.ejs -->

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="/main.css" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <title>Home Page</title>
</head>
<body>
    <h1>Welcome to the Application</h1>

    <p>Choose your role:</p>

    <!-- Form to create a new author -->
    <form action="/authors/home" method="post">
        <input type="hidden" name="authorName" value="">
        <input type="hidden" name="blogTitle" value="">
        <button class="btn mb-3 btn-secondary" type="submit">Author Home</button>
    </form>

    <!-- Form to create a new user -->
```

```html
    <form action="/list-authors" method="get">
        <!-- <input type="hidden" name="userName" value=""> -->
        <button class="btn btn-secondary" type="submit">Reader Home</button>
    </form>
</body>
</html>
```

Reader-article.ejs

```html
<!-- reader-article.ejs -->

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="/main.css" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <title>Reader Article Page</title>
</head>
<body>
    <h1 class="my-5">Reader Article Page</h1>

    <!-- Display article information -->
    <h2 class="d-flex"><%= article.title %><form action="/users/like-article/<%=
userId %>/<%= authorId %>/<%= article.article_id %>" method="post">
        <input type="hidden" name="userId" value="<%= userId %>">
        <button class="btn btn-danger" type="submit">Like</button>
    </form></h2>
    <p>Published: <%= article.published_date %>, Views: <%= article.views %>,
Likes: <%= article.likes %></p>
    <p class="w-75"><%= article.content %></p>

    <!-- Like button -->


    <!-- Add Comment Form -->
    <form action="/users/add-comment/<%= userId %>/<%= authorId %>/<%=
article.article_id %>" method="post">
```

```
        <label for="commenterName">Your Name:</label>
        <input class="form-control" type="text" id="commenterName"
name="commenterName" required>
        <input type="hidden" name="userId" value="<%= userId %>">
        <label for="commentText">Your Comment:</label>
        <textarea class="form-control" id="commentText" name="commentText"
rows="4" required></textarea>

        <button class="btn btn-success mt-2" type="submit">Submit
Comment</button>
    </form>

    <hr class="my-5">
    <!-- List of Previous Comments -->
    <h3 class="mb-3">Previous Comments</h3>
    <ul>
        <% comments.forEach(comment => { %>
            <li class="comment">
                <div>
                    <p><%= comment.commenter_name %></p>
                    <span><%= comment.created_date %></span>
                </div>
                <p><%= comment.comment_text %></p>
            </li>
        <% }); %>
    </ul>

    <!-- Back button -->
    <a class="link-info link-offset-2 link-underline-opacity-25 link-underline-
opacity-100-hover mb-3" href="/users/home/<%= userId %>/<%= authorId %>">Back to
Reader Home</a>
</body>
</html>
```

Reader-home.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="/main.css" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    <title>Reader Home</title>
</head>
<body>
    <h1 class="mb-5">Reader Home Page</h1>
    <p><strong>Blog Title:</strong> <span class="underline"><%= blogTitle
%></span></p>
    <p><strong>Author:</strong> <span class="underline"><%= authorName
%></span></p>
    <hr>
    <h2>Published Articles</h2>
    <ul class="card-container">
        <% publishedArticles.forEach(article => { %>
            <li class="card">
                <h4 class="card-header mb-1"><%= article.title %></h4>
                <p>Published: <%= article.published_date %></p>
                <a class="btn-secondary btn" href="/users/article/<%= userId
%>/<%= authorId %>/<%= article.article_id %>">Read Article</a>
            </li>
        <% }); %>
    </ul>
</body>
</html>
```

Index.js

```
/**
* index.js
* This is your main app entry point
*/

// Set up express, bodyparser and EJS
const express = require('express');
const app = express();
const port = 3000;
var bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: true }));
app.set('view engine', 'ejs'); // set the app to use ejs for rendering
app.use(express.static(__dirname + '/public')); // set location of static files

// Set up SQLite
// Items in the global namespace are accessible throught out the node application
const sqlite3 = require('sqlite3').verbose();
global.db = new sqlite3.Database('./database.db', function(err){
    if(err){
        console.error(err);
        process.exit(1); // bail out we can't connect to the DB
    } else {
        console.log("Database connected");
        global.db.run("PRAGMA foreign_keys=ON"); // tell SQLite to pay attention
to foreign key constraints
    }
});

// // Handle requests to the home page
// app.get('/', (req, res) => {
//     res.send('Hello World!')
// });

const mainRoute = require('./routes/main');
app.use('/', mainRoute);

// Add all the route handlers in usersRoutes to the app under the path /users
const usersRoutes = require('./routes/users');
app.use('/users', usersRoutes);

// Add routes for authors
const authorsRoutes = require('./routes/authors');
app.use('/authors', authorsRoutes);

// Make the web application listen for HTTP requests
app.listen(port, () => {
    console.log(`App listening on port ${port}`)
})
```

Db_schema.sql

```sql
-- This makes sure that foreign_key constraints are observed and that errors will
be thrown for violations
PRAGMA foreign_keys=ON;

BEGIN TRANSACTION;

-- Create your tables with SQL commands here (watch out for slight syntactical
differences with SQLite vs MySQL)

CREATE TABLE IF NOT EXISTS users (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_name TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS authors (
    author_id INTEGER PRIMARY KEY AUTOINCREMENT,
    author_name TEXT NOT NULL,
    blog_title TEXT
);

CREATE TABLE IF NOT EXISTS articles (
    article_id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    content TEXT,
    created_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    published_date DATETIME,
    modified_date DATETIME,
    likes INTEGER,
    views INTEGER,
    status TEXT,
    author_id INTEGER,
    FOREIGN KEY (author_id) REFERENCES authors(author_id)
);

CREATE TABLE IF NOT EXISTS comments (
    comment_id INTEGER PRIMARY KEY AUTOINCREMENT,
    article_id INTEGER,
    commenter_name TEXT,
    comment_text TEXT,
    created_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    user_id INTEGER,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
    FOREIGN KEY (article_id) REFERENCES articles(article_id)
);

CREATE TABLE IF NOT EXISTS likes (
    like_id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    article_id INTEGER,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (article_id) REFERENCES articles(article_id)
);

CREATE TABLE IF NOT EXISTS email_accounts (
    email_account_id INTEGER PRIMARY KEY AUTOINCREMENT,
    email_address TEXT NOT NULL,
    user_id  INT, --the user that the email account belongs to
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);

-- Set up three users
INSERT INTO users ('user_name') VALUES ('Simon Star');
INSERT INTO users ('user_name') VALUES ('Dianne Dean');
INSERT INTO users ('user_name') VALUES ('Harry Hilbert');

INSERT INTO email_accounts ('email_address', 'user_id') VALUES
('simon@gmail.com', 1);
INSERT INTO email_accounts ('email_address', 'user_id') VALUES
('simon@hotmail.com', 1);
INSERT INTO email_accounts ('email_address', 'user_id') VALUES
('dianne@yahoo.co.uk', 2);

COMMIT;
```