This is the Code for The Databse Newtworks and the Web Midterm

database file

database.db

```
-- 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,
    blog_title TEXT NOT NULL
);

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)
);

CREATE TABLE IF NOT EXISTS articles (
    article_id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    author_id INTEGER NOT NULL,
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_modified TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    published_at TIMESTAMP,
    is_published BOOLEAN NOT NULL DEFAULT FALSE, --Come back to check if not null
is needed becuase it defualts to null
    num_reads INTEGER DEFAULT 0,
    num_likes INTEGER DEFAULT 0,
    FOREIGN KEY (author_id) REFERENCES users(user_id)
);

-- Insert default data (if necessary here)

-- Set up users
INSERT INTO users ('user_name', 'blog_title') VALUES ('Test User Name', 'Test Blog
Title');
```

```
INSERT INTO users ('user_name', 'blog_title') VALUES ('Test User Name2', 'Test
Blog Title2');

-- Set up Articles
    -- Published articles tests
INSERT INTO articles
('title','author_id','content','created_at','last_modified','published_at',
'is_published') VALUES ('Published Article',1,'This is a published article.',
CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 1);

INSERT INTO articles
('title','author_id','content','created_at','last_modified','published_at',
'is_published') VALUES ('Published Article2',1,'This is another published
article.', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 1);

    -- Unpublished article tests
INSERT INTO articles
('title','author_id','content','created_at','last_modified','published_at',
'is_published') VALUES ('Unpublished Article',1,'This is a unpublished article.',
CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 0);

INSERT INTO articles
('title','author_id','content','created_at','last_modified','published_at',
'is_published') VALUES ('Unpublished Article2',1,'This is another unpublished
article.', CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP, 0);



COMMIT;
```

Views folder

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" />
    <title>Reader Home Page</title>
</head>
<body>
    <header>
        <div class="header-title">
            <h3>MicroBlogger:</h3>
            <b>Reader Home Page</b>
        </div>
        <nav>
            <a href="/"><button>Go Back to Main Home Page</button></a>
```

```
        </nav>
    </header>
    <div class="blog-info">
        <h1><%= blogTitle %></h1>
        <h2><%= blogAuthor %></h2>
    </div>
    <div class="published-articles">
        <table>
            <caption><h2>Published Articles</h2></caption>
            <thead>
                <tr>
                    <th>Title</th>
                    <th>Created</th>
                    <th>Last Modified</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                <% publishedArticles.forEach(function(article){ %>
                <tr>
                    <td><%= article.title %></td>
                    <td><%= article.created_at %></td>
                    <td><%= article.last_modified %></td>
                    <td>
                        <div class="actionButtons">
                            <button>READ</button>
                        </div>
                    </td>
                </tr>
                <% }) %>
            </tbody>
        </table>
    </div>
</body>
</html>
```

main-homepage.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>Main Home Page</title>
</head>
<body>
    <h1>Welcome to MicroBlogger!</h1>
    <a href="/author/home"><button>Go to Author Home Page</button></a>
    <a href="/reader/home"><button>Go to Reader Home Page</button></a>
```

```
</body>
</html>
```

author-setting.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>Author Settings Page</title>
</head>
<body>
    <header>
        <div class="header-title">
            <h3>MicroBlogger:</h3>
            <b>Author Home Page</b>
        </div>
        <nav>
            <a href="/author/home"><button>Back to Home Page</button></a>
        </nav>
    </header>

    <div class="page-info">
        <h1>Change User Settings</h1>
    </div>

    <div class="settings">
        <form action="update-user" method="post">
            <p>User: <input id="user" type="text" name="user_name" /></p>
            <p>Title: <input id="title" type="text" name="blog_title" /></p>
            <button type="submit">Save Changes</button>
        </form>
    </div>
</body>
</html>
```

author-new.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>Author New Draft</title>
</head>
<body>
```

```
    <header>
        <div class="header-title">
            <h3>MicroBlogger:</h3>
            <b>New Draft</b>
        </div>
        <nav>
            <a href="/author/home"><button>Back to Home Page</button></a>
        </nav>
    </header>

    <div class="page-info">
        <h1>Create New Draft</h1>
    </div>

    <div class="editor">
        <form method="POST" action="/author/create-draft">

            <label for="title">Title:</label>
            <input id="title" type="text" name="title" placeholder="Title
Example"/> <br>

            <label for="content">Start Writing:</label>
            <textarea id="content" name="content" rows="10" cols="50"
placeholder="&quot;No matter how great a writer, artist, or entrepreneur, he is a
mortal, he is fallible. He is not proof against Resistance. He will drop the ball;
he will crash. That&apos;s why they call it rewriting&quot;Steven Pressfield, Do
The Work"></textarea> <br>

            <button type="submit">Save Draft</button>
        </form>
    </div>
</body>
</html>
```

author-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" />
    <title>Author Home Page</title>
</head>
<body>
    <header>
        <div class="header-title">
            <h3>MicroBlogger:</h3>
            <b>Author Home Page</b>
        </div>
        <nav>
```

```
            <a href="/author/settings"><button>Settings</button></a>
            <a href="/"><button>Go Back to Main Home Page</button></a>
        </nav>
    </header>

    <div class="blog-info">
        <h1><%= blogTitle %></h1>
        <h2><%= blogAuthor %></h2>
    </div>

    <div class="articles">

        <div class="draft-articles">
            <table>
                <caption><h2>Draft Articles</h2></caption>
                <thead>
                    <tr>
                        <th>Title</th>
                        <th>Created</th>
                        <th>Last Modified</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
                    <% draftArticles.forEach(function(article){ %>
                    <tr>
                        <td><%= article.title %></td>
                        <td><%= article.created_at %></td>
                        <td><%= article.last_modified %></td>
                        <td>
                            <div class="actionButtons">
                                <a href="/author/edit-article/<%=
article.article_id %>">
                                    <button type="button">EDIT</button>
                                </a>

                                <form   action="/author/publish-article"
method="post">
                                    <input type="hidden" name="article_id" value="
<%= article.article_id %>" />
                                    <button type="submit">PUBLISH</button>
                                </form>

                                <form   action="/author/delete-article"
method="post">
                                    <input type="hidden"  name="article_id"
value="<%= article.article_id %>" />
                                    <button type="submit">DELETE</button>
                                </form>
                            </div>
                        </td>
                    </tr>
                    <% }) %>
                </tbody>
```

```
            </table>
            <a href="/author/new-draft"><button>NEW DRAF</button></a>
        </div>

        <div class="published-articles">
            <table>
                <caption><h2>Published Articles</h2></caption>
                <thead>
                    <tr>
                        <th>Title</th>
                        <th>Created</th>
                        <th>Last Modified</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
                    <% publishedArticles.forEach(function(article){ %>
                    <tr>
                        <td><%= article.title %></td>
                        <td><%= article.created_at %></td>
                        <td><%= article.last_modified %></td>
                        <td>
                            <div class="actionButtons">
                                <button>READ</button>

                                <a href="/author/edit-article/<%=
article.article_id %>">
                                    <button type="button">EDIT</button>
                                </a>

                                <form action="/author/delete-article"
method="post">
                                    <input type="hidden" name="article_id" value="
<%= article.article_id %>" />
                                    <button type="submit">DELETE</button>
                                </form>
                            </div>
                        </td>
                    </tr>
                    <% }) %>
                </tbody>
            </table>
        </div>
    </div>
</body>
</html>
```

author-edit.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>Author Edit Page</title>
</head>
<body>

    <header>
        <div class="header-title">
            <h3>MicroBlogger:</h3>
            <b>Author Home Page</b>
        </div>
        <nav>
            <a href="/author/home"><button>Back to Home Page</button></a>
        </nav>
    </header>


    <div class="page-info">
        <h1>Edit Article</h1>
    </div>

    <div class="article-info">
        <p>Date Created: <%= article.created_at %></p>
        <p>Date Modified: <%= article.last_modified %></p>
    </div>

    <div class="editor">
        <form method="POST" action="/author/update-article">
            <input type="hidden" name="articleId" value="<%= article.article_id
%>">
            <p>Title: <input type="text" name="title" value="<%= article.title
%>"></p>
            <p>Content: <textarea name="content"><%= article.content %></textarea>
</p>
            <button type="submit">Save Changes</button>
        </form>
    </div>

</body>
</html>
```

Routes file

Reader.js

```
/**
 * reader.js
*
 */
```

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

/**
 * @desc Displays the reader home page
 */
router.get("/home", (res) => {
    let userId = 1; // Example id need to take the logic from getting the article
to getting a user

    let userQuery = "SELECT user_name, blog_title FROM users WHERE user_id = ?";
    let articlesQuery = "SELECT * FROM articles WHERE author_id = ?";

    global.db.get(userQuery, [userId], (err, row) => {
        if (err) {
            res.redirect("/");
        } else {
            global.db.all(articlesQuery, [userId], (err, articles) => {
                if (err) {
                    res.redirect("/");
                } else {
                    let publishedArticles = articles.filter(article =>
article.is_published);

                    res.render("reader-home.ejs", {
                        blogTitle: row.blog_title,
                        blogAuthor: row.user_name,
                        publishedArticles: publishedArticles
                    });
                }
            });
        }
    });
});

/**
 * @desc Displays the article to read it
 */
router.get("/read", (res) => {
        res.render("reader-article.ejs");
});


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

main.js

```
/**
 * main.js
 * These are routes for main page 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 Displays the Main home page
 */
router.get("/", (req, res, next) => {
    res.render("main-homepage.ejs");
});


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

author.js

```
/**
 * author.js
 * These are routes for author 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 Displays the author home page with drafs and published articles and Gets
Blog Title and author name for the first User
 */
router.get("/home", (req, res) => {
    let userId = 1; // Example id need to take the logic from getting the article
to getting a user
```

```
    let userQuery = "SELECT user_name, blog_title FROM users WHERE user_id = ?";
    let articlesQuery = "SELECT * FROM articles WHERE author_id = ?";

    global.db.get(userQuery, [userId], (err, row) => {
        if (err) {
            res.redirect("/");
        } else {
            global.db.all(articlesQuery, [userId], (err, articles) => {
                if (err) {
                    res.redirect("/");
                } else {
                    let draftArticles = articles.filter(article =>
!article.is_published);
                    let publishedArticles = articles.filter(article =>
article.is_published);

                    res.render("author-home.ejs", {
                        blogTitle: row.blog_title,
                        blogAuthor: row.user_name,
                        draftArticles: draftArticles,
                        publishedArticles: publishedArticles
                    });
                }
            });
        }
    });
});

/**
 * @desc Publish button updates `is_published` to true/1
 */
router.post("/publish-article", (req, res) => {
    let articleId = req.body.article_id;
    let updateQuery = "UPDATE articles SET is_published = 1 WHERE article_id =
?;";

    global.db.run(updateQuery, [articleId], function (err) {
        if (err) {
            // Handle error appropriately
            return res.status(500).json({ error: "Database error: " + err.message
});
        } else {
            res.redirect("/author/home");  // Redirect back to the author home
page
        }
    });
});

/**
 * @desc Delete button removes article from database
 */
router.post("/delete-article", (req, res) => {
    let articleId = req.body.article_id;
```

```
    let deleteQuery = "DELETE FROM articles WHERE article_id = ?;";

    global.db.run(deleteQuery, [articleId], function (err) {
        if (err) {
            // Handle error appropriately
            return res.status(500).json({ error: "Database error: " + err.message
});
        } else {
            res.redirect("/author/home");  // Redirect back to the author home
page
        }
    });
});

/**
 * @desc Edit button gets the article id and uses passes it to author-edit to
build that page.
 */
router.get("/edit-article/:articleId", (req, res) => {
    let articleId = req.params.articleId;
    let articleQuery = "SELECT * FROM articles WHERE article_id = ?";

    global.db.get(articleQuery, [articleId], function (err, row) {
        if (err) {
            return res.status(500).json({ error: "Database error: " + err.message
});
        } else {
            // Pass the article details to the EJS template
            res.render("author-edit.ejs", { article: row });
        }
    });
});

/**
 * @desc Updates the selected article
 */
router.post("/update-article", (req, res) => {
    let articleId = req.body.articleId;
    let title = req.body.title;
    let content = req.body.content;

    let updateArticleQuery = "UPDATE articles SET title = ?, content = ? WHERE
article_id = ?";
    global.db.run(updateArticleQuery, [title, content, articleId], function (err)
{
        if (err) {
            return res.status(500).json({ error: "Database error: " + err.message
});
        } else {
            res.redirect("/author/home");
        }
    }); // need to update last_modified after updating article
});
```

```
/**
 * @desc Displays a page containing the authors settings to change the users name
and blog title
 */
router.get("/settings", (req,res) => {
    res.render("author-settings.ejs");
});

/**
 * @desc Updates the Authors name and blog title
 */
router.post("/update-user", (req, res) => {
    let userId = 1; // Example id need to take the logic from getting the article
to getting a user

    // Check if user_name or blog_title are empty strings
    if (req.body.user_name === "" || req.body.blog_title === "") {
        return res.status(400).json({ error: "Username and blog title must not be
empty" });
    }

    let updateQuery = "UPDATE users SET user_name = ?, blog_title = ? WHERE
user_id = ?;";
    let queryParams = [req.body.user_name, req.body.blog_title, userId];

    global.db.run(updateQuery, queryParams, function (err) {
        if (err) {
            // Handle the error appropriately
            return res.status(500).json({ error: "Database error: " + err.message
});
        } else {
            res.redirect("/author/home"); // Redirect to the homepage or show a
success message
        }
    });
});

/**
 * @desc Displays a page containing the authors settings to change the users name
and blog title
 */
router.get("/new-draft", (req, res) => {
    res.render("author-new", { article: {} });
});


/**
 * @desc Save the new draft
 */
router.post("/create-draft", (req, res) => {
    let userId = 1; // Example id need to take the logic from getting the article
to getting a user
```

```
    let insertQuery = "INSERT INTO articles (title, author_id, content) VALUES (?,
?, ?);";
    let queryParams = [req.body.title, userId, req.body.content];

    global.db.run(insertQuery, queryParams, function (err) {
        if (err) {
            // Handle the error appropriately
            return res.status(500).json({ error: "Database error: " + err.message
});
        } else {
            res.redirect("/author/home"); // Redirect to the homepage or show a
success message
        }
    });
});

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

public file

main.css

```
/* ---Define global color variables--- */
:root {

    --primaryColour: #043d7a;
    --secondaryColour: #343a40;
    --accentColour1: #ced4da;
    --accentColour2: #007bff;
    --accentColour3: #6c757d;
    --accentColour4: #adb5bd;
    --contrastColour1: #dc3545;

    /*
    --primaryColour: #007bff;
    --secondaryColour: #333333;
    --accentColour1: #7bb2e9;
    --accentColour2: #546e88;
    --accentColour3: #fcfcfc;
    --accentColour4: #ffc107;
    --contrastColour1: #28a745;
    */

}

/* ---Define Basic Styles--- */
body {
    background: var(--accentColour1);
}
```

```
h1 {
    color: var(--primaryColour);
}

h2 {
    color: var(--secondaryColour);
}

h3 {
    color: var(--contrastColour1);
}

button {
    display: inline-block;
    padding: 10px;
    font-weight: bold;
    text-align: center;
    background-color: var(--primaryColour);
    color: var(--accentColour4);
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: var(--accentColour4);
    color: var(--primaryColour);
}

textarea {
    font-size: 16px;
    font-family: Arial, sans-serif;
    padding: 10px;
}
/* ---Header--- */
header {

    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: var(--accentColour3);
    padding: 20px;
}

.header-title h3, .header-title b {
    display: inline;
}

nav button {
    margin-left: 10px;
    background-color: var(--primaryColour);
    color: var(--accentColour4);
    padding: 10px;
    border: none;
```

```
    cursor: pointer;
}

nav button:hover {
    background-color: var(--accentColour4);
    color: var(--primaryColour);
}

/* ---Tables--- */
table {
    width: 50%;
    border: 2px solid var(--secondaryColour);
}

table th {
    background-color: var(--primaryColour);
    color: var(--accentColour4);
    padding: 10px;
}

table td {
    padding: 10px;
}

/* Alternating row colours*/
table tr:nth-child(even) {
    background-color: var(--accentColour1);
}

table tr:nth-child(odd) {
    background-color: var(--accentColour2);
}

/* The Action buttons in the tables */
.actionButtons {
    display: flex;
    gap: 10px;
}

.actionButtons button {
    background-color: var(--primaryColour);
    color: var(--accentColour3);
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
}

/* Hovering in table */
table tr:hover {
    background-color: var(--accentColour3);
    color: var(--secondaryColour);
}
```

```
.actionButtons button:hover {
    background-color: var(--accentColour4);
    color: var(--primaryColour);
}
```