**Blogging Tool Project Report**

**Db/init.sql**

CREATE TABLE IF NOT EXISTS settings (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    blog\_title TEXT NOT NULL,

    author\_name TEXT NOT NULL

);

CREATE TABLE IF NOT EXISTS articles (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    title TEXT NOT NULL,

    content TEXT NOT NULL,

    created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

    updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

    published BOOLEAN DEFAULT FALSE,

    views INTEGER DEFAULT 0,

    likes INTEGER DEFAULT 0

);

CREATE TABLE IF NOT EXISTS comments (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    article\_id INTEGER NOT NULL,

    commenter\_name TEXT NOT NULL,

    comment TEXT NOT NULL,

    created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

    FOREIGN KEY(article\_id) REFERENCES articles(id)

);

INSERT INTO settings (blog\_title, author\_name) VALUES ('My Blog', 'Author Name');

**Middleware/auth.js**

module.exports = (req, res, next) => {

  if (req.session.isAuthenticated) {

    return next();

  } else {

    res.redirect('/auth');

  }

};

**Public/syles.css**

body {

  font-family: Arial, sans-serif;

  margin: 0;

  padding: 0;

  background-color: #f4f4f4;

}

header {

  background-color: #333;

  color: white;

  padding: 1rem;

  text-align: center;

}

nav a, nav form button {

  margin: 0 1rem;

  padding: 0.5rem 1rem;

  background-color: #008cba;

  color: white;

  text-decoration: none;

  border: none;

  border-radius: 5px;

}

nav a:hover, nav form button:hover {

  background-color: #005f5f;

}

main {

  padding: 1rem;

}

h2, h3 {

  color: #333;

}

ul {

  list-style: none;

  padding: 0;

}

ul li {

  background-color: white;

  margin: 0.5rem 0;

  padding: 1rem;

  border: 1px solid #ddd;

  border-radius: 5px;

}

form {

  margin: 1rem 0;

}

form label {

  display: block;

  margin-bottom: 0.5rem;

}

form input, form textarea, form button {

  display: block;

  width: 100%;

  padding: 0.5rem;

  margin-bottom: 1rem;

  border: 1px solid #ddd;

  border-radius: 5px;

}

form button {

  background-color: #008cba;

  color: white;

}

form button:hover {

  background-color: #005f5f;

}

**Routes/auth.js**

const express = require('express');

const router = express.Router();

router.get('/', (req, res) => {

  res.render('login');

});

router.post('/', (req, res) => {

  const { password } = req.body;

  if (password === process.env.AUTHOR\_PASSWORD) {

    req.session.isAuthenticated = true;

    res.redirect('/author');

  } else {

    res.status(401).send('Unauthorized');

  }

});

router.get('/logout', (req, res) => {

  req.session.destroy(err => {

    if (err) {

      return res.status(500).send('Failed to log out');

    }

    res.redirect('/auth'); // Update this path

  });

});

module.exports = router;

**Routes/author.js**

const express = require('express');

const router = express.Router();

const db = require('../db');

const authMiddleware = require('../middleware/auth');

router.use(authMiddleware);

router.get('/', async (req, res) => {

  const blogSettings = await db.getSettings();

  const publishedArticles = await db.getPublishedArticles();

  const draftArticles = await db.getDraftArticles();

  res.render('authorHome', {

    blog\_title: blogSettings.blog\_title,

    author\_name: blogSettings.author\_name,

    publishedArticles: publishedArticles,

    draftArticles: draftArticles

  });

});

// Settings Page

router.get('/settings', async (req, res) => {

  const blogSettings = await db.getSettings();

  res.render('settings', { settings: blogSettings });

});

router.post('/settings', async (req, res) => {

  const { blog\_title, author\_name } = req.body;

  await db.updateSettings(blog\_title, author\_name);

  res.redirect('/author');

});

// Edit Article Page

router.get('/edit/:id', async (req, res) => {

  try {

    const article = await db.getArticleById(req.params.id);

    res.render('editArticle', { article: article });

  } catch (error) {

    res.status(404).send('Article not found');

  }

});

router.post('/edit/:id', async (req, res) => {

  const { title, content } = req.body;

  await db.updateArticle(req.params.id, title, content);

  res.redirect('/author');

});

// Publish Article

router.post('/publish/:id', async (req, res) => {

  await db.publishArticle(req.params.id);

  res.redirect('/author');

});

// Delete Article

router.post('/delete/:id', async (req, res) => {

  await db.deleteArticle(req.params.id);

  res.redirect('/author');

});

// Create a new draft article

router.post('/create', async (req, res) => {

  const newArticleId = await db.createDraftArticle();

  res.redirect(`/author/edit/${newArticleId}`);

});

module.exports = router;

**Routes/index.js**

const express = require('express');

const router = express.Router();

router.get('/', (req, res) => {

  res.render('mainHome');

});

module.exports = router;

**Routes/reader.js**

const express = require('express');

const router = express.Router();

const db = require('../db');

// Reader Home Page

router.get('/', async (req, res) => {

  const blogSettings = await db.getSettings();

  const articles = await db.getPublishedArticles();

  res.render('readerHome', {

    blog\_title: blogSettings.blog\_title,

    author\_name: blogSettings.author\_name,

    articles: articles

  });

});

// Read Article Page

router.get('/article/:id', async (req, res) => {

  const article = await db.getArticleById(req.params.id);

  const comments = await db.getCommentsByArticleId(req.params.id);

  res.render('article', { article: article, comments: comments });

});

// Like Article

router.get('/article/:id/like', async (req, res) => {

  await db.likeArticle(req.params.id);

  res.redirect(`/reader/article/${req.params.id}`);

});

// Add Comment

router.post('/article/:id/comment', async (req, res) => {

  const { commenter\_name, comment } = req.body;

  await db.addComment(req.params.id, commenter\_name, comment);

  res.redirect(`/reader/article/${req.params.id}`);

});

module.exports = router;

**Views/article.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title><%= article.title %></title>

  <link rel="stylesheet" href="/styles.css">

</head>

<body>

  <header>

    <h1>Read Article Page</h1>

  </header>

  <main>

    <h2><%= article.title %></h2>

    <p>Published: <%= article.updated\_at %></p>

    <p>Views: <%= article.views %></p>

    <p>Likes: <%= article.likes %></p>

    <div>

      <p><%= article.content %></p>

    </div>

    <button onclick="window.location.href='/reader/article/<%= article.id %>/like'">Like</button>

    <section>

      <h3>Comments</h3>

      <ul>

        <% comments.forEach(comment => { %>

          <li>

            <p><strong><%= comment.commenter\_name %>:</strong> <%= comment.comment %></p>

            <p>Posted on: <%= comment.created\_at %></p>

          </li>

        <% }) %>

      </ul>

    </section>

    <section>

      <h3>Add a Comment</h3>

      <form action="/reader/article/<%= article.id %>/comment" method="post">

        <label for="commenter\_name">Name:</label>

        <input type="text" id="commenter\_name" name="commenter\_name" required>

        <label for="comment">Comment:</label>

        <textarea id="comment" name="comment" required></textarea>

        <button type="submit">Submit Comment</button>

      </form>

    </section>

    <nav>

      <a href="/reader">Back to Reader Home</a>

    </nav>

  </main>

</body>

</html>

**Views/authorHome.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Author Home</title>

  <link rel="stylesheet" href="/styles.css">

</head>

<body>

  <header>

    <h1>Author Home Page</h1>

  </header>

  <main>

    <h2>Blog Title: <%= blog\_title %></h2>

    <h3>Author: <%= author\_name %></h3>

    <nav>

      <a href="/author/settings">Settings</a>

      <br> <br>

      <form action="/author/create" method="post" style="display:inline;">

        <button type="submit">Create new draft</button>

      </form>

    </nav>

    <section>

      <h2>Published Articles</h2>

      <ul>

        <% publishedArticles.forEach(article => { %>

          <li>

            <h3><%= article.title %></h3>

            <p>Created: <%= article.created\_at %></p>

            <p>Published: <%= article.updated\_at %></p>

            <p>Last Modified: <%= article.updated\_at %></p>

            <p>Views: <%= article.views %></p>

            <p>Likes: <%= article.likes %></p>

            <a href="/reader/article/<%= article.id %>">Read Article</a>

            <form action="/author/delete/<%= article.id %>" method="post">

              <button type="submit">Delete</button>

            </form>

          </li>

        <% }) %>

      </ul>

    </section>

    <section>

      <h2>Draft Articles</h2>

      <ul>

        <% draftArticles.forEach(article => { %>

          <li>

            <h3><%= article.title %></h3>

            <p>Created: <%= article.created\_at %></p>

            <p>Last Modified: <%= article.updated\_at %></p>

            <a href="/author/edit/<%= article.id %>">Edit Article</a>

            <form action="/author/publish/<%= article.id %>" method="post">

              <button type="submit">Publish</button>

            </form>

            <form action="/author/delete/<%= article.id %>" method="post">

              <button type="submit">Delete</button>

            </form>

          </li>

        <% }) %>

      </ul>

    </section>

  </main>

</body>

</html>

**Views/editArticle.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Edit Article</title>

  <link rel="stylesheet" href="/styles.css">

</head>

<body>

  <header>

    <h1>Edit Article Page</h1>

  </header>

  <main>

    <form action="/author/edit/<%= article.id %>" method="post">

      <label for="title">Article Title:</label>

      <input type="text" id="title" name="title" value="<%= article.title %>" required>

      <label for="content">Article Content:</label>

      <textarea id="content" name="content" required><%= article.content %></textarea>

      <button type="submit">Submit Changes</button>

    </form>

    <nav>

      <a href="/author">Back to Author Home</a>

    </nav>

  </main>

</body>

</html>

**Views/index.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Home</title>

  <link rel="stylesheet" href="/styles.css">

</head>

<body>

  <header>

  <h1>Welcome to the Blog</h1>

</header>

  <nav >

    <br> <br>

    <a href="/auth">Author Home Page</a> <!-- Update this link -->

    <br> <br> <br> <br>

    <a href="/reader">Reader Home Page</a>

  </nav>

</body>

</html>

**Views/layout.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title><%= title %></title>

  <link rel="stylesheet" href="/styles.css">

</head>

<body>

  <header>

    <h1>My Blog</h1>

    <nav>

      <a href="/">Home</a>

      <a href="/author">Author Home</a>

      <a href="/reader">Reader Home</a>

    </nav>

  </header>

  <main>

    <%- content %>

  </main>

</body>

</html>

**Views/login.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Login</title>

  <link rel="stylesheet" href="/styles.css">

</head>

<body>

  <header>

  <h1>Author Login</h1>

</header>

  <form action="/auth" method="post">

    <label for="password">Password:</label>

    <input type="password" id="password" name="password" required>

    <button type="submit">Login</button>

  </form>

</body>

</html>

**Views/readerHome.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Reader Home</title>

  <link rel="stylesheet" href="/styles.css">

</head>

<body>

  <header>

    <h1>Reader Home Page</h1>

  </header>

  <main>

    <h2>Blog Title: <%= blog\_title %></h2>

    <h3>Author: <%= author\_name %></h3>

    <section>

      <h2>Published Articles</h2>

      <ul>

        <% articles.forEach(article => { %>

          <li>

            <h3><%= article.title %></h3>

            <p>Published: <%= article.updated\_at %></p>

            <a href="/reader/article/<%= article.id %>">Read Article</a>

          </li>

        <% }) %>

      </ul>

    </section>

  </main>

</body>

</html>

**Views/settings.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Settings</title>

  <link rel="stylesheet" href="/styles.css">

</head>

<body>

  <header>

    <h1>Settings Page</h1>

  </header>

  <main>

    <form action="/author/settings" method="post">

      <label for="blog\_title">Blog Title:</label>

      <input type="text" id="blog\_title" name="blog\_title" value="<%= settings.blog\_title %>" required>

      <label for="author\_name">:</label>

      <input type="text" id="author\_name" name="author\_name" value="<%= settings.author\_name %>" required>

      <button type="submit">Submit</button>

    </form>

    <nav>

      <a href="/author">Back to Author Home</a>

    </nav>

  </main>

</body>

</html>

**.env**

AUTHOR\_PASSWORD=mySuperSecurePassword123!

SESSION\_SECRET=yourVerySecretSessionKey987!

**buildDb.js**

const sqlite3 = require('sqlite3').verbose();

const db = new sqlite3.Database('./db/blog.db');

db.serialize(() => {

  db.run('CREATE TABLE IF NOT EXISTS settings (blog\_title TEXT, author\_name TEXT)');

  db.run('CREATE TABLE IF NOT EXISTS articles (id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT, content TEXT, created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP, updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP, published INTEGER DEFAULT 0, views INTEGER DEFAULT 0, likes INTEGER DEFAULT 0)');

  db.run('CREATE TABLE IF NOT EXISTS comments (id INTEGER PRIMARY KEY AUTOINCREMENT, article\_id INTEGER, commenter\_name TEXT, comment TEXT, created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP, FOREIGN KEY (article\_id) REFERENCES articles (id))');

  db.run('INSERT INTO settings (blog\_title, author\_name) VALUES (?, ?)', ['My Blog', 'Author Name']);

});

db.close();

**Db.js**

const sqlite3 = require('sqlite3').verbose();

const db = new sqlite3.Database('./db/blog.db');

module.exports = {

  getSettings: () => {

    return new Promise((resolve, reject) => {

      db.get('SELECT \* FROM settings', (err, row) => {

        if (err) reject(err);

        else resolve(row);

      });

    });

  },

  updateSettings: (blog\_title, author\_name) => {

    return new Promise((resolve, reject) => {

      db.run('UPDATE settings SET blog\_title = ?, author\_name = ?', [blog\_title, author\_name], (err) => {

        if (err) reject(err);

        else resolve();

      });

    });

  },

  getPublishedArticles: () => {

    return new Promise((resolve, reject) => {

      db.all('SELECT \* FROM articles WHERE published = 1 ORDER BY updated\_at DESC', (err, rows) => {

        if (err) reject(err);

        else resolve(rows);

      });

    });

  },

  getDraftArticles: () => {

    return new Promise((resolve, reject) => {

      db.all('SELECT \* FROM articles WHERE published = 0 ORDER BY updated\_at DESC', (err, rows) => {

        if (err) reject(err);

        else resolve(rows);

      });

    });

  },

  getArticleById: (id) => {

    return new Promise((resolve, reject) => {

      db.get('SELECT \* FROM articles WHERE id = ?', [id], (err, row) => {

        if (err) reject(err);

        else if (row) resolve(row);

        else reject(new Error('Article not found'));

      });

    });

  },

  updateArticle: (id, title, content) => {

    return new Promise((resolve, reject) => {

      db.run('UPDATE articles SET title = ?, content = ?, updated\_at = CURRENT\_TIMESTAMP WHERE id = ?', [title, content, id], (err) => {

        if (err) reject(err);

        else resolve();

      });

    });

  },

  publishArticle: (id) => {

    return new Promise((resolve, reject) => {

      db.run('UPDATE articles SET published = 1, updated\_at = CURRENT\_TIMESTAMP WHERE id = ?', [id], (err) => {

        if (err) reject(err);

        else resolve();

      });

    });

  },

  deleteArticle: (id) => {

    return new Promise((resolve, reject) => {

      db.run('DELETE FROM articles WHERE id = ?', [id], (err) => {

        if (err) reject(err);

        else resolve();

      });

    });

  },

  likeArticle: (id) => {

    return new Promise((resolve, reject) => {

      db.run('UPDATE articles SET likes = likes + 1 WHERE id = ?', [id], (err) => {

        if (err) reject(err);

        else resolve();

      });

    });

  },

  getCommentsByArticleId: (article\_id) => {

    return new Promise((resolve, reject) => {

      db.all('SELECT \* FROM comments WHERE article\_id = ? ORDER BY created\_at ASC', [article\_id], (err, rows) => {

        if (err) reject(err);

        else resolve(rows);

      });

    });

  },

  // Create a new draft article

createDraftArticle: () => {

    return new Promise((resolve, reject) => {

      db.run('INSERT INTO articles (title, content, published) VALUES (?, ?, ?)', ['New Draft', '', 0], function(err) {

        if (err) reject(err);

        else resolve(this.lastID);

      });

    });

  },

  addComment: (article\_id, commenter\_name, comment) => {

    return new Promise((resolve, reject) => {

      db.run('INSERT INTO comments (article\_id, commenter\_name, comment, created\_at) VALUES (?, ?, ?, CURRENT\_TIMESTAMP)', [article\_id, commenter\_name, comment], (err) => {

        if (err) reject(err);

        else resolve();

      });

    });

  }

};

**Index.js**

require('dotenv').config();

const express = require('express');

const session = require('express-session');

const path = require('path');

const app = express();

const PORT = process.env.PORT || 3000;

app.use(express.urlencoded({ extended: true }));

app.use(session({

  secret: process.env.SESSION\_SECRET,

  resave: false,

  saveUninitialized: false,

}));

app.set('view engine', 'ejs');

app.set('views', path.join(\_\_dirname, 'views'));

app.use(express.static(path.join(\_\_dirname, 'public'))); // Serve static files

// Routes

const authRouter = require('./routes/auth');

const authorRouter = require('./routes/author');

const readerRouter = require('./routes/reader');

app.use('/auth', authRouter);

app.use('/author', authorRouter);

app.use('/reader', readerRouter);

// Root route

app.get('/', (req, res) => {

  res.render('index');

});

app.listen(PORT, () => {

  console.log(`Server is running on http://localhost:${PORT}`);

});

**Package.json**

{

  "scripts": {

    "build-db": "node buildDb.js",

    "start": "node index.js"

  },

  "dependencies": {

    "dotenv": "^16.4.5",

    "express-session": "^1.18.0"

  }

}