Created by: Bryan Loo Chun Wai

Database, Networks and the Web
Mid-Term Assignment

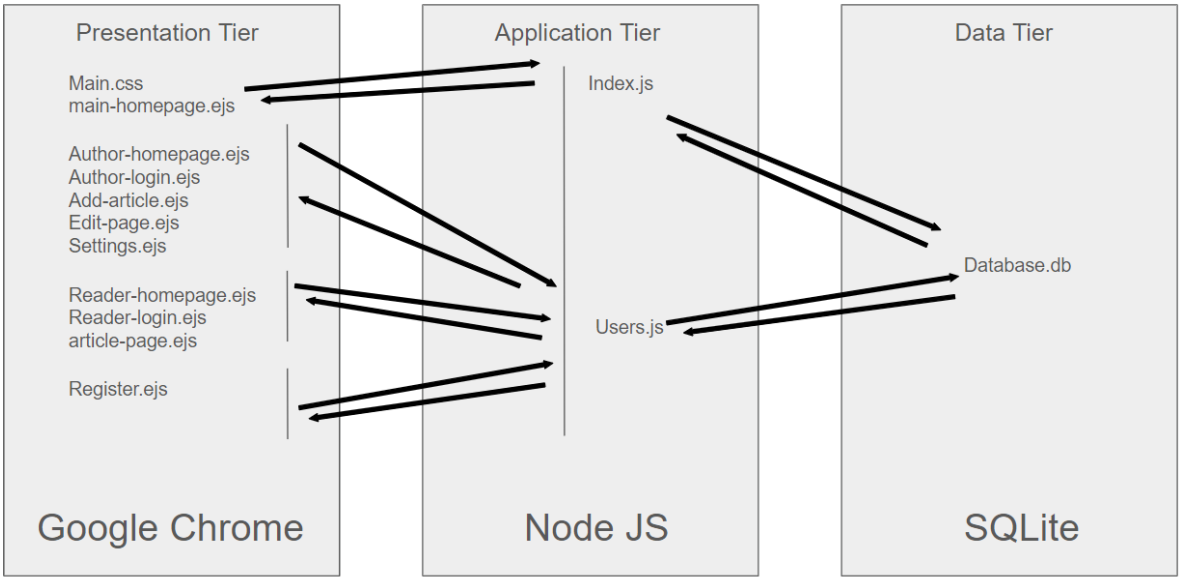
Table of Contents

Section 1 - Schematic Diagram

1.1) Introduction

For this assignment, I have been tasked to develop a blogging tool which a user can deploy their own server and maintain a blog . The project can handle two types of users, a author and a reader role. Some of the features that the project has is the write new articles, save them as a draft to initially store the article and when the author is confident can choose to publish the article. As for the reader role, the reader is able to browse all the published articles and comment on the selected articles. Overall, this application provides an interactive and dynamic user experience and practices efficient use of the 3 layer application.

1.2) 3-layer application diagram



Purpose:

The purpose of this 3 layer application diagram is to offer a visual and detailed explanation of the interactions between the 3 layers. The 3 layers consist of the presentation tier, application tier and lastly the data tier.

1.3) Explanation

Presentation tier

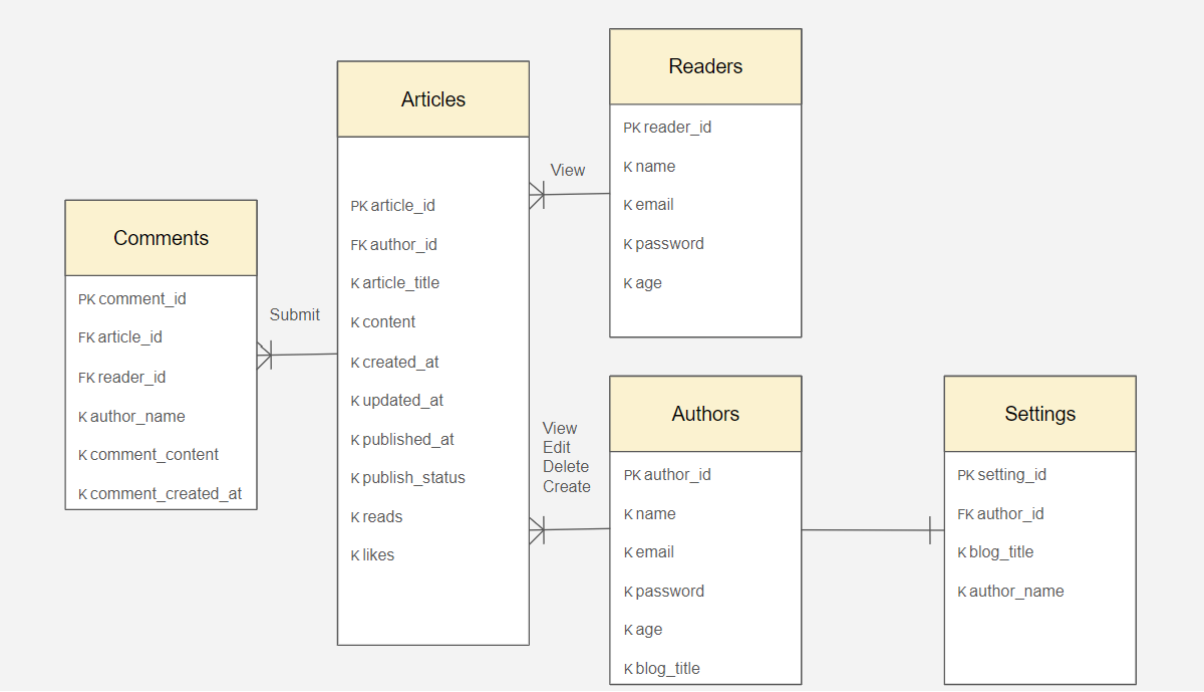
The presentation tier handles the front-end interface that the user interacts with. In other words it is what the user looks at when browsing the site. Some examples of interactions with the browser can include user input and pressing buttons. When the user inserts an input into an input text field, the data is passed to the application tier. This tier uses 3 main languages, HTML, CSS and Javascript. HTML is used to generate the basic elements of the application without any styling. CSS, allows the elements to be styled and structured to make the website presentable and user friendly for the user. And lastly, Javascript which handles the client side operations that are triggered by usually a button. An example of css is my main.css file that is linked to the other pages. And as for HTML, an example would be the main-homepage.ejs.

Application tier
The Application tier. In my application, I used the users.js and index.js to handle the business logic of the application. It is used as the middleware where it connects the interactions of the front end with the back end operations, For example, if the user creates a new article, the application tier will take that data inserted by the user and after the necessary checks passes that data and inserts it into the database. In my application, I used express.js routes to render each page based on the project requirement and specifications.

Data tier
The Data tier. This tier is used for the storing and retrieval of data from the database. In my program, I used SQLite3 as the software to represent my data tier. An example of this, would be the author home page, where it requires retrieval of information for articles that are in the draft status or published status.

Section 2 - Entity Relationship Diagram

2.1) ERD Diagram



Purpose:
The purpose of this Entity relationship diagram is to provide a visualisation of the database structure. It provides a representation of the structure of the database, showing the table names and columns, as well as the dynamic relationship and interactions that each table interacts with each other.

2.2) Explanation

In this ERD diagram, there are a total of 5 tables, Authors, Readers, Articles, Settings and Comments. The authors table has a one to many relationship with the Articles table. The Readers table has a one to many relationship with the Articles table. The Articles table has a one to many relationship with the comments table. The Author tables has a one to one relationship with the settings table. This means that the Author can create many articles and can comment on multiple articles. And one Author can only have one settings record in the table.

Section 3 - Extension

3.1) About extension

My extension for this project was to use the express session module for user verification and login authentication. This ensures user authenticity and integrity. By using validation, it allows any misuse of the application and unauthorised access to the application.

3.2) Code explanation of extension

Diagram 3.2.1 (POST to Author Login)

```
485  //author login verification, verifies if the author exists in the table
486  router.post('/author/login', (req, res) => {
487
488      const { name, password } = req.body;
489      const query = 'SELECT author_id FROM Authors WHERE name = ? AND password = ?'; // Adjusted query to select only author_id
490      const query_parameters = [name, password];
491
492      global.db.get(query, query_parameters, (err, user) => {
493          if (err) {
494              console.error(err);
495              return res.status(500).send('Error logging in.');
496
497          }
498          if (!user) {
499
500              return res.render('author-login.ejs', { error: 'Author does not exist. Please try again!' });
501          }
502
503          console.log("Session has started");
504          req.session.name = name;
505          req.session.userId = user.author_id; // Store user ID
506
507          res.redirect('/users/author'); // Redirect to a protected route
508      });
509  });
```

Explanation:

In diagram 3.2.1, it shows a post method for the author. This happens when the user attempts to log in as an author. In line 489, the user will insert the data into the input text fields and the server checks for existing users.

Diagram 3.2.2 (Middleware verifies user)

```
388   //middleware to check if user is logged in
389   function requireLogin(req, res, next) {
390
391       if (req.session.name) {
392           next(); // User authenticated
393
394       } else {
395           res.redirect('/author/login'); // Redirect to login page if not authenticated
396       }
397   }
398
```

Explanation:

In diagram 3.2.2, it shows a middleware function to check for user existence. In line 395, it redirects the user back to the login page and the server will return an error message for login failure.

Diagram 3.2.3 (Passes function in and uses user info to then display the information)

```
10  // Author home page route, brings user to author home page
11  router.get("/author", requireLogin, (req, res, next) => {
12
13      const draft_query = "SELECT * FROM Articles WHERE publish_status = 'draft'";
14      const published_query = "SELECT * FROM Articles WHERE publish_status = 'published'";
15      const settingsQuery = 'SELECT blog_title FROM Settings WHERE author_id = ?';
16
17      const user_query_parameters = [req.session.userId]; // Use userId from session
18
19      global.db.all(published_query, (err, published_rows) => {
20          if (err) {
21              next(err);
22
23          } else {
24              global.db.all(draft_query, (err, draft_rows) => {
25                  if (err) {
26                      next(err);
27                  } else {
28                      global.db.get(settingsQuery, user_query_parameters, (err, settingsData) => {
29                          if (err) {
30                              next(err);
31                          } else {
32
33                              res.render("author-homepage.ejs", {
34                                  name: req.session.name,
35                                  publishedArticles: published_rows,
36                                  draftArticles: draft_rows,
37                                  blogTitle: settingsData ? settingsData.blog_title : '', // Extracting blog_title
38                                  usersID: req.session.userId
39                              });
40                          }});
41              }});
42      }});
43  });
```

Explanation:

In diagram 3.2.3, it shows the get method for the author. This method uses the middleware function to verify the user. In line 11, it shows the requireLogin function being called. If the user fails to log in the user is redirected back to the log in page.