Author Routes File

```
/**
 * These are the routes for the author feature
 */

//Start of Code written without Assistance for this code extract

/**
 * These are the routes for the author feature
 */

const express = require('express');
const router = express.Router();
const date_time = new Date();
const { check, validationResult } = require('express-validator');


function formatDateForSQLite(date) {
  //Purpose: This function takes a JavaScript Date object and formats
it into a string that can be easily inserted into an SQLite database.
  //Inputs: A JavaScript Date object representing the date and time
to be formatted.
  //Outputs: A string in the format "YYYY-MM-DD HH:MM:SS" that can be
used to insert the date and time into an SQLite database.
  if (!(date instanceof Date)) {
      throw new Error("Invalid Date object");
  }
  const year = date.getUTCFullYear();
  const month = String(date.getUTCMonth() + 1).padStart(2, '0');
  const day = String(date.getUTCDate()).padStart(2, '0');
  const hours = String(date.getUTCHours()).padStart(2, '0');
  const minutes = String(date.getUTCMinutes()).padStart(2, '0');
  const seconds = String(date.getUTCSeconds()).padStart(2, '0');
  const formattedDate = `${year}-${month}-${day}
${hours}:${minutes}:${seconds}`;
  return formattedDate;
}


const requiredPerms = (req, res, next) =>
  // Purpose: This middleware function checks if the user making the
request has admin permissions.
  // Inputs: req (the Express request object), res (the Express
response object), next (the next middleware function in the stack).
  // Outputs: If the user is an admin, calls the next middleware
function. If the user is not an admin, renders the "AccessDenied"
view and sends the response.
  // Check if the user's ID in the session is equal to 1 (assuming 1
represents an admin user)
  {
      // Check if the user's ID in the session is not equal to 1
      if (req.session.userId !== 1)
      {
          // If the user does not have the required permissions,
          // render the "AccessDenied" view and send the response
          res.render("AccessDenied");
      }
```

```
      else
      {
          // If the user has the required permissions,
          // call the next middleware function in the stack
          next();
      }
};
const redirectlogin = (req, res, next) => {
  // Purpose: This middleware function checks if the user making the
request is logged in.
  // If the user is not logged in, it redirects them to the login
page.
  // If the user is logged in, it calls the next middleware function
in the stack.
  // Inputs: req (the Express request object), res (the Express
response object), next (the next middleware function in the stack).
  // Outputs: If the user is not logged in, it redirects them to the
login page. If the user is logged in, it calls the next middleware
function.
  if (!req.session.username) {
    res.redirect('/login');
  } else {
    next();
  }
}


router.get('/create-post', requiredPerms, (req, res) =>
  // Purpose: This route handler function is responsible for
rendering the 'create-post' page, which allows authenticated users
with the required permissions to create a new blog post.
  // Inputs: req (the Express request object), res (the Express
response object), requiredPerms (a middleware function that checks if
the user has the necessary permissions to access this route).
  // Outputs: The function renders the 'create-post' view.
  {
      // Render the 'create-post' view
      res.render('create-post');
  });

router.post('/create-post-process', (req, res, next) => {
  const { title, subtitle, content } = req.body;
  const userId = req.session.userId; // Assuming you have stored the
userId in the session


  const createdAt = new Date();

  const formattedcreated = formatDateForSQLite(createdAt);

  // Insert the new post into the Posts table with the current date
and time
  global.db.run(
    `INSERT INTO Posts (title, subtitle, content, userId, createdAt)
VALUES (?, ?, ?, ?, ?)`,
    [title, subtitle, content, userId, formattedcreated],
    function (err) {
      if (err) {
```

```
        // Handle the error appropriately
        return next(err);
      }

      // The post was successfully created
      const postId = this.lastID;

      // Redirect to the newly created post's page or any other
desired location
      res.redirect('/author/author-home');
    }
  );
});


router.get('/view-post', (req, res) => {
  // Purpose: This route handler function is responsible for
rendering the 'view-post' page, which displays the details of a
specific blog post.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: The function renders the 'view-post' view, passing any
necessary data (e.g., the post details) as part of the response.
  try {
    // Render the 'view-post' view
    res.render('view-post');
  } catch (error) {
    console.error('Error rendering view-post:', error);
    res.status(500).render('error', { message: 'An error occurred
while rendering the view-post page.' });
  }
});

router.get('/author-settings', (req, res) => {
   // Purpose: This route handler function is responsible for
rendering the 'author-settings' page, which allows an author to view
and update their account settings.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: The function renders the 'author-settings' view,
passing the user's current blog settings as part of the response.
  try {
    global.db.get('SELECT * FROM Settings', (err, row) => {
      if (err) {
        console.error('Error fetching settings:', err);
        res.status(500).render('error', { message: 'An error occurred
while retrieving the settings.' });
      } else {
        console.log('Success passing to the page.');
        res.render('author-settings', { settings: row });
      }
    });
  } catch (error) {
    console.error('Error in /author-settings route:', error);
    res.status(500).render('error', { message: 'An unexpected error
occurred.' });
  }
});
```

```
router.post('/update-settings', [
  check('title').not().isEmpty().withMessage('Title cannot be
empty'),
  check('subtitle').not().isEmpty().withMessage('Subtitle cannot be
empty'),
  check('author').not().isEmpty().withMessage('Author cannot be
empty'),
], (req, res) => {
  // Purpose: This route handler function is responsible for updating
the blog settings, such as the title, subtitle, and author name.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: The function either renders the 'author-settings' view
with validation errors, or redirects the user to the '/author/author-
settings' route after successfully updating the settings
  try {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      console.error('Validation errors:', errors.array());
      return res.status(400).render('author-settings', {
        settings: req.body,
        errors: errors.array(),
      });
    }

    const { title, subtitle, author } = req.body;

    const sql = 'UPDATE Settings SET title = ?, subtitle = ?, author
= ?';

    global.db.run(sql, [title, subtitle, author], function (err) {
      if (err) {
        console.error('Error updating settings:', err);
        return res.status(500).render('error', { message: 'An error
occurred while updating the settings.' });
      }
      res.redirect('/author/author-settings');
    });
  } catch (error) {
    console.error('Error in /update-settings route:', error);
    res.status(500).render('error', { message: 'An unexpected error
occurred.' });
  }
});

router.get('/author-home', redirectlogin, requiredPerms, async (req,
res) => {
  // Purpose: This route handler function is responsible for
rendering the author-home page, which displays the blog settings and
the list of posts with their comment counts.
// Inputs: req (the Express request object), res (the Express
response object).
// Outputs: The function renders the 'author-home' view and passes
data to the page such as posts.
  try {
    // Fetch settings from the database
    const settingsRow = await new Promise((resolve, reject) => {
```

```
      global.db.get('SELECT * FROM Settings', (err, row) => {
        if (err) {
          reject(err);
        } else {
          resolve(row);
        }
      });
    });

    // Fetch posts and their comment count from the database
    const postsRows = await new Promise((resolve, reject) => {
      global.db.all('SELECT p.*, (SELECT COUNT(*) FROM Comments c
WHERE c.postId = p.id) AS commentCount FROM Posts p', (err, rows) =>
{
        if (err) {
          reject(err);
        } else {
          resolve(rows);
        }
      });
    });

    console.log("postsRows:", postsRows);

    // Dynamic title
    const dynamicTitle = 'Author Home Page | CuteBlog Blogging';

    // Render the 'author-home' view with the necessary data
    res.render('author-home', {
      settings: settingsRow,
      posts: postsRows,
      title: dynamicTitle,
    });
  } catch (err) {
    console.error(err);
    res.status(500).send('Internal Server Error');
  }
});

router.get('/DashBoard', (req, res) => {
  // Purpose: This route handler function is responsible for
rendering the dashboard page, which displays a list of all published
blog posts.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: The function fetches all published posts from the
database and renders the dashboard view, passing the posts and the
current user's username as data.
      db.all('SELECT * FROM Posts WHERE status = "published"', (err,
posts) => {
        if (err) {
          console.error('Error fetching published posts:', err);
          return res.status(500).send('Error fetching published
posts');
        }

        res.render('DashBoard', {
          title: "Dashboard | CuteBlog Blogging",
```

```
          username: req.session.username,
          posts: posts
        });
      });
});

router.post('/delete-post', (req, res) => {
  // Purpose: This route handler function is responsible for deleting
a post from the database.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: The function redirects the user to the '/author/author-
home' route if the post is successfully deleted, or sends an error
response if there are any issues.
  try {
    const postId = req.body.postid;

    console.log(postId);



    // Delete the post from the database
    db.run('DELETE FROM Posts WHERE id = ?', [postId], function(err)
{
      if (err) {
        console.log(err);
        return res.status(500).send('Error deleting the post.');
      }

      // Check if any rows were affected
      if (this.changes === 0) {
        return res.status(404).send('Post not found.');
      }

      res.redirect('/author/author-home');
    });
  } catch (error) {
    console.error(error);
    res.status(500).send('An error occurred.');
  }
});

router.post('/delete-publishedpost', (req, res) => {
  // Purpose: This route handler function is responsible for deleting
a published post from the database.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: The function redirects the user to the '/author/author-
home' route if the post is successfully deleted, or return an error
message if any issues arises.
  try {
    const postId = req.body._postid;
    const userId = req.session.userId;

    console.log(postId);
    console.log(userId);

    // Check if the post is published before deleting
```

```
    db.get('SELECT status FROM Posts WHERE id = ?', [postId], (err,
row) => {
      if (err) {
        console.error('Error checking post status:', err);
        return res.status(500).render('error', { message: 'An error
occurred while checking the post status.' });
      }

      if (!row || row.status !== 'published') {
        return res.status(404).render('error', { message: 'Post not
found or not published.' });
      }

      // Delete the post from the database
      db.run('DELETE FROM Posts WHERE id = ?', [postId],
function(err) {
        if (err) {
          console.error('Error deleting the post:', err);
          return res.status(500).render('error', { message: 'An error
occurred while deleting the post.' });
        }

        // Check if any rows were affected
        if (this.changes === 0) {
          return res.status(404).render('error', { message: 'Post not
found.' });
        }

        res.redirect('/author/author-home');
      });
    });
  } catch (error) {
    console.error('Error in /delete-publishedpost route:', error);
    res.status(500).render('error', { message: 'An unexpected error
occurred.' });
  }
});

router.post('/edit-post', async (req, res) => {
// Purpose: This route handler function is responsible for rendering
the edit-post page for a specific post.
// Inputs: req (the Express request object), res (the Express
response object).
// Outputs: The function renders the 'edit-post' view and passes the
post information to the view.
  try {
    const postId = req.body.postid_for;

    // Fetch the post from the database
    const post = await new Promise((resolve, reject) => {
      db.get('SELECT * FROM Posts WHERE id = ?', [postId], (err, row)
=> {
        if (err) {
          reject(err);
        } else {
          resolve(row);
        }
      });
```

```
    });

    if (!post) {
      return res.status(404).send('Post not found.');
    }

    // Render the 'edit-post' view and pass the post data
    res.render('edit-post', {
      post: post
    });
  } catch (err) {
    console.error('Error retrieving post:', err.message);
    res.status(500).send('Error retrieving post.');
  }
});

router.post('/update-post', (req, res) => {
  // Purpose: This route handler function is responsible for updating
an existing post.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: The function updates the post in the database and
redirects the user to the author home page.
  const { postId, title, subtitle } = req.body;
  const content = req.body.content;
  const modifiedAt = new Date(); // Get the current time
  const formattedmodifed = formatDateForSQLite(modifiedAt);

  // Fetch the existing post content
  db.get('SELECT content FROM Posts WHERE id = ?', [postId], (err,
existingPost) => {
    if (err) {
      console.log(err.message);
      return res.status(500).send('Error fetching post.');
    }

    // Insert the old content into the PostHistory table
    db.run(
      'UPDATE Posts SET PostHistory = ?, modifiedAt = ?  WHERE id
= ?',
      [existingPost.content, formattedmodifed, postId],
      (err) => {
        if (err) {
          console.log(err.message);
          return res.status(500).send('Error saving post history.');
        }

        // Update the post with the new content
        db.run(
          'UPDATE Posts SET title = ?, subtitle = ?, content = ?,
ModifiedAt = ? WHERE id = ?',
          [title, subtitle, content, formattedmodifed, postId],
          (err) => {
            if (err) {
              console.log(err.message);
              return res.status(500).send('Error updating post.');
            } else {
```

```
              // Redirect the user to the updated post or the list of
posts
              return res.redirect('/author/author-home'); // or
res.redirect('/posts');
            }
          }
        );
      }
    );
  });
});

router.post('/publish-post/:postId', (req, res) => {
  // Purpose: This route handler function is responsible for
publishing a post by updating its status and publishedAt date in the
database.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: If the post is successfully published, the function
redirects the user to the author-home page. If an error occurs, the
function sends an respective error response.
  try {
    const postId = req.params.postId;

    // Get the current date and time
    const publishedDate = new Date();

    const formattedpublish = formatDateForSQLite(publishedDate);

    console.log(formattedpublish);

    // Update the status of the post to 'published' in the database
    db.run('UPDATE Posts SET status = ?, publishedAt = ? WHERE id
= ?', ['published', formattedpublish, postId], function(err) {
      if (err) {
        console.error(err.message);
        return res.status(500).send('Error publishing the post.');
      }

      // Check if any rows were affected
      if (this.changes === 0) {
        return res.status(404).send('Post not found.');
      }

      res.redirect('/author/author-home');
    });
  } catch (error) {
    console.error(error);
    res.status(500).send('An error occurred.');
  }
});


router.post('/version-history', (req, res) => {
  // Purpose: This route handler function is responsible for
rendering the version history for a specific post.
  // Inputs: req (the Express request object), res (the Express
response object).
```

```
  // Outputs: The function renders the 'version-history' view and
passes the past version of the particular post to the view.

  const postId = req.body.postid_;


  // Fetch the post information
  db.get(`SELECT * FROM posts WHERE id = ?`, [postId], (err, post) =>
{
    if (err) {
      console.error(err);
      return res.status(500).send('Error fetching post information');
    }

    if (!post) {
      return res.status(404).send('Post not found');
    }

    // Render the post information to the page
    res.render('version-history', { post });
  });
});

router.get('/',(req,res) =>
  // Purpose: This route handler function is responsible for
rendering the "author-home" view when a user visits the root URL of
the author path.
  // Inputs: req (the Express request object), res (the Express
response object).
  // Outputs: The function renders the "author-home" view and sends
the response back to the client.
{
     res.render('author-home')
});



module.exports = router;


//End of Code written without Assistance for this code extract
```

Reader Routes Files

```
/**
 * These are the routes for the reader feature
 */
const express = require('express');
const router = express.Router();
const date_time = new Date();
const { check, validationResult } = require('express-validator');
const session = require('express-session');
const bodyParser = require('body-parser');
router.use(bodyParser.urlencoded({ extended: false }));
router.use(session({ secret: 'secret-key', resave: false,
saveUninitialized: true }));
const bcrypt = require('bcrypt');
const rateLimit = require('express-rate-limit');
```

```
//Start of Code written without Assistance for this code extract

const loginLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 4, // limit each IP to 4 requests per windowMs
  message: 'Too many login/register attempts from this IP, please try
again after 15 minutes'
});

const redirectlogin = (req, res, next) => {
  if (!req.session.username) {
    res.redirect('/login');
  } else {
    console.log("The session UserId is set as: " +
req.session.userId);
    next();
  }
}

router.get('/', (req, res) => {
  // Purpose: This route is responsible for rendering the main page
of the application.

  // Render the 'mainpage' view and send the response to the client.
  res.render('mainpage');
});

router.get('/register', function (req, res) {
  // Purpose: This route is responsible for rendering the
registration page.

  // Render the 'register' view and pass the title parameter to the
view.
  res.render('register', {
    title: "Account Register | CuteBlog Blogging"
  });
});

router.post('/register-process', [
  // Middleware to validate the input fields
  check('email').isEmail(),
  check('username').not().isEmpty(),
  check('password').isLength({ min: 8 }),
],
loginLimiter,
function(req, res) {
  // Purpose: This route is responsible for processing the user
registration form.

  // Validate the input fields
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    console.log(errors);
    return res.redirect('/register');
  }

  // Sanitize and retrieve the input values
  const plainPassword = req.sanitize(req.body.password);
```

```
  const sanitizedUsername = req.sanitize(req.body.username);
  const email = req.sanitize(req.body.email);

  const saltRounds = 10;

  // Check if the username already exists
  const sql = 'SELECT * FROM Users WHERE username = ?';
  db.get(sql, [sanitizedUsername], function(err, row) {
    if (err) {
      console.error(err);
      return res.status(500).send('Error checking for existing
username');
    }

    if (row) {
      console.log("Username has been taken!");
      return res.render('register', { usernameError: 'Username
already taken' });
    }

    // Hash the password
    bcrypt.hash(plainPassword, saltRounds, function(err,
hashedPassword) {
      if (err) {
        console.error(err);
        return res.status(500).send('Error hashing password');
      }

      // Insert the data into the SQLite database
      const insertSql = 'INSERT INTO Users (username, password,
email) VALUES (?, ?, ?)';
      const values = [sanitizedUsername, hashedPassword, email];

      db.run(insertSql, values, function(err) {
        if (err) {
          console.error(err);
          return res.status(500).send('Error inserting data into the
database');
        }
        else {
          // Sending confirmation message by rendering empty ejs
template with array of strings
          var message = ["Success!", "You are now registered. You may
login as " + sanitizedUsername + " now.", "Your user name is: " +
sanitizedUsername, "Your hashed password is: " + plainPassword];
          res.render('confirmation.ejs', { messages: message });
        }
      });
    });
  });
});

router.get('/login', function (req, res) {
  // Purpose: This route is responsible for rendering the login page.

  // Render the 'login' view and pass the title parameter to the
view.
  res.render('login', {
```

```
    title: "Account Login | CuteBlog Blogging"
  });
});


router.post('/login-process', loginLimiter, function (req, res) {
  // Purpose: This route is responsible for handling the login
process.
  // It receives the user's login credentials, validates them against
the database,
  // and manages the user's session if the login is successful.

  // Inputs:
  // - req.body.password: The plain-text password entered by the
user.
  // - req.body.input: This can be either the user's username or
email.
  // - loginLimiter: A middleware function that limits the number of
login attempts
  //   to prevent brute-force attacks.

  const input = req.body.input;
  const plainPassword = req.body.password;

  db.serialize(function() {
    // Searching the database for the username or email
    db.get('SELECT * FROM users WHERE username = ? OR email = ?',
[input, input], function(err, row) {
      if (err) throw err;

      if (!row) {
        // Username/email not found, sending an error message
        var message = ["Failure", "Login unsuccessful.",
"Username/email incorrect, please try again."];
        res.render('confirmation.ejs', { messages: message });
      } else {
        // Comparing entered password to the hashed password in the
database
        const hashedPassword = row.password;
        bcrypt.compare(plainPassword, hashedPassword, function(err,
result) {
          if (err) throw err;
          else {
            if (!result) {
              // Passwords don't match, sending an error message
              var message = ["Failure", "Login unsuccessful.",
"Password incorrect, please try again."];
              res.render('confirmation.ejs', { messages: message });
            } else {
              // Saving user session since login was successful
              req.session.userId = row.id;
              req.session.email = row.email;
              req.session.username = row.username;
              // Sending a confirmation by rendering empty ejs
template with array of strings
              var message = ["Success!", "You are now logged in."];
              res.render('confirmation.ejs', { messages: message });
            }
```

```
          }
        });
      }
    });
  });
});

router.get('/about', function (req, res) {
  // Purpose: This route is responsible for rendering the "About"
page.

  // Render the 'about' view and pass the title parameter to the
view.
  res.render('about', {
    title: "About the developer | CuteBlog Blogging"
  });
});

router.get('/DashBoard', redirectlogin, (req, res) => {
  // Purpose: This route is responsible for rendering the Dashboard
page, which displays
  // a list of published blog posts and the blog settings.

  // Inputs:
  // - redirectlogin: A middleware function that redirects the user
to the login page
  //   if they are not logged in.
  // - req.session.username: The username of the currently logged-in
user, which is
  //   retrieved from the session.

  // Fetch the published blog posts
  db.all('SELECT * FROM Posts WHERE status = "published" ORDER BY
publishedAt DESC', (err, posts) => {
    if (err) {
      console.error('Error fetching published posts:', err);
      return res.status(500).send('Error fetching published posts');
    }

    // Fetch the blog settings
    db.get('SELECT * FROM Settings', (err, settings) => {
      if (err) {
        console.error('Error fetching blog settings:', err);
        return res.status(500).send('Error fetching blog settings');
      }

      // Render the 'DashBoard' view and pass the following data:
      // - title: The page title
      // - username: The username of the currently logged-in user
      // - posts: An array of published blog posts, ordered by
publication date in descending order
      // - settings: The blog settings
      res.render('DashBoard', {
        title: "Dashboard | CuteBlog Blogging",
        username: req.session.username,
        posts: posts,
        settings: settings
      });
```

```
    });
  });
});

router.get('/view-post/:postid', function (req, res) {
  // Purpose: This route is responsible for rendering the "View Post"
page, which displays
  // the details of a specific blog post and its associated comments.

  // Inputs:
  // - req.params.postid: The ID of the blog post to be viewed.
  // - req.session.userId: The ID of the currently logged-in user.
  // - req.session.username: The username of the currently logged-in
user.

  const postid = req.params.postid;
  const userId = req.session.userId;
  const username = req.session.username;

  // Queries:
  // - commentsQuery: Retrieves the count of comments for the given
post.
  // - commentQuery: Retrieves the comments for the given post,
ordered by creation date in descending order.
  // - postQuery: Retrieves the details of the blog post with the
given ID.
  // - userQuery: Retrieves the details of the user with the given
ID.
  // - updateViewCountQuery: Increments the view count for the given
blog post.

  const commentsQuery = 'SELECT COUNT(*) AS commentCount FROM
Comments WHERE postId = ?';
  const commentQuery = 'SELECT * FROM Comments WHERE postId = ? ORDER
BY createdAt DESC';
  const postQuery = 'SELECT * FROM Posts WHERE id = ?';
  const userQuery = "SELECT * FROM Users WHERE id = ?";
  const updateViewCountQuery = 'UPDATE Posts SET Views = Views + 1
WHERE id = ?';

  // Fetch the comment count for the given post
  db.get(commentsQuery, [postid], (err, commentCount) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ error: 'Internal server
error' });
    }

    // Fetch the comments for the given post
    db.all(commentQuery, [postid], (err, comments) => {
      if (err) {
        console.error(err);
        return res.status(500).json({ error: 'Internal server
error' });
      }

      // Fetch the details of the blog post with the given ID
      db.get(postQuery, [postid], (err, post) => {
```

```
        if (err) {
          console.error(err);
          return res.status(500).json({ error: 'Internal server
error' });
        }

        if (!post) {
          return res.status(404).json({ error: 'Post not found' });
        }

        // Update the view count for the blog post
        db.run(updateViewCountQuery, [postid], (err) => {
          if (err) {
            console.error(err);
            return res.status(500).json({ error: 'Internal server
error' });
          }

          // Fetch the details of the user who created the blog post
          db.get(userQuery, [userId], (err, user) => {
            if (err) {
              console.error(err);
              return res.status(500).json({ error: 'Internal server
error' });
            }

            if (!user) {
              return res.status(404).json({ error: 'User not
found' });
            }

            // Render the 'view-post' view and pass the following
data:
            // - post: The details of the blog post
            // - comments: The comments for the blog post
            // - commentCount: The count of comments for the blog
post
            // - user: The details of the user who created the blog
post
            // - username: The username of the currently logged-in
user
            res.render('view-post', { post: post, comments: comments,
commentCount: commentCount.commentCount, user: user, username:
username });
          });
        });
      });
    });
  });
});

router.post('/like-post', function (req, res) {
  // Purpose: This route is responsible for updating the likes count
of a particular blog post based on post ID.

  // Inputs:
  // - req.body.postLikeId - The ID of the post being liked.
  // - req.body.userId - The ID of the user liking the post.
```

```
  const postId = req.body.postLikeId;

  // Directly update the likes count in the Posts table
  const updateLikesQuery = "UPDATE Posts SET likes = likes + 1 WHERE
id = ?";
  db.run(updateLikesQuery, [postId], (err) => {
    if (err) {
      // If there is an error updating the likes count, log the error
and send a 500 Internal Server Error response.
      console.error(err);
      return res.status(500).send('Error updating likes count');
    }

    // Redirect the user to the view-post page after updating the
likes count.
    res.redirect(`/view-post/${postId}`);
  });
});


router.post('/dislike-post', function (req, res) {
  // Purpose: This route is responsible for updating the dislikes
count of a particular blog post based on post ID.

  // Inputs:
  // - req.body.postdisLikeId - The ID of the post being disliked.
  // - req.body.userrid - The ID of the user disliking the post.

  const postId = req.body.postdisLikeId;
  const userId = req.body.userrid; // Assuming you have the user's ID
in the request

  // Update the dislikes count in the Posts table
  const updateDislikesQuery = "UPDATE Posts SET dislikes = dislikes +
1 WHERE id = ?";
  db.run(updateDislikesQuery, [postId], (err) => {
    if (err) {
      // If there is an error updating the dislikes count, log the
error and send a 500 Internal Server Error response.
      console.error(err);
      return res.status(500).send('Error updating dislikes count');
    }

    // Redirect the user to the dashboard page after updating the
dislikes count.
    res.redirect(`/view-post/${postId}`);
  });
});

router.post('/add-comments', function(req, res) {
  // Purpose: This route is responsible for adding a new comment to
the database.

  // Inputs:
  // - req.body.pstid - The ID of the post the comment is being added
to.
  // - req.body.comment - The text content of the comment.
```

```
  // - req.body.commenter - The username of the user making the
comment.

  const AddComments = 'INSERT INTO Comments (postId, text,
comment_user) VALUES (?, ?, ?)';
  const postId = req.body.pstid;
  const text = req.body.comment;
  const userName = req.body.commenter;

  db.all(AddComments, [postId, text, userName], (err, result) => {
    if (err) {
      // If there is an error adding the comment, log the error and
send a 500 Internal Server Error response with an error message.
      console.error('Error adding comment:', err);
      return res.status(500).json({ error: 'Error adding comment' });
    } else {
      // If the comment is added successfully, redirect the user to
the dashboard.
      res.redirect(`/view-post/${postId}`);
    }
  });
});

router.get('/settings', function (req, res) {
  // Purpose: This route is responsible for rendering the user
settings page.

  // Inputs:
  // - None

  // Render the 'settings' view.
  res.render('settings');
});

//End of Code written without Assistance for this code extract

module.exports = router;
```

Common Route Files

```
const express = require('express');
const router = express.Router();
const date_time = new Date();
const { check, validationResult } = require('express-validator');
const session = require('express-session');
const bodyParser = require('body-parser');
router.use(bodyParser.urlencoded({ extended: false }));
const Joi = require('joi');
router.use(session({ secret: 'secret-key', resave: false,
saveUninitialized: true }));

//Start of Code written without Assistance for this code extract
router.get('/logout', function (req, res) {
  // Purpose: This route is responsible for destroying the user's
session and redirecting them to the login page.

  // Inputs:
  // - req.session - The user's current session.
```

```
  // Destroy the session
  req.session.destroy(function (err) {
    if (err) {
      // If there is an error destroying the session, log the error
and send a 500 Internal Server Error response.
      console.error('Error destroying session:', err);
      res.sendStatus(500);
    } else {
      // If the session is destroyed successfully, log a success
message and redirect the user to the login page.
      console.log("Session has been successfully destroyed");
      res.redirect('/login');
    }
  });
});

router.get('/AccessDenied', function (req, res) {
  // Purpose: This route is responsible for rendering the
'AccessDenied' view when an unauthorized user attempts to access a
restricted resource.

  // Inputs:
  // - None

  // Render the 'AccessDenied' view and pass in a title for the page.
  res.render('AccessDenied', {
    title: "Access Denied | CuteBlog Blogging",
  });
});

router.get('/user-profile', function (req, res) {
  // Purpose: This route is responsible for rendering the user
profile page with the user's information.

  // Inputs:
  // - req.session.userId - The ID of the user whose profile is being
displayed.

  const userId = req.session.userId;

  // Fetch the user information from the database using the userId
  db.get('SELECT * FROM users WHERE id = ?', [userId], (err, user) =>
{
    if (err) {
      // If there is an error fetching the user information, log the
error and send a 500 Internal Server Error response.
      console.error(err);
      return res.status(500).send('Error fetching user information');
    }

    if (!user) {
      // If the user is not found in the database, send a 404 Not
Found response.
      return res.status(404).send('User not found');
    }

    // Get the current date in the local format.
```

```
    const currentDate = new Date().toLocaleDateString();

    // Render the 'user-profile' view and pass in the user's
information and the current date.
    res.render('user-profile', {
      title: "User Settings | CuteBlog Blogging",
      currentDate: currentDate,
      user: user
    });
  });
});

router.post('/user-settings', function (req, res) {
  // Purpose: This route is responsible for updating a user's
settings, including their username and email.

  // Define the validation schema
  const schema = Joi.object({
    username: Joi.string()
      .min(8)
      .max(20)
      .required()
      .messages({
        'string.min': 'Username must be at least 8 characters long',
        'string.max': 'Username must be no more than 20 characters
long',
        'any.required': 'Username is required'
      }),
    email: Joi.string()
      .email({ tlds: { allow: false } })
      .required()
      .messages({
        'string.email': 'Invalid email address',
        'any.required': 'Email is required'
      })
  });

  // Inputs:
  const { username, email } = req.body;
  const userId = req.session.userId;

  // Validate the request body using the schema
  const { error } = schema.validate(req.body);
  if (error) {
    // If there is a validation error, return the first error message
    return res.status(400).json({ error: error.details[0].message });
  }

  db.run(
    'UPDATE users SET username = ?, email = ? WHERE id = ?',
    [username, email, userId],
    function (err) {
      if (err) {
        // If there is an error updating the user's settings, log the
error and send a 500 Internal Server Error response.
        console.error(err);
        return res.status(500).send('Error updating user settings');
      }
```

```
      res.redirect('/common/user-profile');
    }
  );
});

//End of Code written without Assistance for this code extract

module.exports = router;
```

About.css

```
//Start of Code written without Assistance for this code extract

body {
    background-image: url("/assets/background.jpg");
    background-size: cover;
    background-repeat: no-repeat;
    background-color: rgba(255, 255, 255, 0.8);
     }

  /* Heading */
  h1 {
    color: #ff6ac9;
    font-size: 32px;
    font-family: "Comic Sans MS", cursive;
    text-align: center;
  }

  /* Paragraphs */
  p.text {
    color: #c966f0;
    font-size: 18px;
    font-family: "Arial", sans-serif;
    text-align: justify;
    line-height: 1.5;
  }

  /* Image */
  img {
    display: block;
    margin: 0 auto;
    max-width: 300px;
    border-radius: 50%;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
  }

  /* Spacing */
  .wp-block-spacer {
    height: 50px;
  }

  .text-container {
    background-color: #fff1f8;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    max-width: 600px;
```

```
    margin: 0 auto;
  }

  .text-container h1 {
    color: #ff6ac9;
    font-size: 32px;
    font-family: "Comic Sans MS", cursive;
    text-align: center;
    margin-bottom: 20px;
  }

  .text-container p.text {
    color: #ff70b8;
    font-size: 18px;
    font-family: "Arial", sans-serif;
    text-align: justify;
    line-height: 1.5;
  }

 //End of Code written without Assistance for this code extract
```

AccessDenied.css

```
//Start of Code written without Assistance for this code extract
body {
    background-color: #ffe6f2;
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }

  .container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
  }

  .content {
    text-align: center;
  }

  h1 {
    font-size: 36px;
    color: #ff00cc;
    margin-bottom: 10px;
  }

  p {
    font-size: 24px;
    color: #99004d;
    margin-bottom: 20px;
  }

  img {
    width: 300px;
    border-radius: 50%;
```

```
  }

  .Return {
    background-color: lightpink;
    color: #fff;
    border: none;
    padding: 10px 20px;
    font-size: 16px;
    text-decoration: none;
    cursor: pointer;
    border-radius: 5px;
    transition: background-color 0.3s ease;
  }

  .Return:hover {
    background-color: palevioletred;
  }

  .Return a {
    text-decoration: none;
  }

//End of Code written without Assistance for this code extract
```

Author-Home.css

```
/* Author-home.css */

//Start of Code written without Assistance for this code extract

.author-page {
  padding: 20px;
}

.published-articles,
.draft-articles {
  width: 100%;
  border-collapse: collapse;
  margin-bottom: 20px;
  background-color: whitesmoke;
}

.published-articles th,
.draft-articles th {
  background-color: rgba(242, 242, 242, 0.2);
  padding: 10px;
  text-align: left;
}

.published-articles td,
.draft-articles td {
  padding: 10px;
  border-bottom: 1px solid #ddd;
}

.published-articles .like,
.draft-articles .like,
```

```
.published-articles .dislike,
.draft-articles .dislike,
.published-articles .View-post,
.draft-articles .View-post,
.published-articles .Edit-post,
.draft-articles .Edit-post,
.published-articles .Delete-post,
.draft-articles .Delete-post,
.published-articles .Publish-post,
.draft-articles .Publish-post {
  background-color: transparent;
  border: none;
  cursor: pointer;
  padding: 0;
  margin-right: 5px;
}

.published-articles .like img,
.draft-articles .like img,
.published-articles .dislike img,
.draft-articles .dislike img,
.published-articles .Edit-post img,
.draft-articles .Edit-post img {
  width: 20px;
  height: 20px;
  vertical-align: middle;
}

.no-posts {
  text-align: center;
  font-style: italic;
  color: #999;
}

.header-buttons {
  background-color: #64c4a1; /* Soft green color */
  border: none;
  color: white;
  padding: 14px 28px; /* Increased padding */
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 18px; /* Increased font size */
  margin: 6px 4px; /* Increased spacing */
  cursor: pointer;
  border-radius: 20px; /* Rounded corners */
  box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2); /* Add a subtle drop
shadow */
  transition: background-color 0.3s ease; /* Smooth transition on
hover */
}

.header-buttons a {
  color: white;
  text-decoration: none;
}

.header-buttons:hover {
```

```
  background-color: #46b38c; /* Slightly darker green on hover */
}

.dust img
{
  width: 20px;
  height: 20px;
  vertical-align: middle;
}

body {
  background-image: url(/assets/background.png);
  background-repeat: no-repeat;
  background-size: cover;
}

h2
{
  color:white;
}

.subtitle-settings
{
  color:black;
}

.links
{
  text-decoration: none;
}

/* Breadcrumb */
.breadcrumb {
  display: flex;
  list-style: none;
  padding: 10px 0;
  background-color: #f1f1f1;
  margin: 10px 0;
  border-radius: 5px;
}

.breadcrumb li {
  padding-left: 10px;
  margin-right: 10px;
  color: #007bff;
}

.breadcrumb li::after {
  content: "/";
  margin-left: 10px;
  color: #6c757d;
}

.breadcrumb li:last-child {
  color: #6c757d;
}

.breadcrumb li:last-child::after {
```

```
  content: "";
}
//End of Code written without Assistance for this code extract
```

Back-up.css

```
//Start of Code written without Assistance for this code extract
.mh-wrapper
{
    word-wrap: break-word;
    cursor: auto;
    margin: 0;
    border: 0;
    font: inherit;
    display: block;
    background: none;
    padding: 0px;
    color: #fff;
}

.mh-content
{
    word-wrap: break-word;
    cursor: auto;
    color: #fff;
    font: inherit;
    vertical-align: baseline;
    overflow: hidden;
    background: rgba(255, 255, 255, 0.33);
    border-radius: 16px;
    box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
    backdrop-filter: blur(5.7px);
    border: 2px solid rgba(255, 255, 255, 0.8);
    padding: 25px;
    float: none;
    width: 80%;
    margin: 20px auto;
}

.post
{
    word-wrap: break-word;
    cursor: auto;
    color: #fff;
    margin: 0;
    padding: 0;
    border: 0;
    font: inherit;
    display: block;
}

.entry-header
{
    word-wrap: break-word;
    cursor: auto;
    color: #fff;
    margin: 0;
    padding: 0;
    border: 0;
```

```
    font: inherit;
    display: block;
    margin-bottom: 1.25rem;
}

.entry-title
{
    word-wrap: break-word;
    cursor: auto;
    margin: 0;
    padding: 0;
    border: 0;
    font: inherit;
    vertical-align: baseline;
    font-weight: 700;
    font-family: "Viking", "Open Sans", Helvetica, sans-serif;
    text-shadow: 3px 3px 10px black;
    line-height: 1.5;
    color: #fff!important;
    font-size: 1.5rem;
}

.entry-meta
{
    word-wrap: break-word;
    cursor: auto;
    margin: 0;
    border: 0;
    font: inherit;
    vertical-align: baseline;
    font-size: 0.8125rem;
    margin-top: 0.625rem;
    border-top: 1px dotted #ebebeb;
    border-bottom: 1px dotted #ebebeb;
    padding: 5px 0;
    color: #fff;
}

.entry-meta-date
{
    word-wrap: break-word;
    cursor: auto;
    color: #fff;
    margin: 0;
    padding: 0;
    border: 0;
    font: inherit;
    vertical-align: baseline;
    margin-right: 10px;
}

.entry-content
{
    word-wrap: break-word;
    cursor: auto;
    color: #fff;
    margin: 0;
    padding: 0;
```

```
    border: 0;
    font: inherit;
    counter-reset: footnotes;
    display: block;
}

.entry-thumbnail {
    display: block;
    max-width: 100%;
    height: auto;
    margin: 0 auto;
    text-align: center;
    border: 2px white;
}

.entry-thumbnail img {
    max-width: 70%;
    height: auto;
    border: 2px white;
}

p{
    word-wrap: break-word;
    cursor: auto;
    margin: 0;
    padding: 0;
    border: 0;
    font: inherit;
    vertical-align: baseline;
    margin-bottom: 1.25rem;
    color: #fff;
}

body {
    background-image: url(/assets/background.png);
    background-repeat: no-repeat;
    background-size: cover;
  }

  form {
    word-wrap: break-word;
    cursor: auto;
    color: #fff;
    margin: 0;
    padding: 25px;
    border: 2px solid rgba(255, 255, 255, 0.8);
    font: inherit;
    vertical-align: baseline;
    background: rgba(255, 255, 255, 0.33);
    border-radius: 16px;
    box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
    backdrop-filter: blur(5.7px);
    }

    form label {
    display: block;
    margin-bottom: 0.5rem;
    font-weight: bold;
```

```
    text-shadow: 3px 3px 10px black;
    }

    form input, form textarea {
    display: block;
    width: 100%;
    padding: 0.5rem;
    border: 1px solid rgba(255, 255, 255, 0.8);
    border-radius: 4px;
    background-color: rgba(255, 255, 255, 0.5);
    color: #333;
    margin-bottom: 1rem;
    }

    form button {
    display: block;
    width: 100%;
    padding: 0.5rem;
    border: none;
    border-radius: 4px;
    background-color: #007bff;
    color: #fff;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.3s ease;
    }

    form button:hover {
    background-color: #0056b3;
    }

    .Delete-Account {
        background-color: #ff0000; /* Red color */
        border: none;
        color: white;
        padding: 8px 16px; /* Smaller padding */
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 14px; /* Smaller font size */
        margin: 4px 2px; /* Smaller margin */
        cursor: pointer;
        border-radius: 4px; /* Slightly rounded corners */
      }

      .Delete-Account a {
        color: white;
        text-decoration: none;
      }

      .warning {
        font-size: 14px; /* Adjust the font size as needed */
        font-weight: bold; /* Make the text bold */
        color: #ff0000; /* Set the text color to red */
        background-color: #f8f8f8; /* Set a light background color */
        padding: 10px 15px; /* Add some padding around the text */
        border: 1px solid #e0e0e0; /* Add a light border */
        border-radius: 4px; /* Add some rounded corners */
```

```
        margin-top: 10px; /* Add some space above the message */
      }

      .return {
        display: inline-block;
        background-color: #007bff;
        color: #fff;
        border: none;
        border-radius: 4px;
        padding: 0.5rem 1rem;
        text-decoration: none;
        font-size: 1rem;
        font-weight: bold;
        cursor: pointer;
        transition: background-color 0.3s ease;
      }

      .return:hover {
        background-color: #0056b3;
      }

      .return a {
        color: #fff;
        text-decoration: none;
      }

//End of Code written without Assistance for this code extract
```

Change.css

```
//Start of Code written without Assistance for this code extract

.cute-button {
    display: inline-block;
    padding: 10px;
    background-color: pink;
    color: white;
    border: none;
    border-radius: 20px;
    font-family: 'Arial', sans-serif;
    font-size: 16px;
    cursor: pointer;
  }

  .cute-button:hover {
    background-color: hotpink;
  }

//End of Code written without Assistance for this code extract
```

Confirmation.css

```
/* confirmation.css */

//Start of Code written without Assistance for this code extract
body {
    font-family: 'Open Sans', sans-serif;
    background-color: #f7f7f7; /* light gray background */
  }
```

```
  h2 {
    font-size: 24px;
    color: #333; /* dark gray text */
    text-align: center;
    margin-bottom: 20px;
  }

  p {
    font-size: 18px;
    color: #666; /* medium gray text */
    margin-bottom: 10px;
  }

  /* Add some bubbly effects */
  h2, p {
    border-radius: 10px;
    padding: 10px;
    background-color: #fff; /* white background */
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* subtle shadow */
  }

  h2 {
    background-color: #ffe6e6; /* light pink background */
    color: #ff69b4; /* bright pink text */
  }

  p {
    background-color: #f2f2f2; /* light gray background */
  }

  h2:before {
    background-image: url('/assets/success.svg'); /* replace with
your SVG file name */
    background-size: 24px 24px; /* adjust the size to your liking */
    width: 24px;
    height: 24px;
    display: inline-block;
    margin-right: 10px;
    background-repeat: no-repeat;
    vertical-align: middle;
    color: #ff69b4; /* bright pink text */
    content: ""; /* empty content to avoid displaying the pseudo-
element's text */
  }

  p:before {
    display: inline-block;
    width: 1em;
    height: 1em;
    background-image: url('/assets/information.svg');
    background-size: contain;
    margin-right: 10px;
    font-size: 18px;
    color: #666;
    vertical-align: -0.125em;
    content: "";
  }
```

```
//End of Code written without Assistance for this code extract
```

Create-post.css

```
//Start of Code written without Assistance for this code extract
body {
  font-family: Arial, sans-serif;
  padding: 20px;
  background-image: url(/assets/background.png);
  background-repeat: no-repeat;
  background-size: cover;
}

  h1 {
    text-align: center;
    color: white;
  }

  form {
    width: 400px;
    margin: 0 auto;
  }

  label {
    display: block;
    margin-top: 10px;
  }

  input,
  textarea {
    width: 100%;
    padding: 5px;
    margin-top: 5px;
  }

  button[type="submit"] {
    display: block;
    width: 100%;
    padding: 10px;
    margin-top: 10px;
    background-color: #4CAF50;
    color: white;
    border: none;
    cursor: pointer;
  }

  button[type="submit"]:hover {
    background-color: #45a049;
  }
//End of Code written without Assistance for this code extract
```

Edit-Post.css

```
/* edit-post.css */

//Start of Code written without Assistance for this code extract

body {
    font-family: 'Helvetica Neue', Arial, sans-serif;
    background-color: #f8f8f8;
```

```
    color: #333;
    margin: 0;
    padding: 0;
  }

  h1 {
    text-align: center;
    margin-top: 2rem;
    color: #e91e63;
    font-size: 2.5rem;
  }

  form {
    max-width: 600px;
    margin: 2rem auto;
    background-color: #fff;
    padding: 2rem;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }

  label {
    display: block;
    font-size: 1.1rem;
    margin-bottom: 0.5rem;
    color: #e91e63;
  }

  input[type="text"],
  textarea {
    width: 100%;
    padding: 0.8rem;
    font-size: 1rem;
    border: 1px solid #ddd;
    border-radius: 4px;
    box-sizing: border-box;
    margin-bottom: 1.5rem;
  }

  textarea {
    height: 150px;
    resize: vertical;
  }

  button[type="submit"] {
    background-color: #e91e63;
    color: #fff;
    border: none;
    border-radius: 4px;
    padding: 0.8rem 1.5rem;
    font-size: 1.1rem;
    cursor: pointer;
    transition: background-color 0.3s ease;
  }

  button[type="submit"]:hover {
    background-color: #c2185b;
  }
```

```
  .postId input[type="text"]
  {
    background-color: grey;
  }

  .back-button{
    background-color: #e91e63;
    color: #fff;
    border: none;
    border-radius: 4px;
    padding: 0.8rem 1.5rem;
    font-size: 1.1rem;
    cursor: pointer;
    transition: background-color 0.3s ease;

  }

  .back-button a
  {
    text-decoration: none;
    color: white;
  }

//End of Code written without Assistance for this code extract
```

Home.css

```
//Start of Code written without Assistance for this code extract
body {
  font-family: 'Open Sans', sans-serif;
  background-image: url(/assets/background2.PNG);
  background-repeat: no-repeat;
  background-size: cover;
  }

  .time-container {
    background-color: #f9f9f9; /* Light gray background */
    border-radius: 10px; /* Rounded corners */
    padding: 20px; /* Add some padding */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Add a soft shadow */
  }

  h1 {
    display: inline-block;
    margin-right: 10px;
    font-size: 1.2em; /* slightly larger font size for emphasis */
    color: #333; /* a slightly darker gray for the heading */
  }

  h2
  {
    color:white;
  }


  #current-time {
    font-size: 24px; /* Larger font size for current time */
```

```
    color: #ff6347; /* Cute pink color for time */
  }

/* Published Articles Section */
.article-list {
  list-style-type: none;
  padding: 0;
  margin: 0;
}

.article-list li {
  border: 1px solid #ddd;
  border-radius: 5px;
  padding: 20px;
  margin-bottom: 20px;
  background-color: #f9f9f9;
}

.title {
  font-size: 24px;
  font-weight: bold;
  margin-bottom: 10px;
}

.subtitle {
  font-size: 18px;
  color: #666;
  margin-bottom: 10px;
}

.content {
  font-size: 16px;
  line-height: 1.5;
  margin-bottom: 20px;
}

.like, .dislike {
  display: inline-block;
  margin-right: 10px;
  background-color: #fff;
  border: 1px solid #ddd;
  border-radius: 5px;
  padding: 5px 10px;
  font-size: 14px;
  cursor: pointer;
}

.like img, .dislike img {
  height: 16px;
  vertical-align: middle;
  margin-right: 5px;
}

.Time-Created {
  font-size: 14px;
  color: #999;
  margin-top: 10px;
}
```

```
/* Share Button */
.Share-Button {
  display: inline-block;
  background-color: #6c5ce7;
  color: white;
  text-align: center;
  text-decoration: none;
  font-size: 14px;
  font-weight: 500;
  padding: 8px 16px;
  border: none;
  border-radius: 20px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  cursor: pointer;
  transition: transform 0.2s ease;
}

.Share-Button:hover {
  transform: translateY(-2px);
}

.Share-Button a {
  color: white;
  text-decoration: none;
}

/* View Post Button */
.View-post {
  display: inline-block;
  background-color: #00b894;
  color: white;
  text-align: center;
  text-decoration: none;
  font-size: 14px;
  font-weight: 500;
  padding: 8px 16px;
  border: none;
  border-radius: 20px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  cursor: pointer;
  transition: transform 0.2s ease;
}

.View-post:hover {
  transform: translateY(-2px);
}

.View-post a {
  color: white;
  text-decoration: none;
}

.no-posts {
  font-size: 18px;
  text-align: center;
  color: #999;
  padding: 20px;
}
```

```
  body
{
    background-color: beige;
}

.time-container p {
  margin: 5px 0; /* Add margin between paragraphs */
  font-family: 'Arial', sans-serif; /* Use a cute font */
  color: #333; /* Dark text color */
}

.back-to-top
{
  tab-size: 4;
  line-height: inherit;
  font-family: "Roboto", sans-serif;
  font-weight: normal;
  font-style: normal;
  font-size: 15px;
  box-sizing: border-box;
  border-width: 0;
  border-style: solid;
  border-color: #e5e7eb;
  margin: 0;
  padding: 0;
  text-decoration: none;
  position: fixed;
  bottom: 2rem;
  right: 2rem;
  left: auto;
  z-index: 999;
  height: 2.5rem;
  width: 2.5rem;
  align-items: center;
  justify-content: center;
  border-radius: 0.375rem;
  --tw-bg-opacity: 1;
  background-color: rgb(39 181 247 / var(--tw-bg-opacity));
  --tw-text-opacity: 1;
  color: rgb(255 255 255 / var(--tw-text-opacity));
  --tw-shadow: 0 4px 6px -1px rgb(0 0 0 / 0.1), 0 2px 4px -2px rgb(0
0 0 / 0.1);
  --tw-shadow-colored: 0 4px 6px -1px var(--tw-shadow-color),
  0 2px 4px -2px var(--tw-shadow-color);
  box-shadow: var(--tw-ring-offset-shadow, 0 0 #0000),
  var(--tw-ring-shadow, 0 0 #0000), var(--tw-shadow);
  transition-property: color, background-color, border-color,
  text-decoration-color, fill, stroke, opacity, box-shadow,
transform, filter,
  backdrop-filter, -webkit-text-decoration-color, -webkit-backdrop-
filter;
  transition-duration: 300ms;
  transition-timing-function: cubic-bezier(0.4, 0, 0.2, 1);
  display: flex;
}

.border-white
{
```

```
  -webkit-text-size-adjust: 100%;
  tab-size: 4;
  --thumb-width: 15;
  line-height: inherit;
  font-family: "Roboto", sans-serif;
  font-weight: normal;
  font-style: normal;
  font-size: 15px;
  --tw-bg-opacity: 1;
  --tw-text-opacity: 1;
  color: rgb(255 255 255 / var(--tw-text-opacity));
  box-sizing: border-box;
  border-width: 0;
  border-style: solid;
  --tw-translate-x: 0;
  --tw-translate-y: 0;
  --tw-skew-x: 0;
  --tw-skew-y: 0;
  --tw-scale-x: 1;
  --tw-scale-y: 1;
  --tw-pan-x: ;
  --tw-pan-y: ;
  --tw-pinch-zoom: ;
  --tw-scroll-snap-strictness: proximity;
  --tw-ordinal: ;
  --tw-slashed-zero: ;
  --tw-numeric-figure: ;
  --tw-numeric-spacing: ;
  --tw-numeric-fraction: ;
  --tw-ring-inset: ;
  --tw-ring-offset-width: 0px;
  --tw-ring-offset-color: #fff;
  --tw-ring-color: rgba(71, 197, 247, 0.788);
  --tw-ring-offset-shadow: 0 0 #0000;
  --tw-ring-shadow: 0 0 #0000;
  --tw-shadow: 0 0 #0000;
  --tw-shadow-colored: 0 0 #0000;
  --tw-blur: ;
  --tw-brightness: ;
  --tw-contrast: ;
  --tw-grayscale: ;
  --tw-hue-rotate: ;
  --tw-invert: ;
  --tw-saturate: ;
  --tw-sepia: ;
  --tw-drop-shadow: ;
  --tw-backdrop-blur: ;
  --tw-backdrop-brightness: ;
  --tw-backdrop-contrast: ;
  --tw-backdrop-grayscale: ;
  --tw-backdrop-hue-rotate: ;
  --tw-backdrop-invert: ;
  --tw-backdrop-opacity: ;
  --tw-backdrop-saturate: ;
  --tw-backdrop-sepia: ;
  margin: 0;
  padding: 0;
  display: inline-block;
```

```
  text-decoration: none;
  -webkit-transition: all 0.4s ease;
  margin-top: 6px;
  height: 0.75rem;
  width: 0.75rem;
  --tw-rotate: 45deg;
  transform: translate(var(--tw-translate-x), var(--tw-translate-y))
  rotate(var(--tw-rotate)) skewX(var(--tw-skew-x)) skewY(var(--tw-
skew-y))
  scaleX(var(--tw-scale-x)) scaleY(var(--tw-scale-y));
  border-top-width: 1px;
  border-left-width: 1px;
  --tw-border-opacity: 1;
  border-color: rgb(255 255 255 / var(--tw-border-opacity));
}

.blog-information {
  background-color: #f8f8f8;
  border: 1px solid #e5e5e5;
  border-radius: 5px;
  padding: 20px;
  text-align: center;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.blog-information h1 {
  font-size: 32px;
  color: #333;
  margin-bottom: 10px;
}

.blog-information h2 {
  font-size: 24px;
  color: #666;
  margin-top: 0;
}
//End of Code written without Assistance for this code extract
```

Login.css

```
//Start of Code written without Assistance for this code extract
.login-page {
  font-weight: 400;
  font-family: Open Sans,-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Helvetica Neue,Arial,sans-serif;
  font-size: 15px!important;
  color: rgba(255,255,255,.92);
  cursor: auto;
  margin: 10px;
  padding: 0;
  border: 1px solid #69697d;
  display: block;
  text-align: center;
  margin-left: auto!important;
  margin-right: auto!important;
  margin-bottom: 20px;
  max-width: 600px;
  z-index: 1001;
  background-color: rgba(128, 128, 128, 0.5); /* Update this line */
```

```
  }

.section-header
{
    font-family: Open Sans,-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Helvetica Neue,Arial,sans-serif;
    font-size: 15px!important;
    color: rgba(255,255,255,.92);
    cursor: auto;
    text-align: center;
    margin: 0;
    padding: 5px 10px;
    font-weight: 400!important;
    background: #20242a;
}

h2{
        --thumb-width: 15;
    font-family: Open Sans,-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Helvetica Neue,Arial,sans-serif;
    cursor: auto;
    text-align: center;
    margin: 0;
    padding: 0;
    font-size: 22px;
    line-height: 32px;
    font-weight: 400;
    color: #add8f5;
}

.section-body{
    --thumb-width: 15;
    font-weight: 400;
    font-family: Open Sans,-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Helvetica Neue,Arial,sans-serif;
    font-size: 15px!important;
    color: rgba(255,255,255,.92);
    cursor: auto;
    margin: 0;
    text-align: left;
    padding: 5px 10px 10px;
    background: #353b45;
}

.login-recovery
{
    font-weight: 400;
    font-family: Open Sans,-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Helvetica Neue,Arial,sans-serif;
    font-size: 15px!important;
    color: rgba(255,255,255,.92);
    cursor: auto;
    padding: 0;
    text-align: right;
    width: 170px;
    display: inline-block;
    margin: 0 10px 15px 0;
}
```

```
.textbox
{
    margin: 0;
    font-family: Open Sans,-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Helvetica Neue,Arial,sans-serif;
    font-size: 15px!important;
    outline: none;
    padding: 5px 7px;
    box-sizing: border-box;
    min-height: 30px;
    color: rgba(255,255,255,.92);
    background-color: rgba(0,0,0,.15)!important;
    border: 1px #69697d solid!important;
    border-radius: 7px;
}

.alignleft
{
        --thumb-width: 15;
    font-weight: 400;
    font-family: Open Sans,-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Helvetica Neue,Arial,sans-serif;
    font-size: 15px!important;
    color: rgba(255,255,255,.92);
    cursor: auto;
    margin: 0;
    padding: 0;
    text-align: left;
}

.button
{
        --thumb-width: 15;
    margin: 0;
    text-align: center;
    display: inline-block;
    cursor: pointer;
    text-decoration: none;
    vertical-align: bottom;
    font-family: -apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Helvetica Neue,Arial,sans-serif!important;
    font-size: 15px!important;
    line-height: 50px;
    padding: 0 7px 2px 6px;
    color: rgba(255,255,255,.92)!important;
    background: #434b5b;
    border: 1px solid #69697d;
    margin-left: 232px;
}

body {
  background-image: url(/assets/house.jpg);
  background-repeat: no-repeat;
  background-size: cover;
  }

.transparent-layer {
```

```
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    z-index: 1000;
}

.image-container {
    display: flex;
    justify-content: right;
    width: 70%;
}

.image-container img {
    width: 40%;
    height: auto;
}

#eyeIcon
{
    width: 20px;
    height: 20px;
}

.button-secondary
{
    background-color: beige;
    shape-margin: 1px;
}

.switch {
  position: relative;
  display: inline-block;
  width: 60px;
  height: 34px;
}

.switch input {
  display: none;
}

.slider {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #ccc;
  border-radius: 34px;
  cursor: pointer;
  transition:.4s;
}

.slider:before {
  position: absolute;
  content: "";
```

```
  height: 26px;
  width: 26px;
  left: 4px;
  bottom: 4px;
  background-color: white;
  border-radius:50%;
  transition:.4s;
}

input:checked +.slider {
  background-color: #2196F3;
}

input:checked +.slider:before {
  transform: translateX(26px);
}

.light-mode-btn {
  margin-left: 10px;
  cursor: pointer;
  transition: color 0.3s ease;
}

.light-mode-btn:hover {
  color: #2196F3;
}
//End of Code written without Assistance for this code extract
```

Main.css

```
//Start of Code written without Assistance for this code extract
.hero-area
{
    tab-size: 4;
    line-height: inherit;
    font-family: "Roboto", sans-serif;
    font-weight: normal;
    font-style: normal;
    color: #727272;
    font-size: 15px;
    cursor: auto;
    box-sizing: border-box;
    border-width: 0;
    border-style: solid;
    border-color: #e5e7eb;
    margin: 0;
    position: relative;
    background-color: #fff;
    background-image: url("/assets/ponybg.PNG");
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    padding: 110px 0 120px 0;
}

.container
{
    tab-size: 4;
    line-height: inherit;
```

```
    font-family: "Roboto", sans-serif;
    font-weight: normal;
    font-style: normal;
    color: #727272;
    font-size: 15px;
    cursor: auto;
    box-sizing: border-box;
    border-width: 0;
    border-style: solid;
    border-color: #e5e7eb;
    margin: 0;
    padding: 0;
    width: 100%;
    margin-right: auto;
    margin-left: auto;
    padding-right: 16px;
    padding-left: 16px;
    max-width: 720px;
}

.row
{
    -webkit-text-size-adjust: 100%;
    tab-size: 4;
    --thumb-width: 15;
    line-height: inherit;
    font-family: "Roboto", sans-serif;
    font-weight: normal;
    font-style: normal;
    color: #727272;
    font-size: 15px;
    cursor: auto;
    box-sizing: border-box;
    border-width: 0;
    border-style: solid;
    border-color: #e5e7eb;
    margin: 0;
    padding: 0;
}

.wow-animated
{
    -webkit-text-size-adjust: 100%;
    tab-size: 4;
    --thumb-width: 15;
    line-height: inherit;
    font-style: normal;
    cursor: auto;
    text-align: center;
    box-sizing: border-box;
    border-width: 0;
    border-style: solid;
    border-color: #e5e7eb;
    --tw-translate-x: 0;
    --tw-translate-y: 0;
    --tw-rotate: 0;
    --tw-skew-x: 0;
    --tw-skew-y: 0;
```

```
    --tw-scale-x: 1;
    --tw-scale-y: 1;
    --tw-pan-x: ;
    --tw-pan-y: ;
    --tw-pinch-zoom: ;
    --tw-scroll-snap-strictness: proximity;
    --tw-ordinal: ;
    --tw-slashed-zero: ;
    --tw-numeric-figure: ;
    --tw-numeric-spacing: ;
    --tw-numeric-fraction: ;
    --tw-ring-inset: ;
    --tw-ring-offset-width: 0px;
    --tw-ring-offset-color: #fff;
    --tw-ring-color: rgba(71, 197, 247, 0.788);
    --tw-ring-offset-shadow: 0 0 #0000;
    --tw-ring-shadow: 0 0 #0000;
    --tw-shadow: 0 0 #0000;
    --tw-shadow-colored: 0 0 #0000;
    --tw-blur: ;
    --tw-brightness: ;
    --tw-contrast: ;
    --tw-grayscale: ;
    --tw-hue-rotate: ;
    --tw-invert: ;
    --tw-saturate: ;
    --tw-sepia: ;
    --tw-drop-shadow: ;
    --tw-backdrop-blur: ;
    --tw-backdrop-brightness: ;
    --tw-backdrop-contrast: ;
    --tw-backdrop-grayscale: ;
    --tw-backdrop-hue-rotate: ;
    --tw-backdrop-invert: ;
    --tw-backdrop-opacity: ;
    --tw-backdrop-saturate: ;
    --tw-backdrop-sepia: ;
    margin: 0px;
    font-family: 'Barlow', sans-serif;
    animation-duration: 1s;
    animation-fill-mode: both;
    color: #fff;
    font-weight: 500;
    display: inline-block;
    background: #ffffff24;
    padding: 13px 22px;
    border-radius: 4px;
    box-shadow: 0px 0px 10px #00000005;
    font-size: 15px;
    margin-bottom: 12px;
    visibility: visible;
    animation-delay: 0.2s;
}
.wow-h2
{
    -webkit-text-size-adjust: 100%;
    tab-size: 4;
    --thumb-width: 15;
```

```
    font-style: normal;
    cursor: auto;
    text-align: center;
    box-sizing: border-box;
    border-width: 0;
    border-style: solid;
    border-color: #e5e7eb;
    --tw-translate-x: 0;
    --tw-translate-y: 0;
    --tw-rotate: 0;
    --tw-skew-x: 0;
    --tw-skew-y: 0;
    --tw-scale-x: 1;
    --tw-scale-y: 1;
    --tw-pan-x: ;
    --tw-pan-y: ;
    --tw-pinch-zoom: ;
    --tw-scroll-snap-strictness: proximity;
    --tw-ordinal: ;
    --tw-slashed-zero: ;
    --tw-numeric-figure: ;
    --tw-numeric-spacing: ;
    --tw-numeric-fraction: ;
    --tw-ring-inset: ;
    --tw-ring-offset-width: 0px;
    --tw-ring-offset-color: #fff;
    --tw-ring-color: rgba(71, 197, 247, 0.788);
    --tw-ring-offset-shadow: 0 0 #0000;
    --tw-ring-shadow: 0 0 #0000;
    --tw-shadow: 0 0 #0000;
    --tw-shadow-colored: 0 0 #0000;
    --tw-blur: ;
    --tw-brightness: ;
    --tw-contrast: ;
    --tw-grayscale: ;
    --tw-hue-rotate: ;
    --tw-invert: ;
    --tw-saturate: ;
    --tw-sepia: ;
    --tw-drop-shadow: ;
    --tw-backdrop-blur: ;
    --tw-backdrop-brightness: ;
    --tw-backdrop-contrast: ;
    --tw-backdrop-grayscale: ;
    --tw-backdrop-hue-rotate: ;
    --tw-backdrop-invert: ;
    --tw-backdrop-opacity: ;
    --tw-backdrop-saturate: ;
    --tw-backdrop-sepia: ;
    padding: 0;
    font-family: 'Barlow', sans-serif;
    animation-name: fadeInUp;
    font-weight: 700;
    color: #fff;
    margin: 0px 0 18px;
    text-transform: capitalize;
    overflow: hidden;
    font-size: 30px;
```

```
    line-height: 40px;
    visibility: visible;
    animation-delay: 0.4s;
}

.wow
{
    -webkit-text-size-adjust: 100%;
    tab-size: 4;
    --thumb-width: 15;
    font-family: "Roboto", sans-serif;
    font-style: normal;
    cursor: auto;
    text-align: center;
    box-sizing: border-box;
    border-width: 0;
    border-style: solid;
    border-color: #e5e7eb;
    --tw-translate-x: 0;
    --tw-translate-y: 0;
    --tw-rotate: 0;
    --tw-skew-x: 0;
    --tw-skew-y: 0;
    --tw-scale-x: 1;
    --tw-scale-y: 1;
    --tw-pan-x: ;
    --tw-pan-y: ;
    --tw-pinch-zoom: ;
    --tw-scroll-snap-strictness: proximity;
    --tw-ordinal: ;
    --tw-slashed-zero: ;
    --tw-numeric-figure: ;
    --tw-numeric-spacing: ;
    --tw-numeric-fraction: ;
    --tw-ring-inset: ;
    --tw-ring-offset-width: 0px;
    --tw-ring-offset-color: #fff;
    --tw-ring-color: rgba(71, 197, 247, 0.788);
    --tw-ring-offset-shadow: 0 0 #0000;
    --tw-ring-shadow: 0 0 #0000;
    --tw-shadow: 0 0 #0000;
    --tw-shadow-colored: 0 0 #0000;
    --tw-blur: ;
    --tw-brightness: ;
    --tw-contrast: ;
    --tw-grayscale: ;
    --tw-hue-rotate: ;
    --tw-invert: ;
    --tw-saturate: ;
    --tw-sepia: ;
    --tw-drop-shadow: ;
    --tw-backdrop-blur: ;
    --tw-backdrop-brightness: ;
    --tw-backdrop-contrast: ;
    --tw-backdrop-grayscale: ;
    --tw-backdrop-hue-rotate: ;
    --tw-backdrop-invert: ;
    --tw-backdrop-opacity: ;
```

```
    --tw-backdrop-saturate: ;
    --tw-backdrop-sepia: ;
    padding: 0;
    animation-name: fadeInUp;
    font-weight: 400;
    color: #fff;
    margin: 18px 0;
    font-size: 15px;
    line-height: 26px;
    visibility: visible;
    animation-delay: 0.6s;
}

.button-wow
{
    -webkit-text-size-adjust: 100%;
    tab-size: 4;
    --thumb-width: 15;
    line-height: inherit;
    font-family: "Roboto", sans-serif;
    font-weight: normal;
    font-style: normal;
    color: #727272;
    font-size: 15px;
    cursor: auto;
    text-align: center;
    box-sizing: border-box;
    border-width: 0;
    border-style: solid;
    border-color: #e5e7eb;
    --tw-translate-x: 0;
    --tw-translate-y: 0;
    --tw-rotate: 0;
    --tw-skew-x: 0;
    --tw-skew-y: 0;
    --tw-scale-x: 1;
    --tw-scale-y: 1;
    --tw-pan-x: ;
    --tw-pan-y: ;
    --tw-pinch-zoom: ;
    --tw-scroll-snap-strictness: proximity;
    --tw-ordinal: ;
    --tw-slashed-zero: ;
    --tw-numeric-figure: ;
    --tw-numeric-spacing: ;
    --tw-numeric-fraction: ;
    --tw-ring-inset: ;
    --tw-ring-offset-width: 0px;
    --tw-ring-offset-color: #fff;
    --tw-ring-color: rgba(71, 197, 247, 0.788);
    --tw-ring-offset-shadow: 0 0 #0000;
    --tw-ring-shadow: 0 0 #0000;
    --tw-shadow: 0 0 #0000;
    --tw-shadow-colored: 0 0 #0000;
    --tw-blur: ;
    --tw-brightness: ;
    --tw-contrast: ;
    --tw-grayscale: ;
```

```
    --tw-hue-rotate: ;
    --tw-invert: ;
    --tw-saturate: ;
    --tw-sepia: ;
    --tw-drop-shadow: ;
    --tw-backdrop-blur: ;
    --tw-backdrop-brightness: ;
    --tw-backdrop-contrast: ;
    --tw-backdrop-grayscale: ;
    --tw-backdrop-hue-rotate: ;
    --tw-backdrop-invert: ;
    --tw-backdrop-opacity: ;
    --tw-backdrop-saturate: ;
    --tw-backdrop-sepia: ;
    margin: 0;
    padding: 0;
    animation-name: fadeInUp;
    margin-left: 0 !important;
    margin-top: 10px !important;
    width: 100%;
    visibility: visible;
    animation-delay: 0.8s;
}

.btn
{
    -webkit-text-size-adjust: 100%;
    tab-size: 4;
    --thumb-width: 15;
    line-height: inherit;
    font-family: "Roboto", sans-serif;
    font-style: normal;
    text-align: center;
    visibility: visible;
    box-sizing: border-box;
    --tw-translate-x: 0;
    --tw-translate-y: 0;
    --tw-rotate: 0;
    --tw-skew-x: 0;
    --tw-skew-y: 0;
    --tw-scale-x: 1;
    --tw-scale-y: 1;
    --tw-pan-x: ;
    --tw-pan-y: ;
    --tw-pinch-zoom: ;
    --tw-scroll-snap-strictness: proximity;
    --tw-ordinal: ;
    --tw-slashed-zero: ;
    --tw-numeric-figure: ;
    --tw-numeric-spacing: ;
    --tw-numeric-fraction: ;
    --tw-ring-inset: ;
    --tw-ring-offset-width: 0px;
    --tw-ring-offset-color: #fff;
    --tw-ring-color: rgba(71, 197, 247, 0.788);
    --tw-ring-offset-shadow: 0 0 #0000;
    --tw-ring-shadow: 0 0 #0000;
    --tw-shadow: 0 0 #0000;
```

```
    --tw-shadow-colored: 0 0 #0000;
    --tw-blur: ;
    --tw-brightness: ;
    --tw-contrast: ;
    --tw-grayscale: ;
    --tw-hue-rotate: ;
    --tw-invert: ;
    --tw-saturate: ;
    --tw-sepia: ;
    --tw-drop-shadow: ;
    --tw-backdrop-blur: ;
    --tw-backdrop-brightness: ;
    --tw-backdrop-contrast: ;
    --tw-backdrop-grayscale: ;
    --tw-backdrop-hue-rotate: ;
    --tw-backdrop-invert: ;
    --tw-backdrop-opacity: ;
    --tw-backdrop-saturate: ;
    --tw-backdrop-sepia: ;
    text-decoration: none;
    display: inline-block;
    text-transform: capitalize;
    border: none;
    transition: all 0.3s ease;
    border-radius: 4px;
    position: relative;
    z-index: 1;
    overflow: hidden;
    box-shadow: 0px 4px 7px #00000021;
    padding: 14px 25px;
    font-size: 14px;
    font-weight: 500;
    margin: 0;
    background-color: #27b5f7;
    color: #fff;
}
//End of Code written without Assistance for this code extract
```

Nav.css

```
//Start of Code written without Assistance for this code extract
.promo-bar{
    text-shadow: 1px 1px 1px rgba(0,0,0,.004);
    line-height: 1.5em;
    cursor: auto;
    box-sizing: border-box;
    font-family: var(--font--accent--family);
    font-weight: var(--font--accent--weight);
    font-style: var(--font--accent--style);
    letter-spacing: 1px;
    font-size: 13px;
    min-height: 33px;
    width: 100%;
    transition: .3s;
    padding: 7px 10px;
    position: relative;
    overflow: hidden;
    text-align: center;
    text-transform: none;
```

```
    color: var(--color--alternative);
    background-color: rgb(247, 247, 247);
}

.link
{
    text-shadow: 1px 1px 1px rgba(0,0,0,.004);
    line-height: 1.5em;
    font-family: var(--font--accent--family);
    font-weight: var(--font--accent--weight);
    font-style: var(--font--accent--style);
    letter-spacing: 1px;
    font-size: 13px;
    text-align: center;
    text-transform: none;
    box-sizing: border-box;
    text-decoration: none;
    background-color: transparent;
    transition: color .2s ease;
    color: var(--color--alternative);
}

.text
{
        -webkit-text-size-adjust: 100%;
    visibility: inherit;
    --font--section-heading--size: 20px;
    --font--block-heading--size: 16px;
    --font--heading--uppercase: normal;
    --font--paragraph--size: 14px;
    --font--heading--family: Futura, sans-serif;
    --font--heading--weight: 400;
    --font--heading--normal-weight: 400;
    --font--heading--style: normal;
    --font--accent--family: Futura, sans-serif;
    --font--accent--weight: 400;
    --font--accent--style: normal;
    --font--paragraph--family: Futura, sans-serif;
    --font--paragraph--weight: 400;
    --font--paragraph--style: normal;
    --font--bolder-paragraph--weight: 700;
    --image--loading-animation:
url('//www.ginleestudio.com/cdn/shop/t/17/assets/AjaxLoader.gif?v=664
3103100573399637161103752');
    --image--grabbing-icon:
url('//www.ginleestudio.com/cdn/shop/t/17/assets/grabbing.png?v=16299
554155157915417161103755');
    --image--popup:
url('//www.ginleestudio.com/cdn/shop/files/GINLEE7858_362a022f-0e1b-
4252-bc7a-fe71e48a3b61_1200x.jpg?v=1706273686');
    --color--accent: #f2efeb;
    --color-text: #212121;
    --color-page-bg: #ffffff;
    --color-panel-bg: #ffffff;
    --color-border: #ebebeb;
    --color-error: #f66767;
    --color-button: #ffffff;
    --color-button-bg: #cacaca;
```

```
    --color--body--light: #6e6e6e;
    --color--alternative: #000000;
    --color-header: #ffffff;
    --color-header-bg: #c0c0c0;
    --color-menubar: #ffffff;
    --color-cart: #ffffff;
    --color-footer: #fefefe;
    --color-footer-bg: #c0c0c0;
    --color-slider-caption: #272727;
    --color-slider-caption-bg: #ffffff;
    --color-slider-button: #f7f7f7;
    --color-slider-button-bg: #313131;
    --color-slider-button-hover-bg: #444444;
    --color-slider-nav: #444444;
    --color-product-slider-bg: #ffffff;
    --color-featured-promo-bg: #f7f7f7;
    --color-social-feed-bg: #ffffff;
    --color-grid-sale: #ffffff;
    --color-grid-sale-bg: #313131;
    --color-grid-sold-out: #ffffff;
    --color-grid-sold-out-bg: #cccccc;
    --color-tabs-accordions: #ebebeb;
    --swym-remind-cta-bg-color: #00a65a;
    --swym-remind-cta-text-color: white;
    --swym-remind-cta-bg-color-v2: #000000;
    --swym-remind-cta-text-color-v2: white;
    --swym-subscribe-success-bg-color: #AEE9D1;
    --swym-subscribe-success-text-color: #202223;
    --swym-recently-viewed-pointer-pos: 92%;
    --thumb-width: 15;
    -webkit-font-smoothing: antialiased;
    text-shadow: 1px 1px 1px rgba(0,0,0,.004);
    line-height: 1.5em;
    font-family: var(--font--accent--family);
    font-weight: var(--font--accent--weight);
    font-style: var(--font--accent--style);
    letter-spacing: 1px;
    font-size: 13px;
    text-align: center;
    text-transform: none;
    color: var(--color--alternative);
    box-sizing: border-box;
    padding: 3px 28px 0;
    display: block;
}

.close
{
        -webkit-text-size-adjust: 100%;
    visibility: inherit;
    --font--section-heading--size: 20px;
    --font--block-heading--size: 16px;
    --font--heading--uppercase: normal;
    --font--paragraph--size: 14px;
    --font--heading--family: Futura, sans-serif;
    --font--heading--weight: 400;
    --font--heading--normal-weight: 400;
    --font--heading--style: normal;
```

```
    --font--accent--family: Futura, sans-serif;
    --font--accent--weight: 400;
    --font--accent--style: normal;
    --font--paragraph--family: Futura, sans-serif;
    --font--paragraph--weight: 400;
    --font--paragraph--style: normal;
    --font--bolder-paragraph--weight: 700;
    --image--loading-animation:
url('//www.ginleestudio.com/cdn/shop/t/17/assets/AjaxLoader.gif?v=664
3103100573399637161103752');
    --image--grabbing-icon:
url('//www.ginleestudio.com/cdn/shop/t/17/assets/grabbing.png?v=16299
554155157915417161103755');
    --image--popup:
url('//www.ginleestudio.com/cdn/shop/files/GINLEE7858_362a022f-0e1b-
4252-bc7a-fe71e48a3b61_1200x.jpg?v=1706273686');
    --color--accent: #f2efeb;
    --color-text: #212121;
    --color-page-bg: #ffffff;
    --color-panel-bg: #ffffff;
    --color-border: #ebebeb;
    --color-error: #f66767;
    --color-button: #ffffff;
    --color-button-bg: #cacaca;
    --color--body--light: #6e6e6e;
    --color--alternative: #000000;
    --color-header: #ffffff;
    --color-header-bg: #c0c0c0;
    --color-menubar: #ffffff;
    --color-cart: #ffffff;
    --color-footer: #fefefe;
    --color-footer-bg: #c0c0c0;
    --color-slider-caption: #272727;
    --color-slider-caption-bg: #ffffff;
    --color-slider-button: #f7f7f7;
    --color-slider-button-bg: #313131;
    --color-slider-button-hover-bg: #444444;
    --color-slider-nav: #444444;
    --color-product-slider-bg: #ffffff;
    --color-featured-promo-bg: #f7f7f7;
    --color-social-feed-bg: #ffffff;
    --color-grid-sale: #ffffff;
    --color-grid-sale-bg: #313131;
    --color-grid-sold-out: #ffffff;
    --color-grid-sold-out-bg: #cccccc;
    --color-tabs-accordions: #ebebeb;
    --swym-remind-cta-bg-color: #00a65a;
    --swym-remind-cta-text-color: white;
    --swym-remind-cta-bg-color-v2: #000000;
    --swym-remind-cta-text-color-v2: white;
    --swym-subscribe-success-bg-color: #AEE9D1;
    --swym-subscribe-success-text-color: #202223;
    --swym-recently-viewed-pointer-pos: 92%;
    --thumb-width: 15;
    -webkit-font-smoothing: antialiased;
    text-shadow: 1px 1px 1px rgba(0,0,0,.004);
    line-height: 1.5em;
    font-family: var(--font--accent--family);
```

```
    font-weight: var(--font--accent--weight);
    font-style: var(--font--accent--style);
    letter-spacing: 1px;
    font-size: 13px;
    text-align: center;
    text-transform: none;
    box-sizing: border-box;
    cursor: pointer;
    border: none;
    margin: 0;
    padding: 0;
    position: absolute;
    right: 10px;
    top: 50%;
    transform: translateY(-50%);
    color: var(--color--alternative);
}

body {
  margin: 0;
  padding: 0;
  font-family: Arial, sans-serif;
}

.navbar {
  background-color: #c65b74;
  height: 60px;
  display: flex;
  justify-content: center;
  align-items: center;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.navbar a {
  color: whitesmoke;
  text-decoration: none;
  font-size: 18px;
  margin: 0 10px;
  padding: 10px;
  transition: all 0.3s ease;
}

.navbar a:hover {
  background-color: pink;
}

.navbar a.active {
  background-color: #333;
  color: #fff;
}

.search-container {
  position: absolute;
  top: 11.5%;
  right: 0;
  transform: translateY(-50%);
  padding: 0 10px;
  display: flex;
```

```
  align-items: center;
  justify-content: center;
  height: 20%;
  cursor: pointer;
}

.search-container img {
  width: 20px;
  height: 20px;
  fill: white;
}

.search-container:hover img {
  fill: black;
}

:root {
  cursor: url("scripts/cute-kuromi-and-my-melody-cursor.png"),
default;
}

a:hover,
button:hover,
input[type="submit"]:hover,
input[type="button"]:hover {
  cursor: url('scripts/cute-kuromi-and-my-melody-pointer.png'),
pointer;
}
//End of Code written without Assistance for this code extract
```

Register.css

```
//Start of Code written without Assistance for this code extract
.mh-wrapper {
  word-wrap: break-word;
  cursor: auto;
  margin: 0;
  border: 0;
  font: inherit;
  display: block;
  background: none;
  padding: 0px;
  color: #fff;
}

.mh-content {
  word-wrap: break-word;
  cursor: auto;
  color: #fff;
  font: inherit;
  vertical-align: baseline;
  overflow: hidden;
  background: rgba(255, 255, 255, 0.33);
  border-radius: 16px;
  box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
  backdrop-filter: blur(5.7px);
  border: 2px solid rgba(255, 255, 255, 0.8);
  padding: 25px;
  float: none;
```

```
  width: 80%;
  margin: 20px auto;
}

.post {
  word-wrap: break-word;
  cursor: auto;
  color: #fff;
  margin: 0;
  padding: 0;
  border: 0;
  font: inherit;
  display: block;
}

.entry-header {
  word-wrap: break-word;
  cursor: auto;
  color: #fff;
  margin: 0;
  padding: 0;
  border: 0;
  font: inherit;
  display: block;
  margin-bottom: 1.25rem;
}

.entry-title {
  word-wrap: break-word;
  cursor: auto;
  margin: 0;
  padding: 0;
  border: 0;
  font: inherit;
  vertical-align: baseline;
  font-weight: 700;
  font-family: "Viking", "Open Sans", Helvetica, sans-serif;
  text-shadow: 3px 3px 10px black;
  line-height: 1.5;
  color: #fff !important;
  font-size: 1.5rem;
}

.entry-meta {
  word-wrap: break-word;
  cursor: auto;
  margin: 0;
  border: 0;
  font: inherit;
  vertical-align: baseline;
  font-size: 0.8125rem;
  margin-top: 0.625rem;
  border-top: 1px dotted #ebebeb;
  border-bottom: 1px dotted #ebebeb;
  padding: 5px 0;
  color: #fff;
}
```

```
.entry-meta-date {
  word-wrap: break-word;
  cursor: auto;
  color: #fff;
  margin: 0;
  padding: 0;
  border: 0;
  font: inherit;
  vertical-align: baseline;
  margin-right: 10px;
}

.entry-content {
  word-wrap: break-word;
  cursor: auto;
  color: #fff;
  margin: 0;
  padding: 0;
  border: 0;
  font: inherit;
  counter-reset: footnotes;
  display: block;
}

.entry-thumbnail {
  display: block;
  max-width: 100%;
  height: auto;
  margin: 0 auto;
  text-align: center;
  border: 2px white;
}

.entry-thumbnail img {
  max-width: 70%;
  height: auto;
  border: 2px white;
}

p {
  word-wrap: break-word;
  cursor: auto;
  margin: 0;
  padding: 0;
  border: 0;
  font: inherit;
  vertical-align: baseline;
  margin-bottom: 1.25rem;
  color: #fff;
}

body {
  background-image: url(/assets/background.png);
  background-repeat: no-repeat;
  background-size: cover;
}

form {
```

```
  word-wrap: break-word;
  cursor: auto;
  color: #fff;
  margin: 0;
  padding: 25px;
  border: 2px solid rgba(255, 255, 255, 0.8);
  font: inherit;
  vertical-align: baseline;
  background: rgba(255, 255, 255, 0.33);
  border-radius: 16px;
  box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);
  backdrop-filter: blur(5.7px);
}

form label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: bold;
  text-shadow: 3px 3px 10px black;
}

form input, form textarea {
  display: block;
  width: 100%;
  padding: 0.5rem;
  border: 1px solid rgba(255, 255, 255, 0.8);
  border-radius: 4px;
  background-color: rgba(255, 255, 255, 0.5);
  color: #333;
  margin-bottom: 1rem;
}

form button {
  display: block;
  width: 100%;
  padding: 0.5rem;
  border: none;
  border-radius: 4px;
  background-color: #007bff;
  color: #fff;
  font-weight: bold;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

form button:hover {
  background-color: #0056b3;
}
.Delete-Account {
  background-color: #ff0000; /* Red color */
  border: none;
  color: white;
  padding: 8px 16px; /* Smaller padding */
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 14px; /* Smaller font size */
  margin: 4px 2px; /* Smaller margin */
```

```
  cursor: pointer;
  border-radius: 4px; /* Slightly rounded corners */
}

.Delete-Account a {
  color: white;
  text-decoration: none;
}

.warning {
  font-size: 14px; /* Adjust the font size as needed */
  font-weight: bold; /* Make the text bold */
  color: #ff0000; /* Set the text color to red */
  background-color: #f8f8f8; /* Set a light background color */
  padding: 10px 15px; /* Add some padding around the text */
  border: 1px solid #e0e0e0; /* Add a light border */
  border-radius: 4px; /* Add some rounded corners */
  margin-top: 10px; /* Add some space above the message */
}

.return {
  display: inline-block;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 4px;
  padding: 0.5rem 1rem;
  text-decoration: none;
  font-size: 1rem;
  font-weight: bold;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.return:hover {
  background-color: #0056b3;
}

.return a {
  color: #fff;
  text-decoration: none;
}

/* Style for the header title container */
.header-title-container {
  background-color: rgba(255, 255, 255, 0.7); /* Translucent white
background */
  padding: 20px; /* Padding around the container */
  border-radius: 12px; /* Rounded corners */
  text-align: left; /* Align the title to the left */
  margin: 20px auto; /* Center the container horizontally */
  width: fit-content; /* Adjust width to fit the content */
}

/* Style for the header title */
.header-title {
  font-family: "Open Sans", Helvetica, Arial, sans-serif; /* Match
the font style */
```

```
  font-size: 2rem; /* Adjust size for prominence */
  font-weight: 700; /* Bold font for emphasis */
  color: #333333; /* Dark gray color for contrast */
  text-shadow: 2px 2px 10px rgba(0, 0, 0, 0.5); /* Subtle shadow for
depth */
}

.button {
  background-color: #6cc070;
  color: white;
  padding: 14px 28px;
  font-size: 18px;
  font-weight: bold;
  border: none;
  border-radius: 20px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
  cursor: pointer;
  transition: background-color 0.3s, box-shadow 0.3s;
}

.button:hover {
  background-color: #54b558;
  box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
}

.login-prompt {
  font-size: 16px;
  color: #333;
  margin-top: 20px;
  text-align: left;
}

.login-prompt a {
  color: #ff6347;
  text-decoration: underline;
  transition: color 0.3s ease;
}

.login-prompt a:hover {
  color: #e04c31;
  text-decoration: none;
}

.error-message {
  color: #ff0000;
  font-weight: bold;
  font-size: 14px;
  margin-top: 5px;
}

//End of Code written without Assistance for this code extract
```

Settings.css

```
//Start of Code written without Assistance for this code extract
body {
      font-family: Arial, sans-serif;
      background-image: url(/assets/background.jpg);
}
```

```
.container {
      max-width: 600px;
      margin: 40px auto;
      padding: 20px;
      background-color: #fff;
      border: 1px solid #ddd;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
      color: #333;
      margin-top: 0;
}

label {
      display: block;
      margin-bottom: 10px;
}

input[type="text"] {
      width: 80%;
      padding: 10px;
      margin-bottom: 20px;
      border: 1px solid #ccc;
      border-radius: 5px;
}

button[type="submit"] {
      background-color: #4CAF50;
      color: #fff;
      padding: 10px 20px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
}

button[type="submit"]:hover {
      background-color: #3e8e41;
}

.back-button {
      background-color: #4CAF50;
      color: #fff;
      padding: 10px 20px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      margin-top: 20px;
}

.back-button:hover {
      background-color: #3e8e41;
}

.back-button a
{
```

```
      text-decoration: none;
}
//End of Code written without Assistance for this code extract
```

Style.css

```
//Start of Code written without Assistance for this code extract
.site-footer
{
font-family: "Poppins", "Open Sans", system-ui, -apple-system, "Segoe
UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", "Liberation Sans",
sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol",
"Noto Color Emoji";;
font-size: 1rem;
font-weight: 400;
line-height: 1.5;
-webkit-text-size-adjust: 100%;
-webkit-tap-highlight-color: rgba(52,58,64,0);
cursor: auto;
box-sizing: border-box;
padding: 1rem 0;
color: #fff;
text-align: center;
background: #17173e;
z-index: 10;
flex-shrink: 0;
}

h1
{
  cursor: auto;
  color: #fff;
  text-align: center;
  box-sizing: border-box;
  margin-top: 0;
  margin-bottom: .5rem;
  font-family: "Poppins", "Open Sans", system-ui, -apple-system,
"Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", "Liberation
Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI
Symbol", "Noto Color Emoji";;
  font-weight: 700;
  line-height: 1.2;
  font-size: calc(1.375rem + 1.5vw);
}

.mb-0
{
  cursor: auto;
  color: #fff;
  text-align: center;
  box-sizing: border-box;
  margin-top: 0;
  margin-bottom: 0;
}

.social-links
{
  font-family: var(--bs-font-sans-serif);
  font-size: 1rem;
```

```
  font-weight: 400;
  line-height: 1.5;
  -webkit-text-size-adjust: 100%;
  -webkit-tap-highlight-color: rgba(52,58,64,0);
  cursor: auto;
  color: #fff;
  text-align: center;
  box-sizing: border-box;
  margin-bottom: .5rem;
}
//End of Code written without Assistance for this code extract
```

User-nav.css

```
//Start of Code written without Assistance for this code extract

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

/* Body styles */
body {
  font-family: Arial, sans-serif;
  background-color: #f7e9f6;
}

/* Header styles */
header {
  background-color: #ffdde1;
  padding: 10px;
}

/* Navigation styles */
nav ul {
  list-style-type: none;
  text-align: center;
}

nav ul li {
  display: inline-block;
  margin: 10px;
}

nav ul li a {
  text-decoration: none;
  color: #61185f;
  font-weight: bold;
  font-size: 18px;
  padding: 5px 10px;
  border-radius: 20px;
  background-color: #ffe5ea;
}

nav ul li a:hover {
  background-color: #ffced6;
}
//End of Code written without Assistance for this code extract
```

View-Post.css

```
//Start of Code written without Assistance for this code extract

body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  color: #333;
  margin: 0;
  padding: 0;
}

/* Post Container */
.post-container {
  max-width: 800px;
  margin: 50px auto;
  padding: 20px;
  background-color: #fff;
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
  border-radius: 8px;
}

/* Post Title */
.post-title {
  font-size: 2.5em;
  color: #333;
  margin-bottom: 10px;
  text-align: center;
}

/* Post Subtitle */
.post-subtitle {
  font-size: 1.5em;
  color: #666;
  margin-bottom: 20px;
  text-align: center;
}

/* Post Content */
.post-content {
  font-size: 1.2em;
  line-height: 1.6;
  margin-bottom: 20px;
  color: #444;
}

/* Post Stats */
.post-stats {
  display: flex;
  justify-content: space-around;
  margin: 20px 0;
}

.post-stats p {
  font-size: 1.2em;
  color: #666;
}
```

```
/* Like and Dislike Forms */
form {
  display: inline-block;
  margin-right: 10px;
}

form button {
  background-color: #007bff;
  color: white;
  border: none;
  padding: 10px 20px;
  font-size: 1em;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

form button:hover {
  background-color: #0056b3;
}

/* Comments Container */
.comments-container {
  max-width: 800px;
  margin: 20px auto;
  background-color: #fff;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}

.comments-container h2 {
  margin-bottom: 20px;
  color: #333;
  text-align: center;
}

.comments-list {
  list-style-type: none;
  padding: 0;
}

.comment-item {
  padding: 10px;
  border-bottom: 1px solid #ddd;
}

.comment-item:last-child {
  border-bottom: none;
}

.comment-date {
  font-size: 0.9em;
  color: #999;
}

.comment-text {
  font-size: 1.1em;
```

```
  color: #444;
  margin-bottom: 10px;
}

/* Comment Form */
.comment-form {
  display: flex;
  flex-direction: column;
  margin-top: 20px;
  max-width: 800px;
  margin: 20px auto;
}

.comment-form input[type="text"] {
  padding: 10px;
  margin-bottom: 10px;
  border-radius: 5px;
  border: 1px solid #ddd;
  font-size: 1em;
}

.comment-form button {
  align-self: flex-end;
  background-color: #28a745;
  color: white;
  border: none;
  padding: 10px 20px;
  font-size: 1em;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.comment-form button:hover {
  background-color: #218838;
}

/* Return Button */
.return {
  display: block;
  text-align: center;
  margin: 20px auto;
  background-color: #17a2b8;
  color: white;
  border: none;
  padding: 10px 20px;
  font-size: 1em;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.return a {
  color: white;
  text-decoration: none;
}

.return:hover {
```

```
  background-color: #138496;
}

.return a:hover {
  color: white;
}

/* Responsive Design */
@media (max-width: 768px) {
  .post-container, .comments-container, .comment-form {
    margin: 20px;
    padding: 15px;
  }

  .post-title {
    font-size: 2em;
  }

  .post-subtitle {
    font-size: 1.3em;
  }

  .post-content {
    font-size: 1em;
  }

  .post-stats p {
    font-size: 1em;
  }

  form button, .comment-form button, .return {
    font-size: 0.9em;
    padding: 8px 15px;
  }

  .comment-form input[type="text"] {
    font-size: 0.9em;
  }
}
//End of Code written without Assistance for this code extract
```

Author-Nav.ejs

```
<!DOCTYPE html>
<html>
<head>
 <link rel="stylesheet" type="text/css" href="/css/user-nav.css">
</head>
<body>
 <header>
  <nav>
   <ul>
    <li><a href="/author/author-settings">Settings</a></li>
    <li><a href="/author/create-post">Create Posts</a></li>
    <li><a href="/author/author-home">Home</a></li>
    <li><a href ="/Dashboard">Dashboard</a></li>
    <li><a href="/common/user-profile">Profile</a></li>
```

```
      <li><a href="/common/logout">Logout</a></li>
    </ul>
   </nav>
  </header>
</body>
</html>
```

Footer.ejs

```
<html>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<head>
<link rel="stylesheet" href="/css/style.css">
</head>
<footer class="site-footer"><h1>Check out below!</h1><p class="mb-0"></p><div
class="social-links">
</a></div><p>Created by a student of UOL Computer Science. Copyright all belongs to me @
2024.</p>
</footer>
</html>
```

Header.ejs

```
<html>
<head>
<link rel="stylesheet" href="/css/nav.css">
</head>
<div class="navbar">

  <a href="/DashBoard" class="active">Home</a>
  <a href="/about">About Me</a>
  <a href="/"> Main Page</a>
  <a href="/common/user-profile">Profile</a>
  <a href="/common/logout">Log Out</a>
</div>
</html>
```

Metadata.ejs

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
```

About.ejs

```
<html>
<head>
<link rel="stylesheet" href="/css/about.css">
<title> <%= title %> </title>
<%- include('./partials/header'); %>
</head>
<body>
<div id="content" class="page-wrap">
                    <div class="content-wrapper">
```

```
                                <div class="row">
          <div id="primary" class="fp-content-area">
                    <main id="main" class="site-main" role="main">
        <div class="text-container">
<div class="entry-content">

<h1 class="wp-block-heading"><strong>About me (Developer of CuteBlog)</strong></h1>



<p class="text">Welcome to CuteBlog! This is the place where users can connect and share posts about everything, anything they desire!</p>



<p class="text">I am currently a student in SIM-UOL studying Computer Science Degree and this is my mid-terms application coursework for the module CM2040.</p>



<p class="text">This is my take of the Mini Blogging tool mid-terms. The slogan is "Where cuteness meets Blogging.".</p>

<img src="/assets/AdleyPFP.png">

<p class="text" style="text-align: center;">CuteBlog is currently in early stages of development.</p>

<div style="height:100px" aria-hidden="true" class="wp-block-spacer"></div>

        </div>
      </main>
          </div>
        </div>
   </div>
   </div>
</div>
   </body>
   <%- include('./partials/footer'); %>
</html>
```

AccessDenied.ejs

```
<!-- access-denied.ejs -->
<!DOCTYPE html>
<html>
<head>
  <title>Access Denied</title>
  <link rel="stylesheet" href="/css/AccessDenied.css">
  <%- include('./partials/header'); %>
</head>
<body>
  <div class="container">
```

```
    <div class="content">
      <h1>Access Denied</h1>
      <p>Oops! It seems like you don't have permission to access this page.</p>
      <img src="/assets/sadcat.png" alt="Cute Cat Image">
      <br>
      <br>
      <button class="Return"><a href="/Dashboard">Return to Home Page</a></button>
    </div>
  </div>
</body>
</html>
```

Author-Home.ejs

```
//Start of Code written without Assistance for this code extract
<!DOCTYPE html>
<html>
<head>
  <title><%= title %></title>
  <link rel="stylesheet" type="text/css" href="/css/Author-home.css">
  <%- include('./partials/author-nav'); %>
  <%- include('./partials/metadata'); %>
</head>
<body>
<nav aria-label="Breadcrumb">
  <ol class="breadcrumb">
    <li>
      Author Home Page
    </li>
</nav>
<div class="author-page">
  <h1><%= settings.title %></h1>
  <h2 class="subtitle-settings"><%= settings.subtitle%></h2>
  <h3 class="author-heading">Author: <%= settings.author%></h3>
  <button class="header-buttons"><a href="/author/author-
settings"><strong>Settings</strong></a></button>
  <button class="header-buttons"><a href="/author/create-post"><strong>Create new
draft</strong></a></button>
  <br>
  <br>
  <!-- Published Articles List -->
  <h2><strong>Published Articles</strong></h2>
  <table class="published-articles">
    <thead>
      <tr>
        <th>Title</th>
        <th>Subtitle</th>
        <th>Content</th>
        <th>Likes</th>
        <th>Dislikes</th>
        <th>View Count</th>
        <th>Created At</th>
        <th>Modified At</th>
```

```
      <th>Published At</th>
      <th>Share</th>
      <th>Edit</th>
      <th>History</th>
      <th>Delete</th>
    </tr>
  </thead>
  <tbody>
   <% if (posts.length === 0 || !posts.some(post => post.status === 'published')) { %>
    <tr>
     <td colspan="11">No published posts to display.</td>
    </tr>
   <% } else { %>
    <% posts.forEach(function(post) { %>
     <% if (post.status === 'published') { %>
      <tr>
       <td><%= post.title %></td>
       <td><%= post.subtitle %></td>
       <td><%= post.content %></td>
       <td>
        <button class="like"><img src="/assets/heart.svg"><%= post.likes %></button>
       </td>
       <td>
        <button class="dislike"><img src="/assets/heart-crack.svg"><%=
post.dislikes %></button>
       </td>
       <td>
        <%= post.Views %>
       </td>
       <td><%=post.createdAt%></td>
       <td><% if (post.ModifiedAt) { %><%=post.ModifiedAt%><% } %></td>
       <td>
        <%= post.publishedAt%>
       </td>
       <td>
        <button class="Share-Button"><a href="#">Share Post</a></button>
       </td>
       <td>
        <form class="edit-post-form" method="POST" action="/author/edit-post">
         <input type="hidden" name="postid_for" value="<%=post.id%>">
         <button type="submit" class="Edit-post">
          <img src="/assets/edit.svg" alt="Edit">
         </button>
        </form>
       </td>
       <td>
        <form class="post-history-form" method="POST" action="/author/version-history">
        <input type="hidden" name="postid_" value="<%= post.id %>">
        <button><strong>Post History</strong></a></button>
        </form>
       </td>
```

```
          <td>
           <form method="POST" action="/author/delete-publishedpost">
            <input type="hidden" name="_postid" value="<%= post.id %>">
            <button class="Delete-post" type="submit">Delete
            </button>
           </form>
          </td>
         </tr>
        <% } %>
       <% }) %>
     <% } %>
   </tbody>
 </table>
 <!-- Draft Articles List -->
 <h2><strong>Draft Articles</strong></h2>
 <table class="draft-articles">
   <thead>
    <tr>
     <th>Title</th>
     <th>Content</th>
     <th>Subtitle</th>
     <th>Likes</th>
     <th>Dislikes</th>
     <th>Created At</th>
     <th>Publish Post</th>
     <th>Delete Post</th>
     <th>View Post</th>
     <th>Edit Post</th>
    </tr>
   </thead>
   <tbody>
    <% if (posts.length === 0 || !posts.some(post => post.status === 'draft')) { %>
     <tr>
      <td colspan="8">No draft posts to display.</td>
     </tr>
    <% } else { %>
     <% posts.forEach(function(post) { %>
      <% if (post.status === 'draft') { %>
       <tr>
        <td><%= post.title %></td>
        <td><%= post.content %></td>
        <td><%= post.subtitle %></td>
        <td>
         <button class="like"><img src="/assets/heart.svg"><%= post.likes %></button>
        </td>
        <td>
         <button class="dislike"><img src="/assets/heart-crack.svg"><%=
post.dislikes %></button>
        </td>
        <td><%=post.createdAt%></td>
        <td>
```

```
          <form class="Publish" action="/author/publish-post/<%= post.id %>"
method="POST">
           <button class="Publish-post">Publish</button>
          </form>
         </td>
         <td>
          <form class="delete-form" method="POST" action="/author/delete-post">
           <input type="hidden" name="postid" value="<%= post.id %>">
           <button class="Delete-post" type="submit">Delete</button>
          </form>
         </td>
         <td>
          <button class="View-post"><a href="/author/view-post/<%= post.id %>"
class="links">View Post</a></button>
         </td>
         <td>
          <form class="edit-post-form" method="POST" action="/author/edit-post">
           <input type="hidden" name="postid_for" value="<%=post.id%>">
           <button type="submit" class="Edit-post">
            <img src="/assets/edit.svg" alt="Edit">
           </button>
          </form>
         </td>
        </tr>
       <% } %>
     <% }) %>
    <% } %>
   </tbody>
  </table>
  <script>
   function showAlert() {
    var button = document.querySelector('.Share-Button');
    var postID = button.dataset.postid;
    var link = "/reader/view-post/" + postID;
    var message = '<a href="' + link + '">Share Post</a>';
    alert(message);
   }
  </script>
</div>
</body>
</html>
//End of Code written without Assistance for this code extract
```

Author-Settings.ejs

```
//Start of Code written without Assistance for this code extract
<!DOCTYPE html>
<html>
<head>
<%- include('./partials/author-nav'); %>
<%- include('./partials/metadata'); %>
          <title>Settings Page</title>
          <link rel="stylesheet" href="/css/settings.css">
```

```
</head>
<body>
	<div class="container">
		<h1>Settings Page</h1>
		<form id="settings-form" method="POST" action="/author/update-settings">
			<label for="blog-title">Blog Title:</label>
			<input type="text" id="blog-title" name="title"
value="<%=settings.title%>" required>
			<br><br>
			<label for="subtitle">Subtitle:</label>
			<input type="text" id="subtitle" name="subtitle"
value="<%=settings.subtitle%>" required>
			<br><br>
			<label for="author-name">Author Name:</label>
			<input type="text" id="author-name" name="author"
value="<%=settings.author%>" required>
			<br><br>
			<button type="submit">Update Settings</button>
		</form>
		<button class="back-button"><a href="/author/author-home">Back to Author
Home Page</a></button>
	</div>
</body>
</html>
//End of Code written without Assistance for this code extract
```

Confirmation.ejs

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="/css/confirmation.css" media="screen"/>
  <%- include('./partials/header'); %>
</head>
<body>
 <h1>Confirmation Page</h1>
 <div class="container">
  <h2><%= messages[0] %></h2>
  <% for (var i = 1; i<messages.length; i++){ %>
   <p><%= messages[i] %></p>
  <% } %>
 </div>
</body>

</html>
```

Create-post.ejs

```
<%- include('./partials/author-nav'); %>
<%- include('./partials/metadata'); %>
<!DOCTYPE html>
<html>
<head>
 <title>Create Post</title>
```

```
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <link rel="stylesheet" type="text/css" href="/css/create-post.css">
</head>
<body>
  <div class="container my-5">
    <h1>Create Post</h1>
    <form action="/author/create-post-process" method="POST" id="create-post-form">
      <div class="form-group">
        <label for="title">Title:</label>
        <input type="text" class="form-control" id="title" name="title" required>
        <div class="invalid-feedback">Please enter a title.</div>
      </div>
      <div class="form-group">
        <label for="subtitle">Subtitle:</label>
        <input type="text" class="form-control" id="subtitle" name="subtitle" required>
        <div class="invalid-feedback">Please enter a subtitle.</div>
      </div>
      <div class="form-group">
        <label for="content">Content:</label>
        <textarea class="form-control" id="content" name="content" rows="5"
required></textarea>
        <div class="invalid-feedback">Please enter the content.</div>
      </div>
      <button type="submit" class="btn btn-primary">Create</button>
    </form>
  </div>

  <script>
    // Enable Bootstrap form validation
    (function() {
      'use strict';
      window.addEventListener('load', function() {
        var forms = document.getElementsByClassName('needs-validation');
        var validation = Array.prototype.filter.call(forms, function(form) {
          form.addEventListener('submit', function(event) {
            if (form.checkValidity() === false) {
              event.preventDefault();
              event.stopPropagation();
            }
            form.classList.add('was-validated');
          }, false);
        });
      }, false);
    })();
  </script>
</body>
</html>
```

DashBoard.ejs

```
<html>
<head>
```

```
  <%- include('./partials/header'); %>
  <%- include('./partials/metadata');%>
  <link rel="stylesheet" href="/css/home.css">
</head>
<body>
  <div class="time-container">
    <p>Welcome back, <%= username %>!</p>
  </div>
<br>
<div class="blog-information">
  <h1>Blog title: <%=settings.title%></h1>
  <h2>Blog Author:<%=settings.author%></h2>
</div>
  <div class="blog-container">
    <h2><strong>DashBoard Page</strong></h2>
    <ul class="article-list">
      <% if (posts.length === 0 || !posts.some(post => post.status === 'published')) { %>
        <li>
          <p class="no-posts">No published posts to display.</p>
        </li>
      <% } else { %>
        <% posts.forEach(function(post) { %>
          <% if (post.status === 'published') { %>
            <li>
              <a href="/view-post/<%=post.id%>">
                <p class="title" style="color: black;"><%=post.title%></p>
              </a>
              <p class="subtitle"><%= post.subtitle %></p>
              <br>
              <p class="content"><%= post.content %></p>
              <button class="like"><img src="/assets/heart.svg"><%= post.likes %></button>
              <button class="dislike"><img src="/assets/heart-crack.svg"><%=
post.dislikes %></button>
              <button class="dislike"><img src="/assets/eye-svgrepo-com.svg"><%=
post.Views %></button>
              <p class="Time-Created">Published at <%= post.publishedAt%></p>
            </li>
          <% } %>
        <% }) %>
      <% } %>
    </ul>
  </div>
  <a href="javascript:void(0)" class="back-to-top">
    <span class="border-white"></span>
  </a>
</body>
<footer>
  <%- include('./partials/footer'); %>
</footer>
</html>
```

Edit-Post.ejs

```
<!-- Add your desired HTML structure and EJS tags here -->
<html>
<head>
 <link rel="stylesheet" type="text/css" href="/css/edit-post.css" media="screen"/>
</head>
<h1>Edit Blog Post</h1>
<form action="/author/update-post" method="POST">
 <label for ="CreatedAt">Post Created at:</label>
 <input type="text" name="created-at-time" value="<%= post.createdAt%>" readonly>
 <label for ="CreatedAt">Post Modifed at:</label>
 <input type="text" name="created-at-time" value="<%= post.ModifiedAt%>" readonly>
 <label for ="PostId">PostId</label>
 <input type="text" name="postId" value="<%= post.id %>" readonly>
 <label for="title">Title</label>
 <input type="text" name="title" value="<%= post.title %>">

 <label for="subtitle">Subtitle</label>
 <input type="text" name="subtitle" value="<%= post.subtitle %>">

 <label for="content">Content</label>
 <textarea name="content"><%= post.content %></textarea>

 <button class="back-button"><a href="/author/author-home">Back to Author Home
Page</a></button>
 <button type="submit">Save Changes</button>
</form>
</html>
```

Login.ejs

```
<html>

<head>
<title> <%= title %> </title>
<link rel="stylesheet" href="/css/login.css">
</head>
<body>
<h1>Login Page</h1>
<section class="login-page">
        <div class="section-header">
          <h2>Login to Cute Blog Here!</h2>
          <p><strong>The place where cuteness meets Blogging!</strong></p>
        </div>
        <div class="section-body">
          <div class="image-container">
            <img src="/assets/Login-Mascot.png" class="Mascot" alt="Image" style="padding:
10px;">
          </div>
          <form method="POST" id="username-form" action="/login-process">
           <div class="login-recovery"><strong>Username/Email</strong></div>
           <input class="textbox" type="text" name="input" value="" id="input" size="30"
placeholder="Username/Email">
```

```
          <div class="login-recovery"><strong>Password</strong></div><input class="textbox"
type="password" name="password" size="30" placeholder = "Password here
(e.g.password123)" id="password">
          <br>
          <div class="alignleft">
            <input type="submit" class="button" name="login" id="login-button"
value="Login">
          <p>New to CuteBlog? Register <a href="/register" style="text-decoration: none; color:
#ff6347;">here</a>!</p>
          </div>
         </form>
        </div>

    </section>
</form>
</body>
<br>
<br>
<br>
<footer>
<%- include('./partials/footer'); %>
</footer>
</html>
```

Mainpage.ejs

```
<html>
<head>
  <link rel="stylesheet" href="/css/main.css">
</head>
<body>
          <section class="hero-area">
    <div class="container">
      <div class="row">
        <div class="col-lg-6 offset-lg-3 col-md-12 col-12">
          <div class="hero-content">
            <h5 class="wow-animated" data-wow-delay=".2s" style="visibility: visible;
animation-delay: 0.2s;">Main Landing Page</h5>
            <h2 class="wow-h2" data-wow-delay=".4s" style="visibility: visible; animation-
delay: 0.4s;">Welcome to the world of CuteBlog!</h2>
            <p class="wow" data-wow-delay=".6s" style="visibility: visible; animation-delay:
0.6s;"> Choose where you would like to head to by pressing the buttons below!</p>
                                                <div class="button-wow" data-wow-delay=".8s"
style="visibility: visible; animation-delay: 0.8s;">
                                                    <a href="/author/author-home"
class="btn ">Author Home Page</a>
                                    </div>
                                    <div class="button-wow" data-wow-delay=".8s"
style="visibility: visible; animation-delay: 0.8s;">
                                            <a href="/DashBoard" class="btn ">Reader Home
Page</a>
                    </div>
          </div>
```

```
        </div>
      </div>
    </section>



</body>
<footer>
          <%- include('./partials/footer'); %>
          </footer>
</html>
```

Register.ejs

```
<html>
<head>
<link rel="stylesheet" href="../css/register.css">
<%- include('./partials/metadata'); %>
</head>
<body>
<section class="login-page">
  <div class="section-header">
    <h2 class="header-title">Account Registration</h2>
  </div>
  <br>
  <div class="section-body">
    <div id="page-register">
      <div id="part-pick-username">
        <form method="post" id="username-form" action="/register-process">
          <input type="hidden" name="send" value="send">

          <ol id="errors" class="alignleft error hidden"></ol>
          <div class="p10b">
            <div class="inline alignright"
style="width:170px"><strong>Username:</strong></div> <input type="text" class="textbox"
style="width:300px" name="username" id="Username" value="" maxlength="150"
autocomplete="off" fdprocessedid="co5bqp" required>
          </div>
          <% if (typeof usernameError !== 'undefined') { %>
            <p class="error-message" id="username-error"><%= usernameError %></p>
          <% } %>
          <div class="p10b">
            <div class="inline alignright" style="width:170px"><strong>Email:</strong></div>
<input type="text" class="textbox" style="width:300px" name="email" id="Username"
value="" maxlength="150" autocomplete="off" fdprocessedid="co5bqp" required>
          </div>
          <div class="p10b">
           <div class="inline alignright"
style="width:170px"><strong>Password:</strong></div> <input type="password"
class="textbox" style="width:300px" name="password" id="password" value=""
maxlength="150" autocomplete="off" fdprocessedid="jc7hf">
          <br>
          <p class="error-message" style="margin-bottom: 10px;"></p>
```

```
            <br>
            <p class="error-message" id="username-error" style="margin-bottom: 10px;"></p>
            <br>
            <p class="error-message" id="Email-Error"></p>
            </div>
            <br>
            <p class="login-prompt"><bold>Already have an account? Login</bold> <a href =
"/login" style="text-decoration: none; color: #ff6347;">here!</a>
            <div class="alignright">
              <input class="button" type="submit" name="proceed" id="proceed"
value="Proceed with account creation.">
            </div>
          </form>
        </div>
      </div>
    </div>
</section>

<script>
 const passwordInput = document.getElementById("password");
 const errorMessage = document.querySelector(".error-message");
 const submitButton = document.getElementById("proceed");
 const UsernameInput = document.getElementById("Username");
 const errorMessage2 = document.getElementById("username-error");

 passwordInput.addEventListener("input", () => {
  if (passwordInput.value.length < 8 && passwordInput.value.length > 0 ) {
   errorMessage.textContent = "Password too short. Password must be at least 8
characters.";
   submitButton.disabled = true;
  } else {
   errorMessage.textContent = "";
   submitButton.disabled = false;
  }
 });

 UsernameInput.addEventListener("input", () => {
  if (UsernameInput.value.length < 5 && UsernameInput.value.length > 0 ) {
   errorMessage2.textContent = "Username must be at least 5 characters.";
   submitButton.disabled = true;
  } else {
   errorMessage2.textContent = "";
   submitButton.disabled = false;
  }
 });




  </script>
  </body>
```

```
</html>
```

User-Profile.ejs

```
<html>
<head>
  <link rel="stylesheet" href="/css/back-up.css">
</head>
<div class="mh-wrapper">
  <div id="main-content" class="mh-content" role="main" itemprop="mainContentOfPage">
    <article id="post-7407" class="post">
      <header class="entry-header">
        <h1 class="entry-title">User Settings</h1>
        <p class="mh-meta entry-meta">
          <span class="entry-meta-date"><i class="fa fa-clock-o"></i>Date: <%=
currentDate %></span>
        </p>
      </header>
      <div class="entry-content mh-clearfix">
        <figure class="entry-thumbnail">
          <img src="/assets/AdleyBlackBoard(Smaller).png" alt=""
title="applications3_verysmall">
        </figure>
                                    <br>
                                    <br>
        <form>
          <label for="username">Username:</label>
          <input type="text" id="username" name="username" placeholder=" <%=
user.username %>">
                                          <label for = "email"> Email:</label>
                                          <input type="text" id="email" name="email"
placeholder="<%= user.email%>">
          <button type="submit">Save Changes</button>
        </form>
      <br>
      <br>
      </div>
      <div class="entry-content mh-clearfix">
        <form method="POST" action="/common/process-delete">
          <p class="warning">Note that Deleting of Account is not reversible.</p>
          <button class="Delete-Account"><a href="/common/process-delete">Delete
Account</a></button>
        </form>
      </div>
      <br>
      <br>
      <button class="return"><a href="/DashBoard">Return</a></button>
    </article>
  </div>
</div>
</html>
```

Version-History.ejs

```
<!DOCTYPE html>
<html>
<head>
 <style>
  body {
   font-family: Arial, sans-serif;
   background-color: #f5f5f5;
   margin: 0;
   padding: 0;
  }

  .container {
   max-width: 800px;
   margin: 0 auto;
   padding: 20px;
  }

  h1 {
   text-align: center;
   color: #333;
   margin-bottom: 20px;
  }

  table {
   width: 100%;
   border-collapse: collapse;
   background-color: white;
   box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
  }

  th, td {
   padding: 10px;
   text-align: left;
   border-bottom: 1px solid #ddd;
  }

  th {
   background-color: #f2f2f2;
  }

  p {
   text-align: center;
   color: #666;
   margin-top: 20px;
  }

  .Revert-Version {
 background-color: #6495ED; /* Cornflower Blue */
 border: none;
 color: white;
 padding: 8px 16px;
```

```
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 14px;
  border-radius: 4px;
  cursor: pointer;
}

.Revert-Version:hover {
  background-color: #4169E1; /* Dark Blue on hover */
}

.Return {
  background-color: #F5F5F5; /* Light gray background */
  color: #333333; /* Dark gray text color */
  border: none; /* No border */
  padding: 12px 24px; /* Moderate padding */
  text-align: center; /* Center the text */
  text-decoration: none; /* No underline */
  display: inline-block; /* Display as an inline block */
  font-size: 16px; /* Medium font size */
  font-family: 'Roboto', sans-serif; /* Use Roboto font */
  border-radius: 4px; /* Slightly rounded corners */
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1); /* Subtle drop shadow */
  transition: background-color 0.3s ease, box-shadow 0.3s ease; /* Smooth transitions */
}

.Return:hover {
  background-color: #E0E0E0; /* Slightly darker gray on hover */
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.15); /* Slightly larger drop shadow on hover */
}

.Return:active {
  background-color: #D0D0D0; /* Even darker gray when clicked */
  box-shadow: 0 1px 2px rgba(0, 0, 0, 0.1); /* Smaller drop shadow when clicked */
}

.Return a
{
  text-decoration: none;
  color: black;
}
  </style>
</head>
<body>
  <h1>Post Version History</h1>
  <p>This will only show the last version before the current version.</p>

  <% if (post.PostHistory !== null && post.PostHistory !== '') { %>
    <table>
      <thead>
```

```
      <tr>
        <th>Modifed Date</th>
        <th>Content</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td><%= new Date(post.ModifiedAt).toLocaleString() %></td>
        <td><%= post.PostHistory %></td>
        <td><button class="Revert-Version">Revert</button></td>
      </tr>
    </tbody>
  </table>
 <% } else { %>
  <p>No version history available for this post.</p>
 <% } %>
 <br>
 <br>
 <button class="Return"><a href="/author/author-home">Return</a></button>
</body>
</html>
```

View-Post.ejs

```
<!DOCTYPE html>
<html>
<head>
 <title>View Post</title>
 <link rel="stylesheet" type="text/css" href="/css/view-post.css">
 <%- include('./partials/metadata'); %>
</head>
<body>
 <div class="post-container">
  <h4 class="post-id">View Post <%= post.id %></h4>
  <h1 class="post-title"><%= post.title %></h1>
  <h3 class="post-subtitle"><%= post.subtitle %></h3>
  <p class="post-content"><%= post.content %></p>
  <div class="post-stats">
   <p class="post-dislikes">Views: <%= post.Views%></p>
   <p class="post-likes">Likes: <%= post.likes %></p>
   <p class="post-dislikes">Dislikes: <%= post.dislikes %></p>
  </div>
 </div>

 <form action="/like-post" method="POST">
  <input type="hidden" name="postLikeId" value="<%=post.id%>">
  <input type="hidden" name="userid" value="<%= user.id %>">
  <button type="submit">Like</button>
 </form>
 <form action="/dislike-post" method="POST">
  <input type="hidden" name="postdisLikeId" value="<%=post.id%>">
  <input type="hidden" name="userrid" value="<%= user.id %>">
```

```
  <button type="submit">Dislike</button>
 </form>

 <div class="comments-container">
  <h2>Comments(<%= commentCount.commentCount %>)</h2>
  <% if (comments.length === 0) { %>
   <p>No comments for this post to display</p>
  <% } else { %>
   <ul class="comments-list">
    <% comments.forEach(function(comment) { %>
     <li class="comment-item">
      <p class="comment-date">Commented by <%= comment.comment_user%> </p>
      <p class="comment-text"><%= comment.text %></p>
      <p class="comment-date"> Commented on <%= comment.createdAt %></p>
     </li>
    <% }) %>
   </ul>
  <% } %>
 </div>

 <form action="/add-comments" method="POST" class="comment-form">
  <label for="commenter">Comment As:</label>
  <input type="text" name="commenter" placeholder="<%=username%>">
  <label for="comment">Comment:</label>
  <input type="text" name="comment" placeholder="Write a comment.....">
  <input type="hidden" name="pstid" value="<%=post.id%>">
  <button type="submit">Submit</button>
 </form>
<br>
<br>
<button class="return"><a href="/DashBoard">Return</a></button>
</body>
</html>
```

Index.js

```
const express = require('express')
const app = express()
const router = express.Router();
const compression = require('compression');
const { check, validationResult } = require('express-validator');
//inclusion off express-validator module
const port = 3000; //Port used for Connection
const expressSanitizer = require('express-sanitizer'); //inclusion of
express-sanitizer module
const session = require('express-session');
const sqlite3 = require('sqlite3').verbose();
const rateLimit = require('express-rate-limit');
const helmet = require('helmet');
const expiryDate = new Date(Date.now() + 60 * 60 * 1000); //Cookie
Expires after 1 hour of inactivity
app.use(compression()); //greatly decrease the size of the response
body and hence increase the speed of the application.
const ApplicationLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
```

```
  max: 100,
  message: 'Too many attempts from this IP, please try again after 15
minutes',
  standardHeaders: true, //Return rate limit infomation in the
RateLimit -* headers
  legacyHeaders: false, // Disable the X-RateLimit-* headers
});
app.use(
  helmet.contentSecurityPolicy({
    directives: {
      defaultSrc: ['self', 'localhost'],
      scriptSrc: ["'self'", "'localhost'", "'unsafe-
inline'",'https://stackpath.bootstrapcdn.com'],
    },
  })
);
app.use(helmet.frameguard({
  action: 'deny'
}));
app.disable('x-powered-by');
// X-XSS-Protection
app.use(helmet.xssFilter());
// items in the global namespace are accessible throught out the node
application
global.db = new sqlite3.Database('./database.db', function (err){
if (err)
{
  console.error(err);
  process.exit(1); //Bail out as we can't connect to the DB
}
else
{
  console.log('Database Connected');
  global.db.run('PRAGMA foreign_keys=ON'); //This tells SQLite to pay
attention to foreign key constraints
}
});
// configure body parser to be able to receive the request body
const bodyParser = require ("body-parser");
app.use(bodyParser.urlencoded({ extended: true }));
// set the app to use ejs for rendering
app.set('view engine', 'ejs');
// use static files on the assets,css and Image Folders
const cssdirectory = express.static(__dirname + '/css');
const scriptdirectory = express.static(__dirname + '/scripts');
const ImageDirectory = express.static(__dirname + '/assets');
app.use(expressSanitizer()); //added for sanitisation
// add the reader routes to the app under the path /
const readerRoutes = require('./routes/reader');
app.use('/', readerRoutes);
// add the author routes to the app under the path /author
const authorRoutes = require('./routes/author');
app.use('/author', authorRoutes);
// add the common routes between reader and author under the path
/common
const commonRoutes = require('./routes/common');
app.use('/common', commonRoutes);
```

```
//Setting up Express.js to serve static files from three different
directories
app.use('/css/',cssdirectory);
app.use('/scripts/',scriptdirectory);
app.use('/assets/', ImageDirectory);
//add the Rate Limit to the application of CuteBlog
app.use('/author', ApplicationLimiter);
app.use('/common', ApplicationLimiter);
app.use(session({
  secret: 's3Cur3',
  name: 'sessionId',
  resave: false,
  saveUninitialized: true,
  cookie: {
    expires: expiryDate,
    domain: 'localhost'
  },
  rolling: true
}))
app.listen(port, () => {
  console.log(`CuteBlog Application listening on port ${port}`)
});
```