**Project Name:**

**ASTA – Informative Article Platform**

**Project Overview:**

ASTA is a web application designed to help researchers and readers access informative articles written by professional or experienced writers who sign up on the platform. The platform enables users to:

* Read and engage with articles.
* Comment on posts and react to them.
* Writers can reply to comments, creating an interactive environment.

Additionally, an **admin dashboard** will allow platform administrators to manage users, posts, and site settings.

This project also serves as a demonstration of my skills in **HTML**, **CSS**, **Node.js**, **Express**, **MySQL**, and related tools.

**Functional Requirements:**

**1. User Authentication**

* Implement a secure authentication system for:
  + **Regular Users:**
    - Sign up, log in, and log out.
    - Only logged-in users can:
      * Write posts.
      * Comment on articles.
      * React to posts or comments.
  + **Admin Users:**
    - Access an admin dashboard to manage the platform.

**2. Admin Features**

* **User Management:**
  + View and delete user accounts.
  + Ban or unban users if necessary.
* **Post Management:**
  + Delete inappropriate or irrelevant posts and comments.
  + Edit or feature important posts on the home page.
* **Header Image Management:**
  + Add, update, or delete header images displayed on the home page.
* **Analytics (Optional):**
  + View platform activity, such as the number of users, posts, and comments.

**3. Article Management**

* **For Regular Users:**
  + Authenticated users can create and publish articles.
  + Support text formatting and image uploads in posts.
* **Display:**
  + Show recent articles on the sidebar for easy access.
  + Provide a detailed view for reading individual articles.

**4. User Profiles**

* Allow users to view writer profiles, including their bio and published articles.
* Optional: Add functionality to follow writers for updates on their posts.

**5. Communication Features (Optional)**

* Enable real-time comment updates, so replies appear without refreshing the page.
* Optional: Provide private messaging or group discussions for better collaboration.

**Technical Requirements:**

* **Frontend:**
  + HTML5, CSS3, Bootstrap for styling and responsive design.
* **Backend:**
  + **Node.js** with **Express** for server-side logic.
  + **MySQL** for database management.
* **Admin Panel Framework:**
  + Use tools like **AdminBro**, **express-admin**, or build a custom admin interface for backend management.
* **Version Control:**
  + Use **Git** to track and manage project changes.
* **Deployment:**
  + Deploy the application on a hosting service like **Glitch**, **Railway**, or **Vercel**, ensuring public accessibility.

**Additional Considerations:**

1. **User Experience (UX):**
   * Design a clean, intuitive interface to make the platform easy to use for readers, writers, and admins.
   * Ensure mobile and cross-browser compatibility for a seamless experience.
2. **Testing:**
   * Conduct rigorous testing for user authentication, admin actions, article posting, and commenting features.
   * Test on multiple devices and browsers to ensure compatibility and performance.
3. **Documentation:**
   * Provide clear documentation covering:
     + Project structure and setup.
     + Admin panel functionalities.
     + Core platform features and how to use them.
     + Include comments within the codebase for better readability and maintainability.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

When users successfully register with their username, email, and password, you can ask them to update or provide additional profile information to make their profiles more complete and personalized. Here's a list of suggested information:

**Basic Profile Information**

1. **Full Name** *(optional but recommended)*:
   * Example: "John Doe."
   * This can be used for display purposes on their profile or posts.
2. **Profile Picture** *(optional but highly recommended)*:
   * Allow users to upload a profile photo to make their profile visually identifiable.
3. **Bio/Description**:
   * A short paragraph where users can describe themselves, their expertise, or interests.
4. **Location** *(optional)*:
   * City, State, or Country.
   * Helps other users identify where the writer is from or adds context to their posts.

**Professional/Writer Information *(Optional for regular users but important for professional writers)***

1. **Occupation/Profession**:
   * Example: "Freelance Writer," "Researcher," or "Software Engineer."
2. **Expertise/Focus Areas**:
   * Example: "Technology," "Health," "Business," etc.
   * This can help categorize their posts and attract the right audience.
3. **Website/Portfolio Link**:
   * If they have a personal website, blog, or professional portfolio.
4. **Social Media Links** *(Optional)*:
   * LinkedIn, Twitter, or any professional social platform for networking.

**Preferences/Optional Features**

1. **Notification Preferences**:
   * Whether users want to receive email notifications for:
     + New comments on their posts.
     + New followers (if following is enabled).
     + Updates from the platform.
2. **Privacy Settings**:
   * Whether their profile is public or private.
   * Option to hide email or other sensitive details.

**Optional Extras for Gamification or Personalization**

1. **Favorite Topics/Tags**:
   * Allow users to select their interests, which could be used to recommend articles.
   * Example tags: Technology, Science, Education, Lifestyle, etc.
2. **Custom Username**:
   * If you plan to show a username different from their email during interactions (e.g., for comments or posts).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## ****Workflow for the System****

### ****1. Registration and Login Workflow****

#### **1.1 User Registration**

* **Step 1:** User visits the registration page.
* **Step 2:** User provides **username**, **email**, and **password**.
* **Step 3:** Validate inputs on both client-side (basic checks) and server-side (e.g., unique email).
* **Step 4:** Encrypt the password (e.g., using bcrypt) and save user details in the database.
* **Step 5:** Redirect the user to the login page upon successful registration.

#### **1.2 User Login**

* **Step 1:** User visits the login page and enters their **email** and **password**.
* **Step 2:** Authenticate user credentials by verifying them against the database.
* **Step 3:** If successful:
  + Generate and assign a session or JWT token for the user.
  + Redirect the user to the homepage or dashboard.
* **Step 4:** If unsuccessful, show an error message and allow retry.

#### **1.3 Admin Login**

* **Step 1:** Admin uses a separate login page or the same one with role-based access.
* **Step 2:** Authenticate credentials with admin roles defined in the database.
* **Step 3:** Redirect to the **admin dashboard** upon successful login.

### ****2. User Workflow****

#### **2.1 Update Profile**

* **Step 1:** After logging in, the user is prompted to complete their profile (optional).
* **Step 2:** User can update their **full name, bio, profile picture, location, and interests** from the profile settings page.

#### **2.2 Creating a Post**

* **Step 1:** Logged-in user accesses the "Create Post" page.
* **Step 2:** User writes content using a rich text editor and can upload images.
* **Step 3:** Validate inputs (e.g., required title and body).
* **Step 4:** Save the post in the database, associating it with the user ID.
* **Step 5:** Redirect the user to the post view page upon successful posting.

#### **2.3 Interacting with Posts**

* **Step 1:** Users can browse posts on the homepage or sidebar.
* **Step 2:** Clicking on a post title redirects the user to the detailed post view.
* **Step 3:** Logged-in users can:
  + Leave comments.
  + React to posts (e.g., like, dislike, etc.).
* **Step 4:** Writers can reply to comments directly, creating an interactive experience.

### ****3. Admin Workflow****

#### **3.1 Admin Dashboard**

* **Step 1:** Admin logs in and is redirected to the admin dashboard.
* **Step 2:** Admin sees summarized stats like:
  + Total users.
  + Number of posts.
  + Comments and reactions.
  + Reports (if any).

#### **3.2 User Management**

* **Step 1:** Admin accesses the user management section.
* **Step 2:** Admin can:
  + View all users.
  + Delete or ban users as necessary.

#### **3.3 Post Management**

* **Step 1:** Admin accesses the post management section.
* **Step 2:** Admin can:
  + Delete inappropriate posts or comments.
  + Feature or highlight posts on the homepage.

#### **3.4 Manage Header Images**

* **Step 1:** Admin accesses the settings section.
* **Step 2:** Admin can upload, delete, or update the homepage header images.

### ****4. Homepage Workflow****

#### **4.1 Guest Users**

* Guests (unauthenticated users) can:
  + View posts.
  + Read comments but cannot interact (comment, react, or post).
  + See a **CTA (Call-to-Action)** to log in or sign up.

#### **4.2 Logged-In Users**

* Logged-in users can:
  + Read posts and comments.
  + React to and comment on posts.
  + Create new posts.

### ****5. Notifications Workflow (Optional)****

* **Step 1:** If notifications are enabled, users receive alerts (via email or in-app):
  + When someone replies to their comment.
  + When a writer they follow posts new content.

### ****6. Database Interaction Workflow****

#### **6.1 Key Actions**

1. **User Registration/Login:**
   * Insert user details or verify credentials.
2. **Create Post:**
   * Insert the post into the posts table with user ID as a foreign key.
3. **Comment or React:**
   * Insert comment/reaction into the database, linking it to both the user and the post.
4. **Admin Actions:**
   * Update/delete records in the users, posts, or comments tables as needed.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### ****7. Deployment Workflow****

1. **Development Setup:**
   * Use Git for version control.
   * Use a local server (e.g., Node.js with Express) and test the database connection.
2. **Testing:**
   * Thoroughly test user authentication, post creation, and admin features locally.
3. **Deployment:**
   * Deploy on platforms like **Railway**, **Vercel**, or **Glitch**.
   * Set up the database on a cloud service (e.g., Amazon RDS, PlanetScale).
4. **Maintenance:**
   * Regularly monitor logs and user activity.
   * Apply updates or fixes as needed.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Relationships Overview**

1. **Users** → **Posts**:
   * A user can create multiple posts, but each post belongs to one user.
2. **Posts** → **Comments**:
   * A post can have multiple comments, and a comment must belong to one post.
   * Supports threaded comments with parent\_comment\_id.
3. **Users** → **Reactions**:
   * A user can react to multiple posts, but each reaction is tied to one user and one post.
4. **Admin Actions**:
   * Tracks all administrative actions for transparency and auditing.
5. **Header Images**:
   * Admins can upload multiple header images, which are used for the homepage.

### ****Sample Data Flow****

#### **Users Table**

|  |  |  |  |
| --- | --- | --- | --- |
| id | username | email | is\_admin |
| 1 | admin | admin@test.com | TRUE |
| 2 | john\_doe | john@test.com | FALSE |

#### **Posts Table**

|  |  |  |
| --- | --- | --- |
| id | title | user\_id |
| 1 | First Blog Post | 2 |

#### **Comments Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | post\_id | user\_id | comment | parent\_comment\_id |
| 1 | 1 | 2 | Great post! | NULL |
| 2 | 1 | 1 | Thanks, John! | 1 |