Historical Artifacts Tracker

Project Overview and Discussion

This project involves creating a web application for tracking historical artifacts such as Rosetta Stone, Antikythera Mechanism etc. The application will allow users to browse artifacts, view details, and add their own entries. The system will also allow users to like artifacts and keep track of their own contributions. The application should provide a user-friendly interface for managing and displaying information about various historical items.

Deployment Guideline

If your Deployment is not okay you will get 0 and may miss the chance of our upcoming rewards.

- Ensure that your server is working perfectly on production and not throwing any **CORS / 404 / 504** Errors.
- Ensure that your Live Link is working perfectly and that it is not showing errors on Landing in your system.
- ⚠️ ensure that the page doesn't throw any error on reloading from any routes.
- ⚠️ Add your domain for authorization to Firebase if you use **Netlify** / **surge** / **Vercel**
- ⚠️ Logged in User must not redirect to Login on reloading any private route

---

Layout & Page Structure

1. **Navbar:** The navbar will contain the
    - Website Name/Logo: Should reflect the theme,
    - Home,
    - All Artifacts
    - Add Artifacts (Private/Protected Route)
    - Conditional "Login" and "Logout"

a. The “Login” button is conditional, if the user is not logged in it will show the “Login” Button. If a user clicks on “Login” it will redirect to the login page.
b. But If the user is logged in here you will show the user photoURL, when you hover over the image it will show the displayName. And it will show the “Logout” button.
c. On clicking on My Profile it will show a dropdown menu. It will show the following routes -
    i. My Artifacts (Private/Protected Route)
    ii. Liked Artifacts (Private/Protected Route)

2. **Main Section:** Main Section will show different pages based on routes.
3. **Footer:** A Footer with all relevant information and eye-catching design.

Authentication System

4. **Login Page:** When you click the login button on the navbar it redirects to the login page. You have to use a password and email-based authentication to log in. The login page will have-
    a. Email
    b. Password
    c. Google login/ GitHub- implement any of one
    d. A link that will redirect to the Register page

5. **Register Page:** You have to use a password and email-based authentication to register. The Register page will have the following -
    a. Name
    b. Email
    c. photoURL
    d. password
    e. A Link that will redirect to the login page

    ★ For password verification you need to follow this -

- Must have an Uppercase letter in the password
- Must have a Lowercase letter in the password
- Length must be at least 6 character

★ If any of this isn't fulfilled it will show an error message /toast

★ After successful login or Register you need to show toast/sweet alert

---

Home Page

6. **Banner/Slider:** Add a slider (you can use any type of slider/carousel) with a minimum of 3 slides and meaningful information. Make sure the banner/slider seems eye-catching.

7. **Featured Artifacts:** This section will show a maximum of 6 Artifacts with the highest like count.
   Each card will show the following info-
   a. Artifact Image
   b. Artifact Name
   c. Short description
   d. Like Count
   e. View Details button

Clicking on the "View Detail" button will redirect users to the Artifact Details Page, where they can access comprehensive information about the selected artifact. (see requirement 10)

🎯To show the 6 Artifacts with the highest like count you can use the MongoDB sort method.

➔ **See all button:** Below the 6 cards, there will be a see all button that will redirect the user to the All Artifacts Page (see requirement 12).

8. **Extra Section:** Add 2 relevant and meaningful extra sections on the Home page.

Add Artifacts Page (Private route)

9. Create an “Add Artifact” page where there will be a form for the user to add an Artifact. The form will have:
    a. Artifact Name
    b. Artifact Image (valid URL)
    c. Artifact Type (dropdown: Tools, Weapons, Documents, Writings, etc.)
    d. Historical Context
    e. Created At (string, e.g., ”100 BC”)
    f. Discovered At (string, e.g., ”1799”)
    g. Discovered By
    h. Present Location
    i. Artifact adder name and email (Logged-in user email & name) (read-only)
    j. Add Artifact button

    This will be a private/protected route.
    Clicking on the ”Add Artifact” button. Upon successful submission, store the data in the database and display a success message using a toast or a SweetAlert notification. **Initially the like count will be 0 for an Artifact.**

Artifact Details Page (Private route)

10. The Artifact Details page will be a private/protected route. Please make sure that if the user is not logged in, the private route redirects to the login page. On this page, you will show all the information you have stored in the database about an artifact and there will be a Like Button and Like Count.
11. Clicking on the Like Button will increase the Like Count by 1. This updated count will also be saved to the database.

All Artifacts Page

12. Display all artifacts in a card format, showing 2-4 pieces of information about each artifact. Each card will include a ”View Detail” button. Clicking on the ”View Detail” button will redirect users to the Artifact Details Page, where they can access comprehensive information about the selected artifact. (see requirement 10).

Liked Artifacts Page (Private Route)

13. **It will be a private/protected route.** On this page, a user can see all the artifacts that the user has liked.
    🎯If there is no data for Liked Artifacts page then show a meaningful message or something relevant.

My Artifacts Page (Private route)

14. **It will be a private/protected route.** On this page, displays all the artifacts added by the logged-in user, ensuring that each user can view only their own submissions. For each artifact listed, there will be an Update button and a Delete button, allowing the user to modify or remove their own data as needed. However, users will not have access to update, or delete artifacts added by others, ensuring data privacy and security.
    🎯If there is no data for My Artifacts page then show a meaningful message or something relevant.
15. **Update Page:** When a user clicks on the "Update Button" it will take the user to the Update page, where there will be a form for the user to update an artifact. All the previous data will show as the default value. The form will have the following:
    a. Artifact Name
    b. Artifact Image (valid URL)
    c. Artifact Type (dropdown: Tools, Weapons, Documents, Writings, etc.)
    d. Historical Context
    e. Created At (string, e.g., "100 BC")
    f. Discovered At (string, e.g., "1799")
    g. Discovered By
    h. Present Location
    i. Update Artifact button

This will be a private/protected route. When you fill in the data and submit the "Update Artifact" button, this data will be updated to the previous data in your database and you will show a success message through toast/sweet alert. **The update operation must be implemented in such a way that it does not affect**

the value of the like count and artifact adder information under any circumstances.

🎯🎯**(Optional):** If you don't want to create an Update page, you can also use a **modal** to update your data. For this when you click on the "Update" button it will open a modal but make sure you are logged in before updating the data.

🎯**For all the CRUD operations, show relevant toast/ notification/ sweet alert with a meaningful message**

16. **Delete Button**- If the user clicks the delete button, the Artifact will be removed from the database. Before the delete, ask for a delete confirmation. After successful deletion, the user will be redirected to the All Artifacts page (see requirement 12).

Additional

- **Dynamic Title:** Make your website title Dynamic. For every Route change, The Website Title will be changed based on that route.
- **404 page:** Add a 404 page/Not Found Page.
- **Spinner:** Show a loading spinner when the data is in a loading state.
- **Toast:** For all the CRUD operations, show relevant toast/ notification/ sweet alert with a meaningful message.

Challenge Requirement Guideline

- **JWT Authentication:** Upon login, you will create a JWT token and store it on the client side. You will send the token with the call and verify the user. Implementing 401 and 403 is optional. Ensure you have implemented the JWT token, create a token, and store it on the client side for both email/password-based authentication and social login. You must implement JWT on your private routes.
- **Toggle Like Button:** The Like Button allows users to toggle between "Liked" and "Disliked" states. Clicking the button updates its visual state (e.g., icon or color change) and modifies the like count displayed on the page. If the artifact is liked, the count increases by one, and if disliked,

the count decreases. These changes are reflected both on the page and in the database, ensuring synchronization between the user interface and backend data.

- **Search Functionality:** On the top of the All Artifacts Page, you need to implement search functionality through a search input based on the Artifact Name. You can implement it through the backend.

Optional (But Highly Recommended):

Implement any two tasks from the following optional list:

1. Try to use any other tailwind CSS library like - mamba Ui, shadcn, chakra UI, flowbite.
2. Add a spinner when the data is in a loading state. You can add a gif/jpg, use any package or customize it using CSS.
3. Explore and implement any of the animations from the Framer Motion.
4. Add one extra feature of your own. This will help you in the future to differentiate your project from others.

What to Submit

Live Site Link :
Github Repository (**server**) :
Github Repository (**client**) :

Additional information:

1. You can host images anywhere.
2. You can use vanilla CSS or any library.
3. Try to host your site on Firebase (Netlify hosting will need some extra configurations)
   Firebase Hosting Setup Complete Issue
4. Host your server-side application on Vercel. If needed, you can host somewhere else as well.
   How to deploy a Node/Express server using Vercel CLI
   Some Common Vercel Errors
5. Make Sure you deploy server-side and client-side on the first day. If you have any issues with hosting or GitHub push, please join the "Github and deploy" related support session.

Some Guidelines:

1. Do not waste much time on the website idea. Just spend 15-20 minutes deciding, find a sample website, and start working on it.
2. Do not waste much time finding the right image. You can always start with a simple idea. Make the website and then add different images.
3. Don't look at the overall task list. Just take one task at a time and do it. Once it's done, pick the next task. If you get stuck on a particular task, move on to the next Task.
4. Stay calm, think before coding, and work sequentially. You will make it.
5. Be strategic about the electricity issue.
6. use chatGPT to generate JSON data. You can use chatGPT for other purposes as well.