Test Progressive Assistants	
Web Design and Technology	
For 24-2 Generation	
Periode Berlaku Semester Genap 2024/2025 ***Valid on*** *Even Year 2024/2025*	**Software Laboratory Center** **Academic Development**

1. Peserta TPA tidak diperkenankan untuk:
   a. **Memublikasikan jawaban dan/atau soal** sebelum waktu pelaksanaan TPA berakhir, baik dengan sengaja maupun tanpa disengaja.
   b. **Membuka dan/atau menyalin jawaban** dari buku, video, ataupun peserta lainnya.
   c. **Melakukan tindakan yang menyebabkan jawaban dicontek** oleh orang lain atau kelompok lain, baik disengaja maupun tidak disengaja.
   d. **Melakukan tindakan kecurangan lainnya**.

2. Jika peserta TPA terbukti melakukan tindakan seperti yang dicantumkan pada butir ke-1, maka **nilai TPA** yang melakukan kecurangan, baik menyontek atau dicontek, akan **dinolkan** sesuai dengan peraturan yang berlaku. Dan akan diberikan **sanksi** sesuai prosedur yang berlaku.

3. Jawaban yang dapat diterima dan dinilai adalah jawaban yang **dikumpulkan sebelum batas waktu** yang telah ditentukan.

Jakarta, 25 April 2025

Catherine Alyssa Rapito
Assistant Development Officer

Soal
Case

AY.com

AY.com is a web-based **social media platform** designed for seamless thread-based conversations. It allows users to share short posts, images, and videos, engage through likes, replies, reposts, and explore trending topics.

AY is a **dedicated project manager**, known for his ability to keep projects on track and deliver results efficiently. He ensures that every aspect of the project runs smoothly, from planning and execution to final delivery.

AY wants you to develop **AY.com**. During development, **AY** emphasizes the importance of **understanding the tech stack and all tools you use to build the platform**. **AY** wants you to use **Git** as the **version control system** because he wants to **track your progress** and ensure that development stays on track. Additionally, **AY** expects **clear** and **well-structured commit messages** because they ensure transparency, accountability, and long-term maintainability of the project. **AY** wants you to use the **GitHub repository** provided by the Assistant Development Officer to store and manage your projects. **AY** expects you to demonstrate **integrity** by ensuring that every project update you present **accurately** reflects the state of your work as recorded in the **last commit before the deadline**.

In this project for the frontend, you must use **Svelte** as the **framework** and **Vite** as the **build tool**, with **TypeScript** as the programming language to ensure type safety. **AY** wants you to use **pure CSS/SASS/SCSS** for **styling** in **AY.com** to maintain full control over the design system and avoid unnecessary dependencies. You are **prohibited** from using any **CSS libraries** or **UI component libraries** such as Tailwind, Bootstrap, Svelte Material UI, etc. In the design factor, the web design must be **responsive** with **3 (three) breakpoints: desktop**, **tablet**, and **mobile**. Other than **UI** and **Design**, consider the **UX** of the application, because **the experience** is what makes customers loyal to the web, **not the design**.

For the backend, you must use the **Go** programming language using **Microservices** architecture. Each **microservice** should handle a **specific functionality**. To enable efficient communication between microservices, you must use **gRPC** for high-performance, low-latency

remote procedure calls. Additionally, a **message broker** such as **RabbitMQ** must be implemented for asynchronous messaging and event-driven workflows. Each **microservice** must be **fully independent**, meaning that every **service** should have its own **separate database** to ensure loose coupling and autonomy.

You must implement an **API Gateway** to act as an **intermediary between** the **frontend** and the **backend microservices**. Since frontend relies on HTTP, it cannot directly interact with the microservices, which communicate internally using **gRPC**. Also, do not forget to add **API Documentation** using **Swagger**.

For the database you must use **PostgreSQL**. You have **flexibility** to **design** and **structure the database** for each **microservice** as needed, but every design choice must be **backed by a strong and logical reason**. Be prepared, **AY** might ask **why**, and you better have a **solid answer**!

You must implement **Redis** caching to enhance **performance** and **reduce database load**. Additionally, **logging** must be integrated into the **application** to provide **visibility into system activity** and **support effective debugging**. The **logging** system must support **multiple log levels** (e.g., debug, info, warning, error, etc.), and it must be **configurable** so that the **log level displayed** can be **dynamically adjusted** based on the **environment**. There are two environments: **Development** and **Production**. The logs must be **output only to the terminal** (standard output), and should not be written to files or external systems.

Furthermore, **unit testing** must be implemented to ensure **code reliability** because **AY** does not trust code that has not proven itself. **Assertions** should be used to validate expected outcomes, while **mocking** allows for **testing components** in **isolation** by simulating dependencies like **databases** or **external services**.

For security, **AY** wants you to use **JSON Web Tokens (JWT)** to ensure secure **authentication** and **authorization** across the platform. Additionally, you must implement **salting** for the **user's password** to enhance security. **AY** also requires the implementation of both **access tokens** and **refresh tokens**.

You can use **Supabase** to store **media content** such as images and videos. **AY** wants you to use **Flask** for deploying the **AI** components of the **application**. **AY** wants you to use **Docker** to **containerize frontend, Flask** and **each microservice**, ensuring consistency across different

environments and simplifying deployment. Additionally, you must use **Docker Compose** to manage and **run multiple Docker containers together**. You must use **ESLint** that is given by **AY**.

While working on this project, remember to **focus** on the **main functionality** of the web application. Also, remember this project is a **great opportunity to learn something new**. If you're feeling tired, take a break and recharge. It's important to stay fresh and energized. Most importantly, **have fun** with the process and embrace your **creativity**.

You can use this following **logo** or **icon** for the website:

Figure 1. AY.com Dark Logo

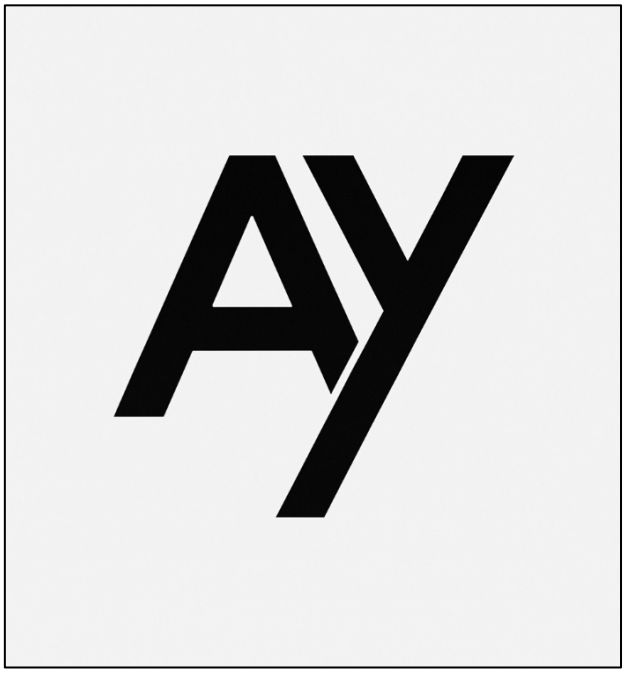

Figure 2. AY.com Light Logo

Here are the **Project Requirements** you need to build for this project:

- **General**
  - Pay attention to **performance** in frontend, backend, and database.
  - Able to explain the **purpose** and the **reason** for the code and action.
  - Follow the **provided description** for each page as a **requirement**, while the given **prototype** serves only as a **reference**.

➢ **All Page**

- Pay attention to **UI** and **UX** for every page.
- **Responsive** on desktop, tablet, and mobile.
- It has **2 (two) themes**: **Light** and **Dark**, the default is Light. There must be a **toggle** to **swap** between these themes.
- Have proper **authorization** and **authentication**.
- Display **error message** if an error occurs.

➢ **Landing Page**

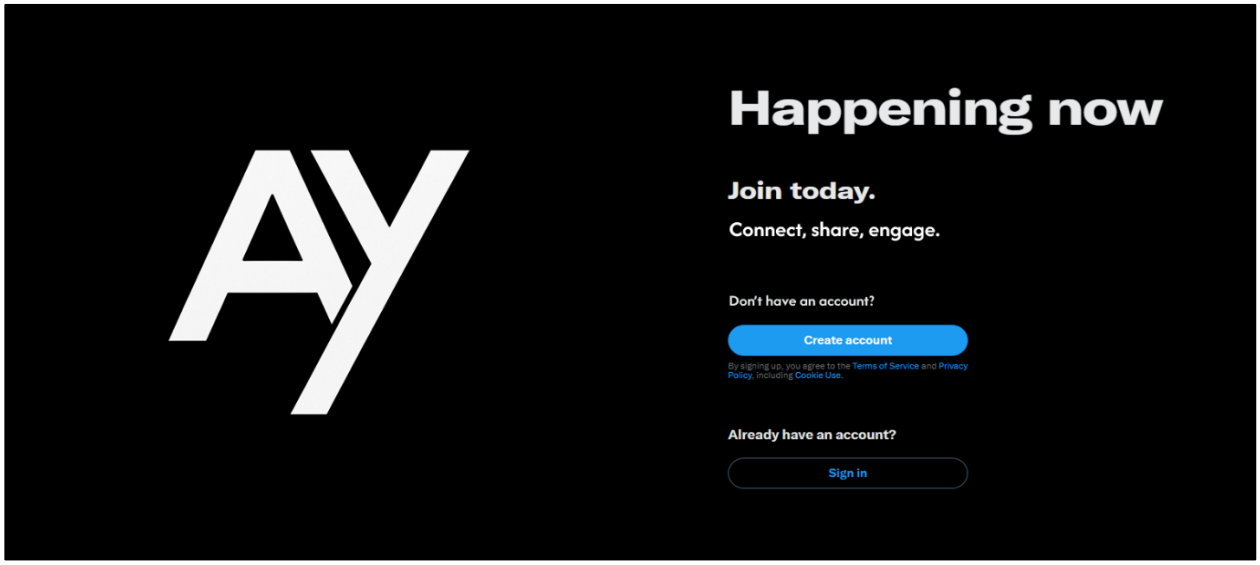

Figure 3. Landing Page

- This page is accessible to **guests** only (non-logged-in-users).
- Displays the **AY.com** logo.
- Provides a **link** to the **Register Page**.
- Provides a **link** to the **Login Page**.

➢ Register Page

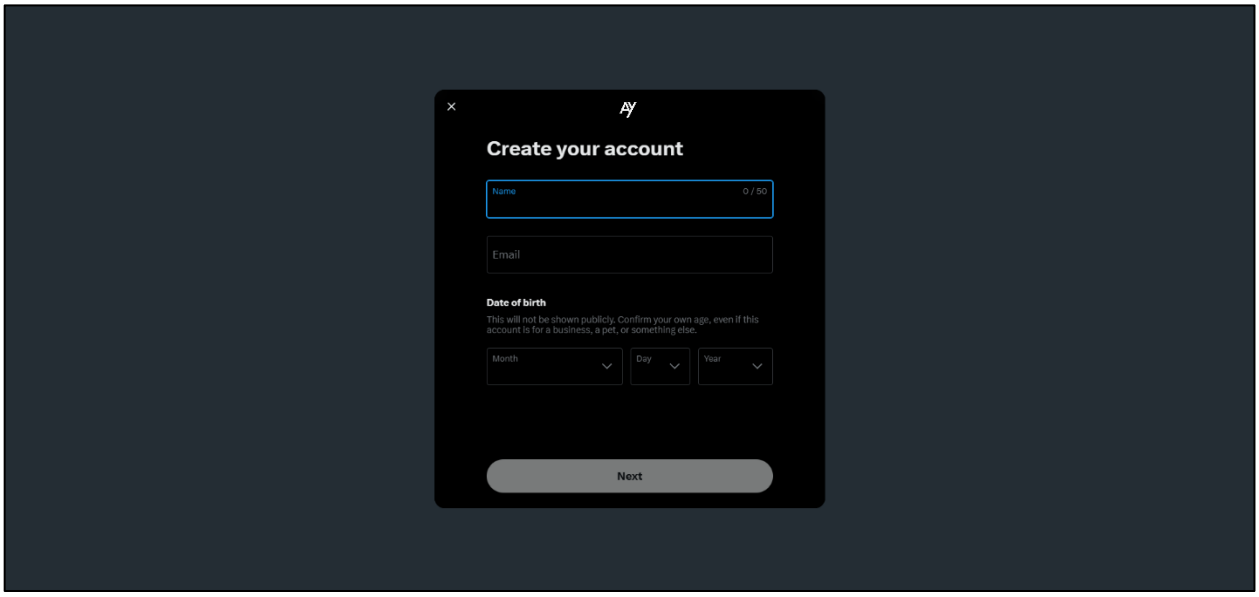

Figure 4. Register Page (Registration Step 1)

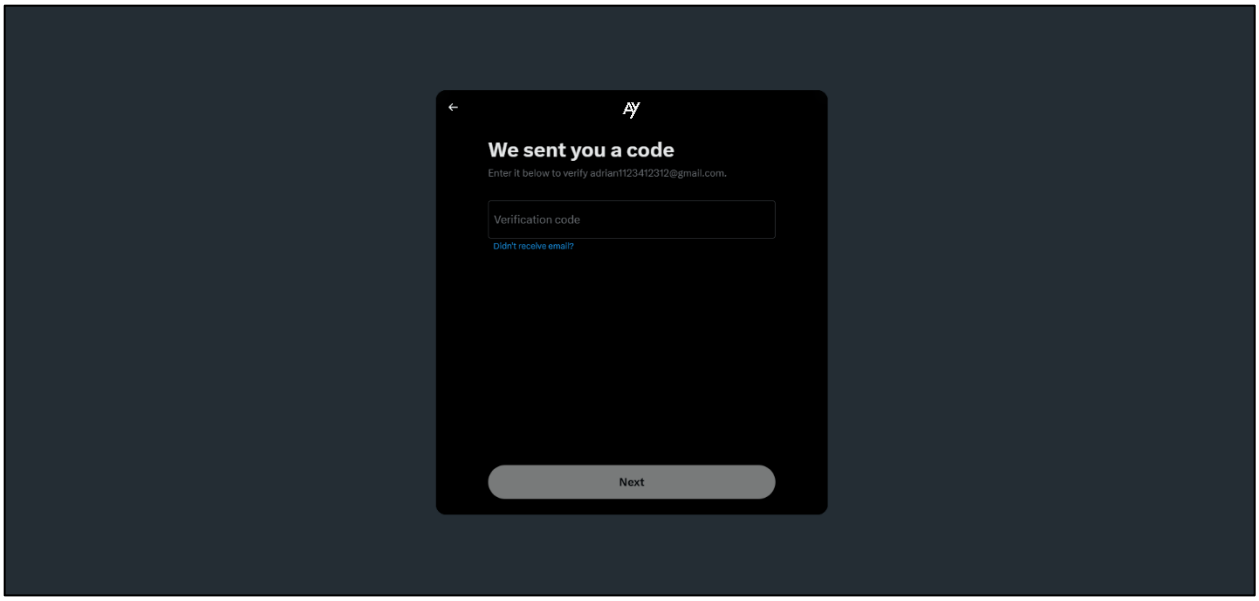

Figure 5. Register Page (Registration Step 2)

- This page is accessible to **guests** only (non-logged-in-users).
- Create an option for the user to authenticate using their **Google Account**. Note that some registration fields may not be fully completed through Google Authentication. User may need to **manually** fill in any **missing information** to complete the registration.

- Register Page (Registration Step 1):
  - The user must fill out the **form** with the following required details: **name**, **username**, **email**, **password**, **confirm password**, **gender**, **profile picture**, **banner**, **date of birth**, and **at least 1 (one) personal security question**.
  - Validate the **name** must be **more than 4 (four) characters** and there must be **no symbols or numbers**.
  - Validate the **username** must be **unique** and can only be used by one account.
  - Validate the **email** to match the following pattern **[email name]@[domain].com** (ex: adrian.yu@gmail.com).
  - Validate the **email** must be **unique** and can only be used by one account.
  - Make **at least 4 (four)** different validations in the **password** field.
  - Validate the **password** and **confirm password** must be the **same**.
  - Validate the **gender** must be **male** or **female**.
  - Validate the **age** must be **greater than** or **equal** to **13 years old**.
  - For **personal security questions**, provide a **dropdown menu** using the following list of questions:
    1. What was the name of your first pet?
    2. What city were you born in?
    3. What is your favorite video game?
    4. What was the name of your first school?
    5. What was your childhood nickname?
  - The user can choose to **subscribe to newsletters** by checking a checkbox.
  - Send a **verification code** to the user's email. The code is valid for **5 minutes** before it expires.
  - If the user **doesn't receive the email** or the **verification code expires**, they can request a new code to be **resent**.
  - Implements **reCAPTCHA** to avoid spam.
  - Provides a **link** to the **next or previous step**.
  - Provides a **link** to the **Landing Page**.

- Register Page (Registration Step 2):
  - The user must enter the **verification code** sent to their email.
  - If the **verification code is valid**, activate the user's account. Then, send an **email** to inform the user that their registration is successful and **redirect them to the Login Page**.
  - Provides a **link** to the **next or previous step**.

➢ **Login Page**

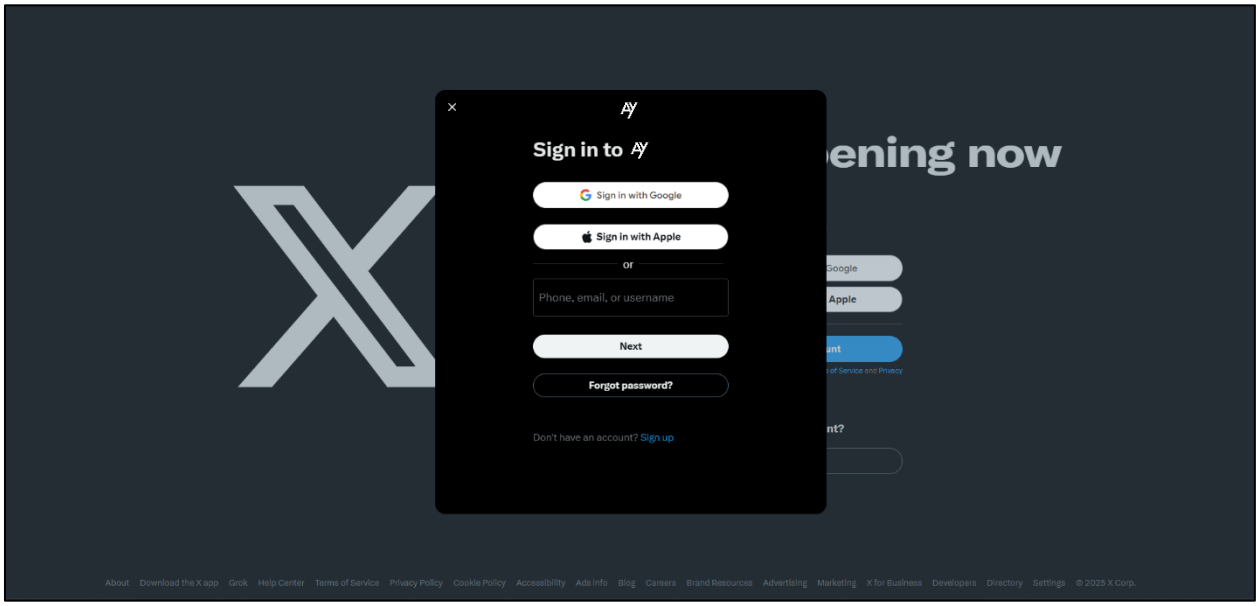

Figure 6. Login Page

- This page is accessible to **guests** only (non-logged-in-users).
- The user must fill out the **form** with the following required details: **email** and **password**.
- Only **activated accounts** that are **not banned** and **not deactivated** are **permitted access**.
- Redirect **guests** to this page if they attempt to **access an unauthorized or restricted page**.
- Implements **reCAPTCHA** to avoid spam.
- Create an option for the user to authenticate using their **Google Account**. Use Google service to authenticate users and **automatically log users in**, then redirect to **Home Page**.
- Provides a **link** to the **Landing Page**.
- Provides a **link** to the **Forgotten Account Page**.

➢ **Forgotten Account Page**

- This page is accessible to **guests** only (non-logged-in-users).
- Only **registered emails** that are **not banned** can be used.
- Retrieve the **user's personal security question** and allow them to provide an **answer**.
- Ensure the answer is **correct** before proceeding.
- The user must fill out the **form** with the following required details: **new password**.
- Validate the **new password can't be the same** as **the old one**.
- Provides a **link** to the **Landing Page**.

➢ **Component Layout**

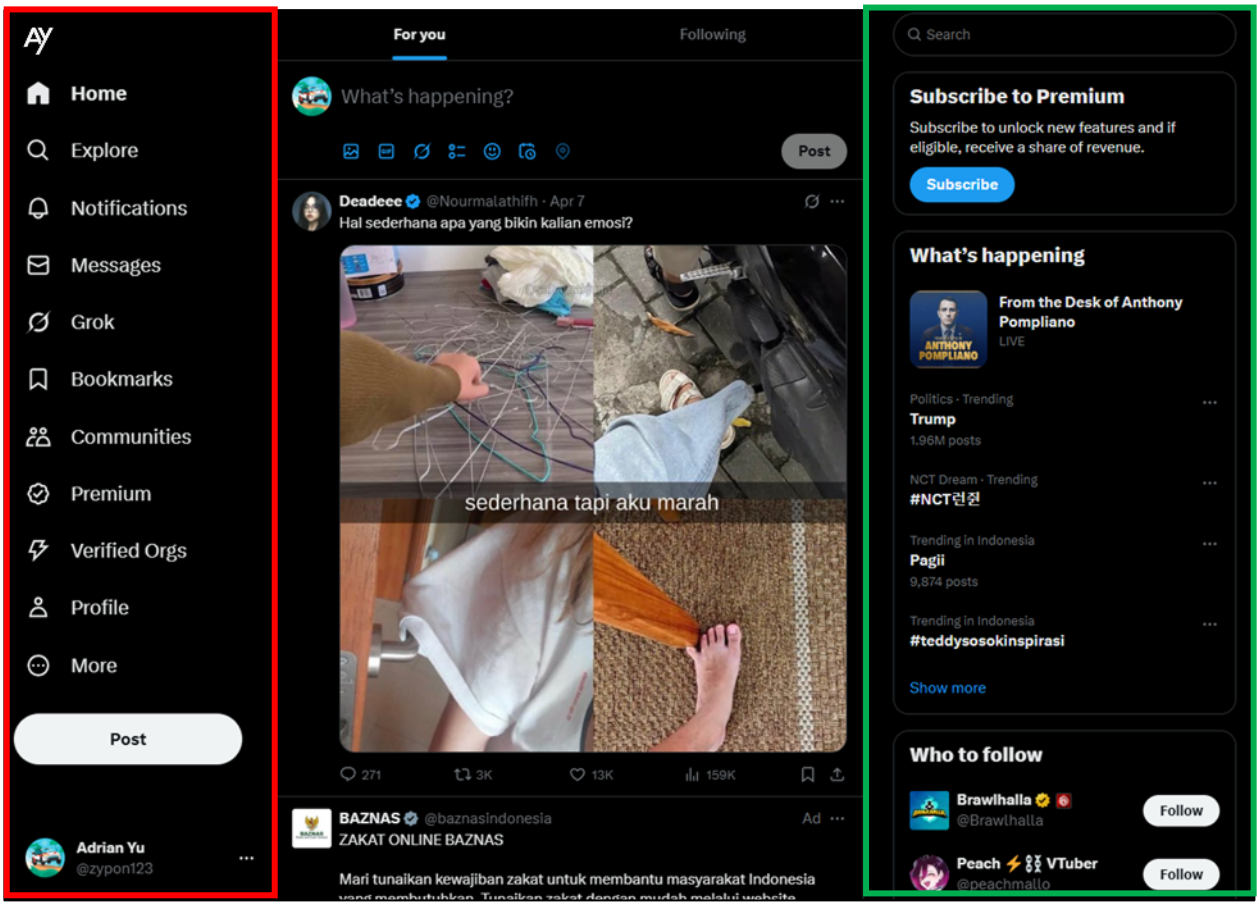

Figure 7. Component Layout

- The **layout** must strictly follow these **rules**.

- The **left-side bar** is designated for **navigation**, enabling users to navigate between pages. It also shows the **user's information**.
- The **right-side bar** of the page is designated for displaying the **search bar**, **a "What's happening"** section, and **a "Who to follow"** section.
- The **left and right sidebars** are **independently scrollable**, allowing users to navigate through their contents without affecting the scroll position of the main content area.

➢ **Left-Side Bar**

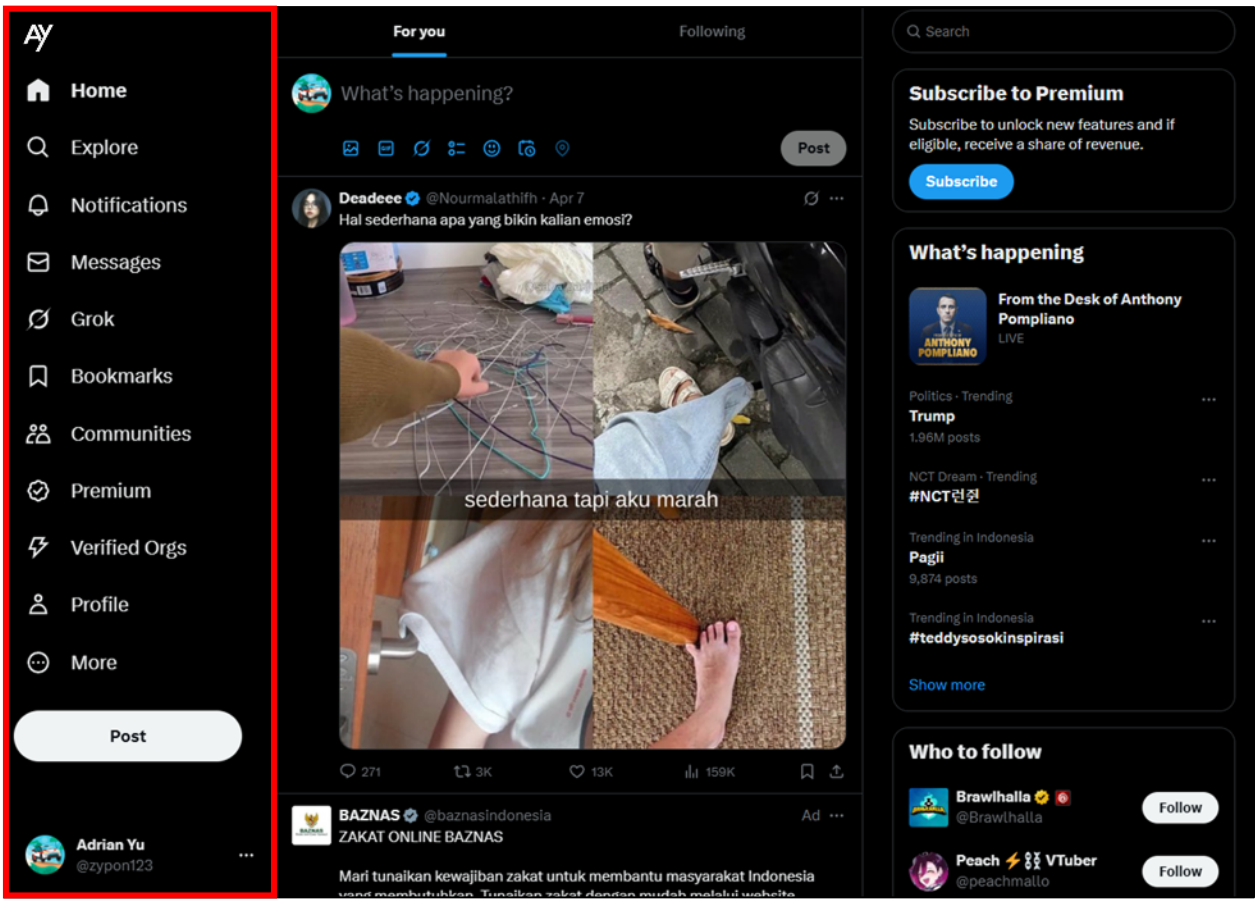

Figure 8. Left-Side Bar

- Contains links to the following pages:
  - **Home Page**
  - **Explore Page**
  - **Notifications Page**

- **Messages Page**
- **Bookmarks Page**
- **Communities Page**
- **Premium Page**
- **Profile Page**
- **Settings Page**

- Contains "**Post**" button. When clicked, it opens a modal where the user can **create a new thread** by entering the content.
- Contains a **button** to switch **between dark** and **light mode**.
- Contains the **user's information**, including their **name**, **username**, and **profile picture**.
- Clicking the profile picture reveals options to **log out**.

➢ **Create New Thread Modal**

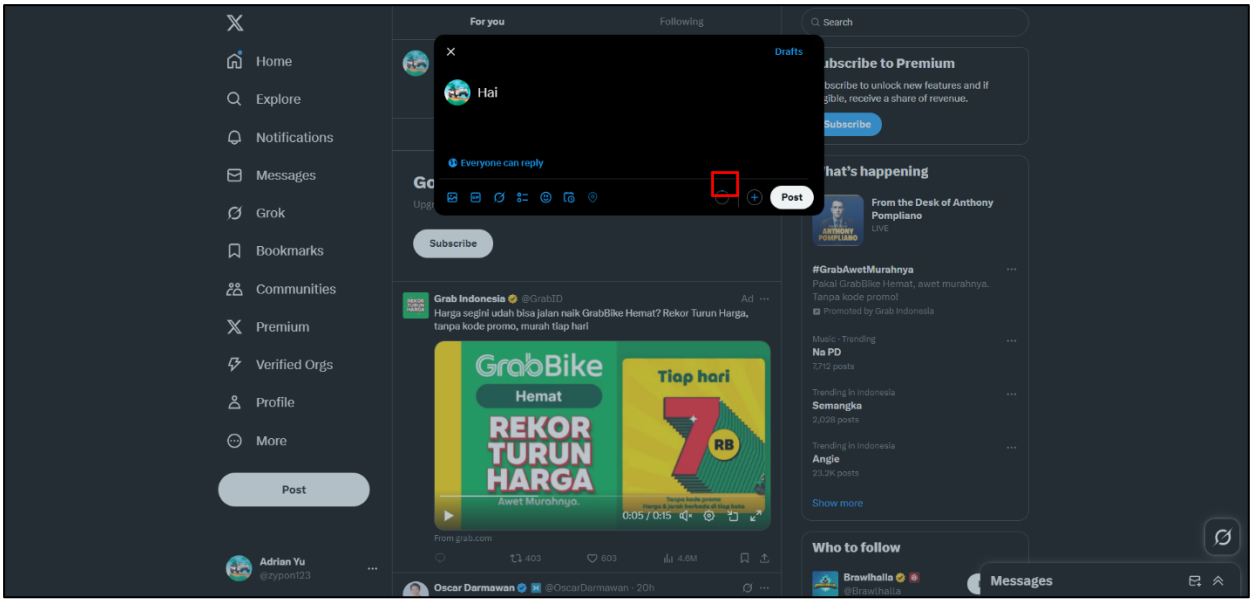

Figure 9. Create New Thread Modal

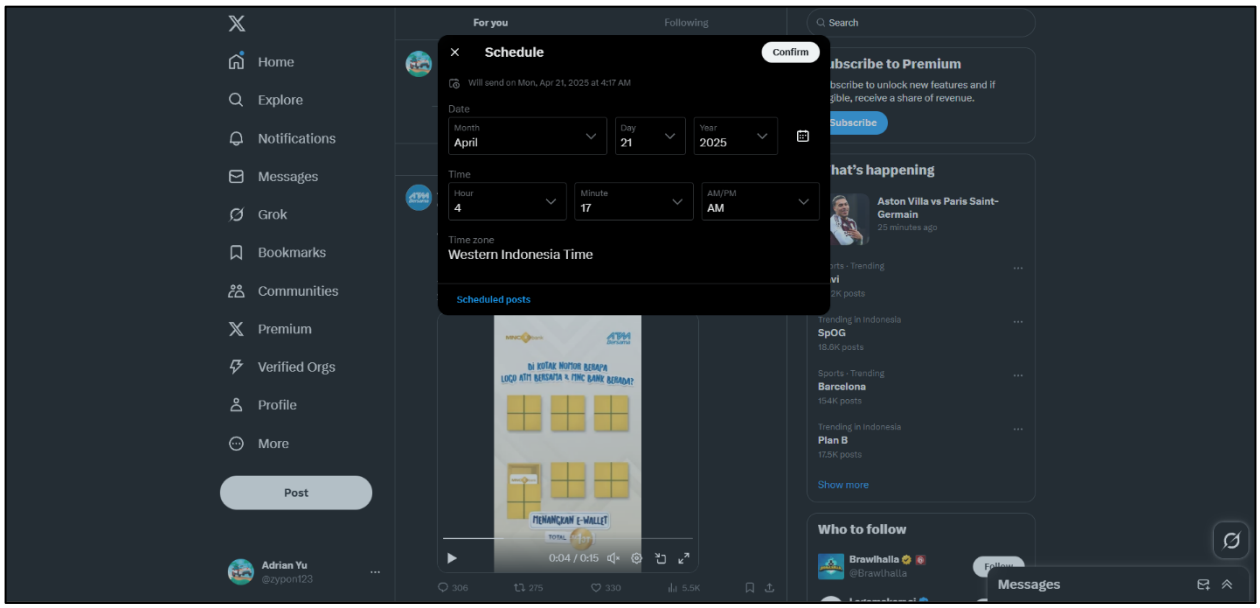

Figure 10. Create New Thread Modal (Schedule Thread)

- The user can **create a new thread** by entering the content.
- The user can **attach multiple files**, including **images**, **GIFs**, and **videos**, as well as **create polls**. They can also choose who can **reply** to the **thread** or **participate** in the **poll**: **Everyone**, **Accounts You Follow**, or **Verified Accounts**.
- The user can add **categories** to the **thread**, helping categorize the thread.
- The user can **schedule** the **thread** to be posted at **a specific date and time** they choose.
- The user can choose to post the thread either **personally** or for a **community**. If posting for a **community**, the user must **first select the community**.
- Display a **word counter** while typing the content. The word counter should appear as a **circular progress indicator** that updates live in **real-time** as the user types.
- Any thread posted by an **admin** user will be **recognized** and **displayed** as an **advertisement thread**.

➢ **Right-Side Bar Component**

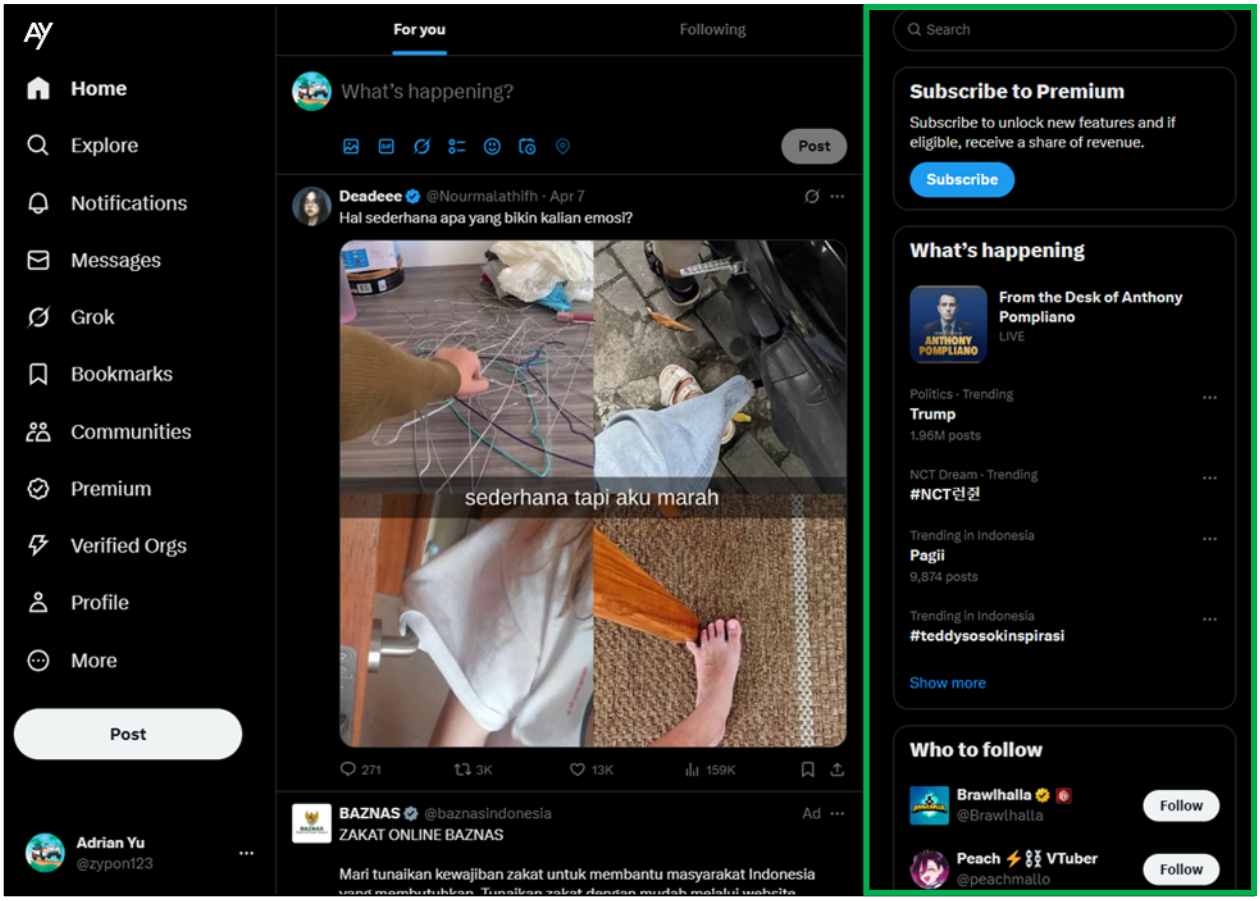

Figure 11. Right-Side Bar

- Contains **a search bar**. When clicked, the search bar displays the user's **3 (three) most recent searches**. Recent search data **must not be saved to the database**.
- Contains "**Clear All**" button to clear the **user's search history**.
- When the user presses "**Enter**" or click the **search icon**, the app redirects to the **Explore Page** with the **search query**.
- Contains **ads** asking users to **subscribe** to the **premium**.
- Contains a **button** that redirects users to the **Premium Page**.
- Contains a "**What's happening"** section showcasing the **top 5 (five) trending hashtags** along with the **count of threads** that uses the hashtags.
- Contains a "**Who to follow"** section showcasing the **top 3 (three) accounts** with the **highest follower count**.

- **Rich Text**
  - **User mentions** must have a **different color** from normal text and when clicked will navigate to the **Profile Page of the tagged user**.
  - **Hashtag** must have a **different color** from the normal text and when clicked will navigate to **Explore Page** to **search for thread with such tag**.
  - "**@**" is used to denote **User mentions** (ex: Hello @adrian!)
  - "**#**" is used to denote **Hashtag** (ex: #minecraft, #valorant, #cr7) and must not contain any space.

- **Verified User**
  - Verified users **always have** a **blue checkmark** displayed **beside their name across the platform**, indicating their verified status.

➢ Home Page

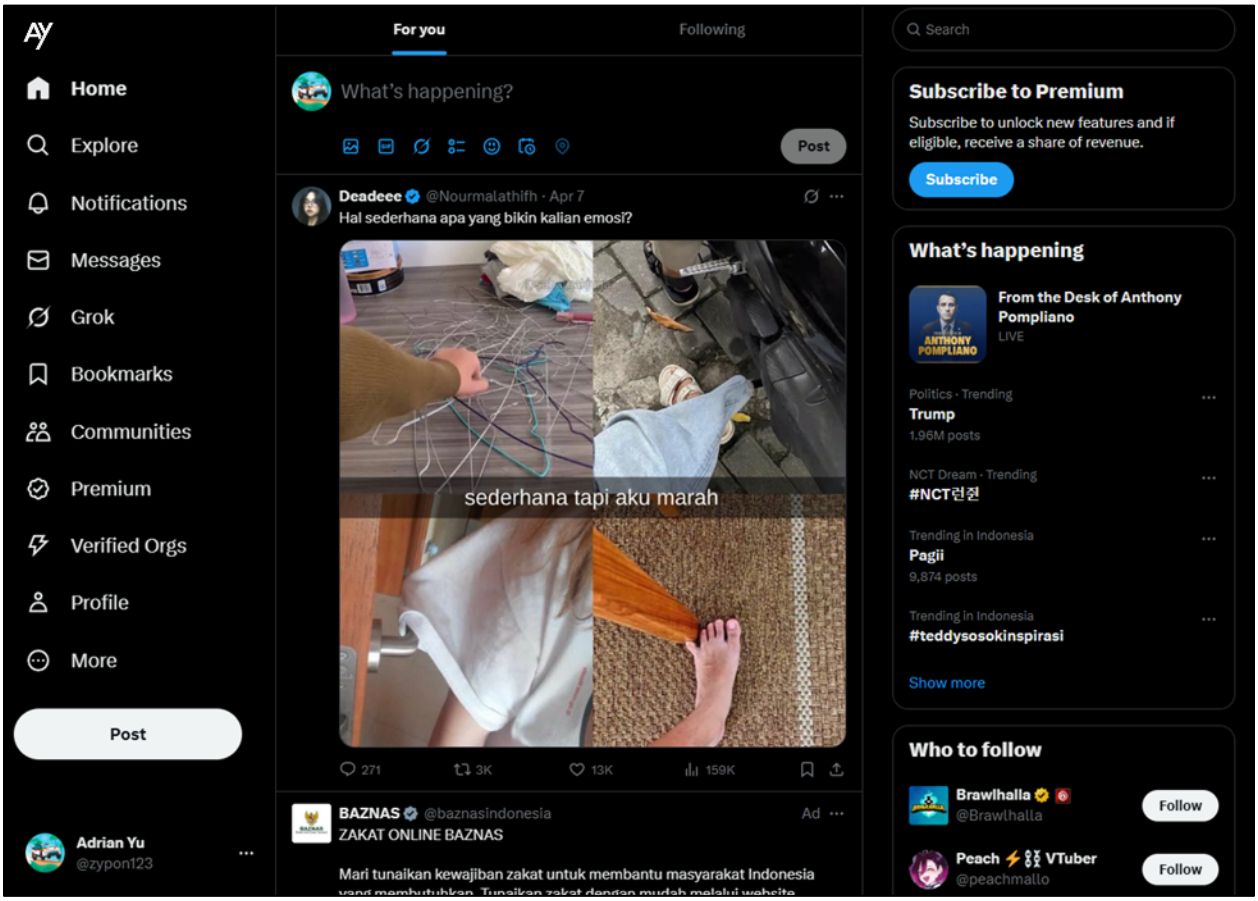

Figure 12. Home Page

- This page is accessible to **authorized users** only (logged-in-users).
- There are 2 (two) main tabs: **For you** and **Following**.
- In each tab, at the very top, there is a **form** to **create a new thread**. The requirement is the **same** as the **Create New Thread Modal**.
- In the "**For you**" tab, threads are displayed with **infinite scrolling**, starting from the **most recent** threads. If a thread is from a **community**, it will **only appear** if the user is a **part of that community**, ensuring a personalized feed of relevant content.
- In the "**Following**" tab, threads made by the user's following accounts are displayed with **infinite scrolling**, starting from the **most recent** threads.
- After **every 5 (five) thread**, an **advertisement thread** will be displayed to users.

- While **threads** are loading, display a **loading skeleton UI** to indicate content is being fetched.

➢ Explore Page

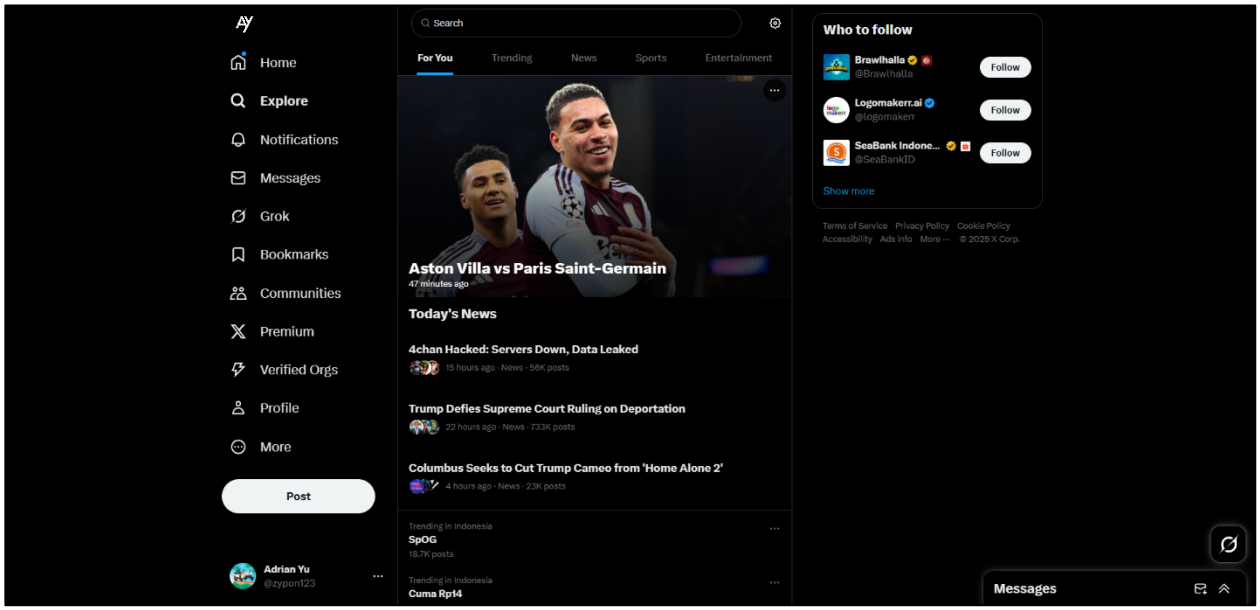

Figure 13. Explore Page

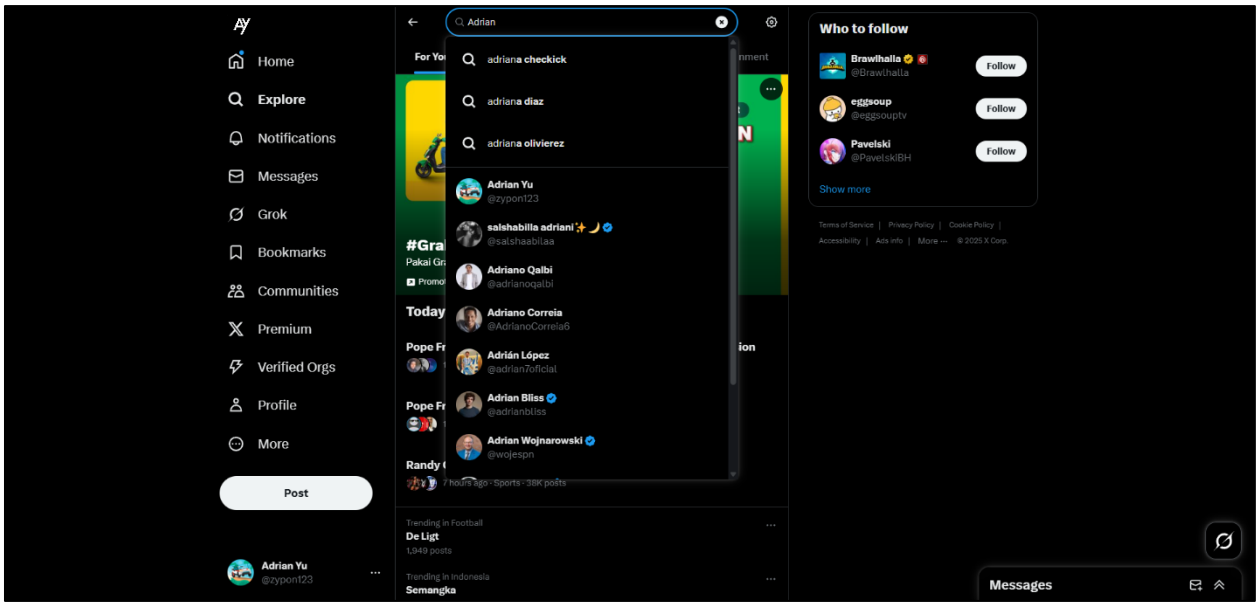

Figure 14. Explore Page (Searching)

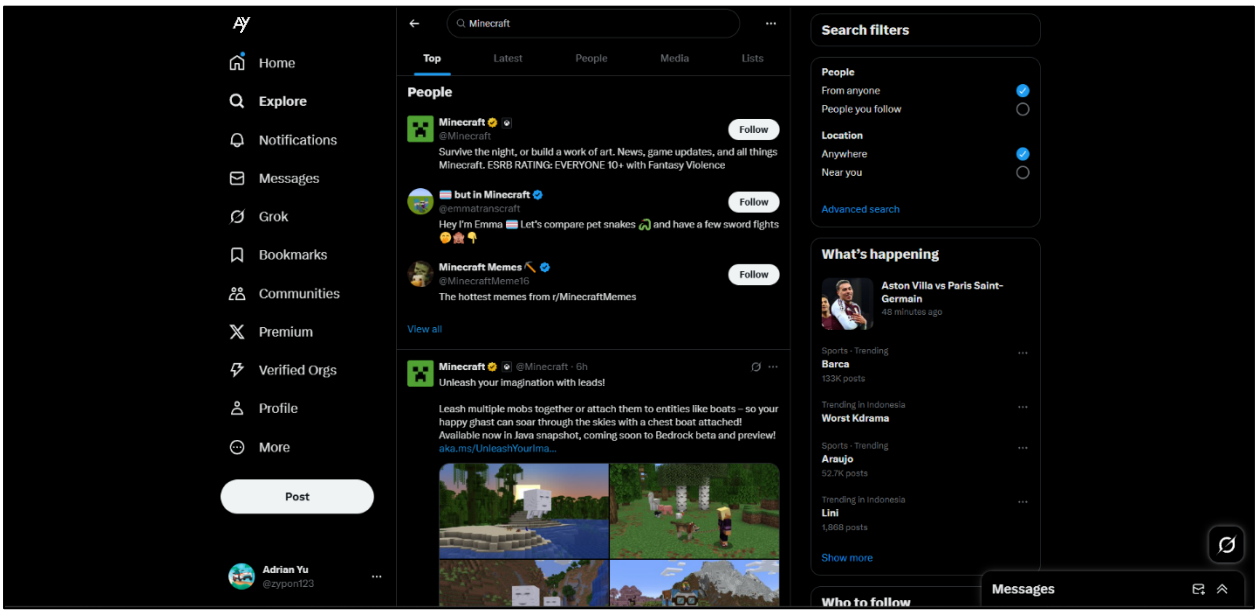

Figure 15. Explore Page (Searched)

- This page is accessible to **authorized users** only (logged-in-users).
- Contains **a search bar**. When clicked, the search bar displays the user's **3 (three) most recent searches**. Recent search data must be saved locally, not in the database.
- Contains "**Clear All**" button to clear the **user's search history**.
- Contains a **filter** that allows user to **filter search results** based on "**People you follow**", "**Verified accounts only**" or "**Everyone**".
- Contains a **filter** that allows users to **filter search results** based on **thread categories**.
- When **no search query is entered**, the user sees the **trending tab** with the **top 10 (ten) popular hashtags**. Clicking on a hashtag shows a **list of threads with that hashtag**, **sorted descending** by the number of **likes**.
- Related must be determined using **Damerau-Levenshtein distance** for accurate, fuzzy matching.
- While typing a **search query**, display recommended user profiles that are **related** to the **search query**. Use a **debounce** mechanism to optimize performance.
- When the user presses "**Enter**" or click the **search icon**, the results are shown across **4 (four) tabs**:
  - **Top**

Display the **top 3 (three) user profiles** based on the follower count that related to the search query, with a "**View All**" button that redirects to the **People** for a complete list of results. Below, show a **list of threads** that include the search keyword in their content.

- **Latest**

  Display the **most recent threads** that related to the search query, sorted in **descending order**, with the newest threads appearing first.

- **People**

  Display **user profiles** related to the search query, showing the **user's name**, **username**, **profile picture**, and **bio**. Include a "**Follow**" button to allow users to follow others directly. Clicking a profile will navigate to that **user's profile page**. Implement **pagination** for a limited number of profiles per page, with options to **display 25, 30, or 35 profiles per page**.

- **Media**

  Display **media content** (images, GIFs, and videos) from threads related to the search query in **a responsive grid** with **3 (three) media items per row** and **infinite scrolling**. Clicking on any media item will navigate the user to the **corresponding Thread Detail Page**.

- **Communities**

  Display **a list of communities** related to the search query. Each entry should include the **community's name**, **description**, and **logo**. Provide a "**Request to Join**" button for users to send a join request to the community. Clicking on a community will navigate the user to the **corresponding Community Detail Page**. Implement **pagination** with options to **display 25, 30, or 35 communities per page**.

- While **search results** are loading, display a **loading skeleton UI** to indicate result is being fetched.

➢ Notifications Page

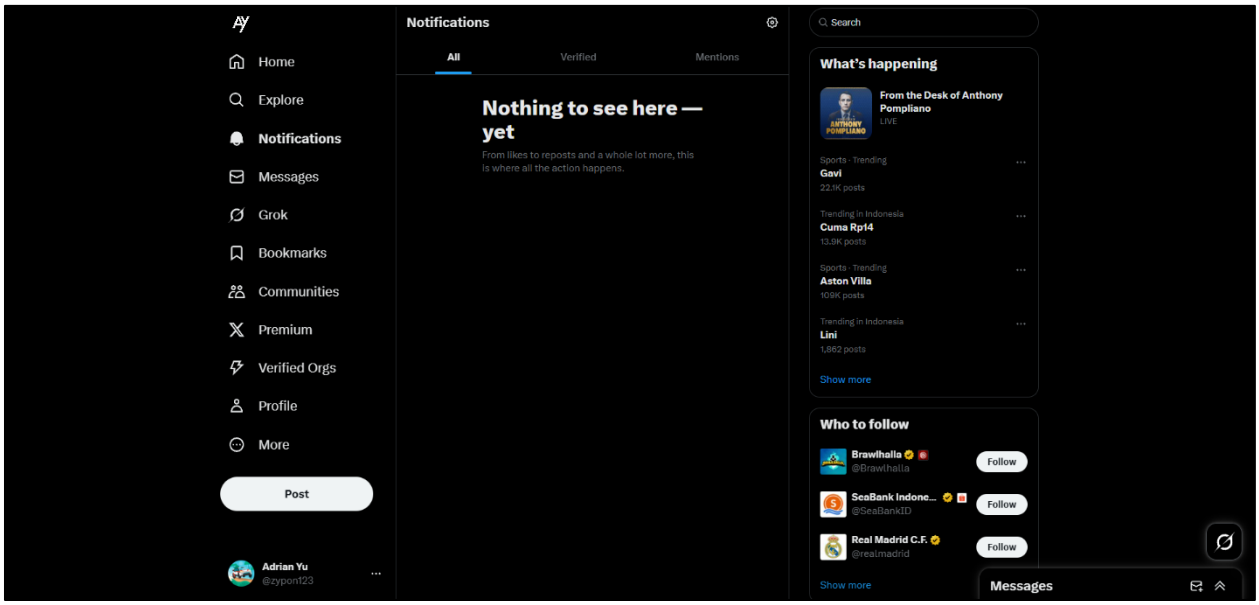

Figure 16. Notifications Page

- This page is accessible to **authorized users** only (logged-in-users).
- Ensure that the **notifications** are delivered in **real-time**.
- Each **notification** must also be sent to the **user's email address**.
- There are 2 (two) main tabs:
  - **All**

    Display notifications when **other users like, repost, or follow the user**. Each notification includes details such as the **user who performed the action** and, if applicable, **the thread that was liked or reposted**. Clicking a notification will take the user directly to the **corresponding Thread Detail Page**.
  - **Mentions**

    Display notifications when other users **mention the user's username** in their threads. Clicking on a notification will take the user directly to the **corresponding Profile Page of the user who mentioned them**.

➢ Messages Page

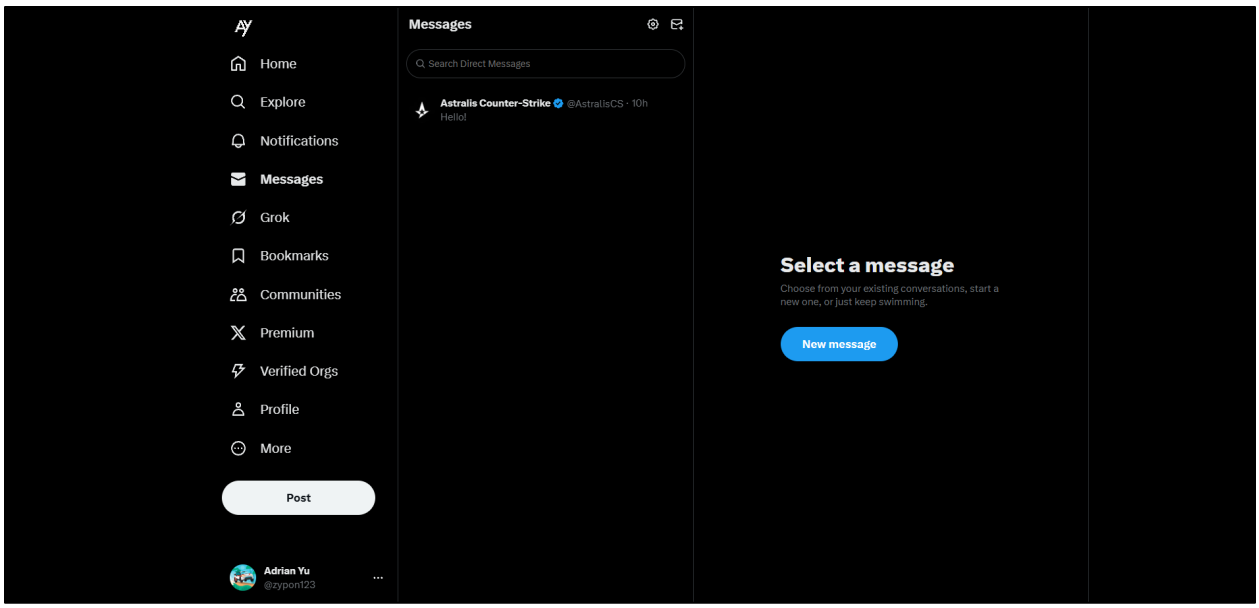

Figure 17. Messages Page (No Active Chat)

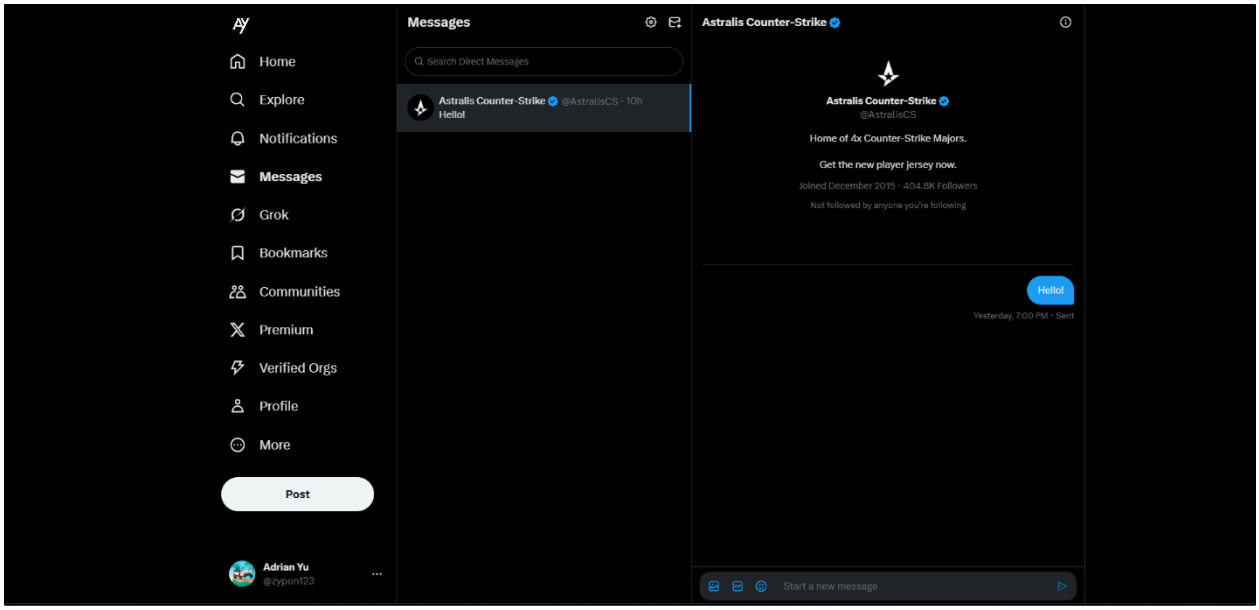

Figure 18. Messages Page (Active Chat)

- This page is accessible to **authorized users** only (logged-in-users).
- The user can message either an **individual** or **a group**.
- To chat with a group, the user can **select participants**.
- The user can send **images**, **GIFs**, and **videos** within the chat.
- Each chat bubble must include a **timestamp**.

- The user can **switch** between different **active chats** easily.
- The user can **delete conversations** from their **chat history**.
- The user can **add or remove people** from an existing group.
- The user can **unsend a message** within **1 minute** of sending it.
- The chat must be updated in **real-time**.
- The user can also **search** through their chats to find specific messages or conversations.

➢ **Bookmarks Page**

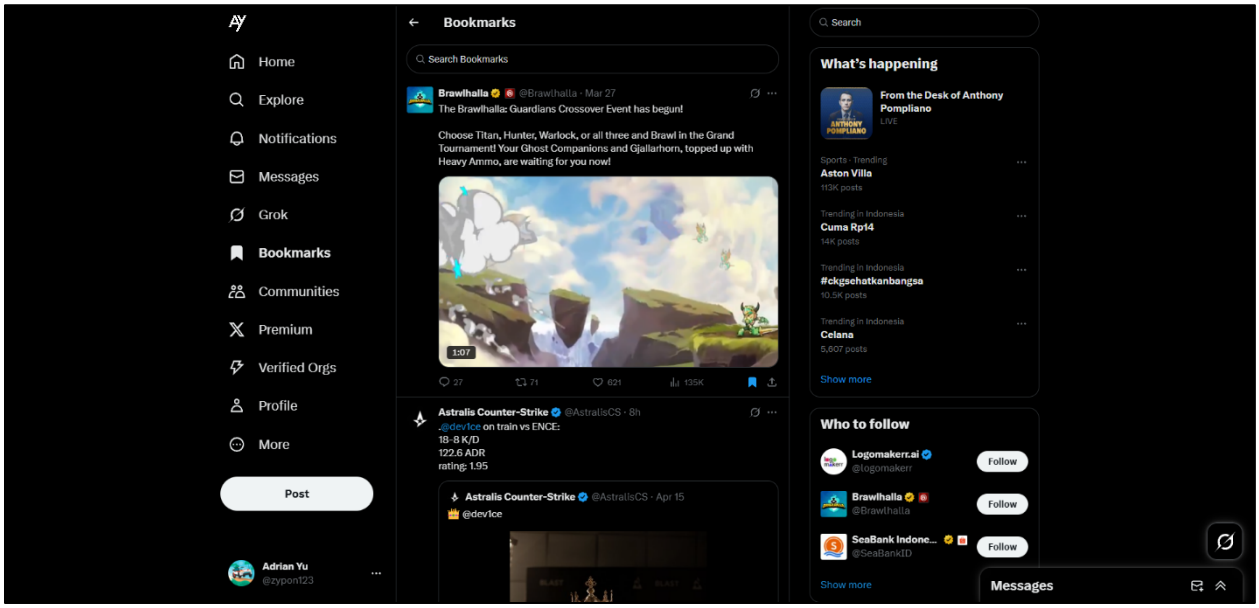

Figure 19. Bookmarks Page

- This page is accessible to **authorized users** only (logged-in-users).
- Display the **search bar**, allowing the user to easily search through all their **bookmarked threads**.
- The user can view all **threads** they have **bookmarked**.

➢ **Communities Page**

- This page is accessible to **authorized users** only (logged-in-users).
- Display a **list of communities the current user has joined**, with pagination options to display 25, 30, or 35 communities per page.

- Display a **list of communities the user has pending join requests for**, with pagination options to display 25, 30, or 35 communities per page.
- Display a **list of available communities on the platform**, with pagination options to display 25, 30, or 35 communities per page.
- Each community entry should include the **community's name**, **description**, **logo, and categories**.
- Contains a "**Request to Join**" button to allow users to **send a join request** to communities.
- Clicking on a **community** should navigate to its **detailed page**.
- Users can **search** for communities by query.
- Users can **filter** the search results by **categories** like gaming, sports, food, etc., to find communities based on their **interests**.
- Contains "**Create Community"** button that redirects users to the **Create Community Page**.

➢ **Community Detail Page**

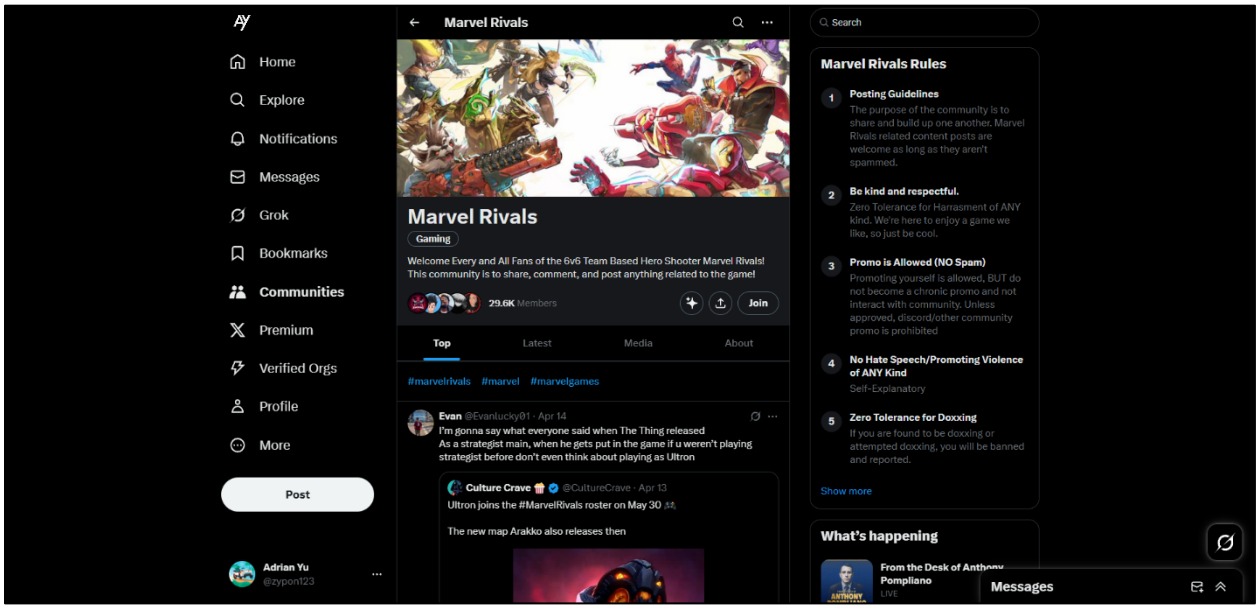

Figure 20. Community Detail Page

- This page is accessible to **authorized users** only (logged-in-users).
- Display the **community's name**, **description**, **logo**, **categories** and **banner**.

- Contains a "**Request to Join**" button to allow users to **send a join request** to communities.
- Show the **member count** with a clickable link to open a modal displaying the **list of members**. The modal has two tabs: **Members** and **Moderators**. Includes a **search** feature and **pagination** with options to display 25, 30, or 35 members per page. Clicking a **profile** navigates to that **user's profile page**. Owners can **promote members** to **moderators** and **demote moderators** back to **members** directly from the Members tab.
- There are 5 (five) main tabs:
    - **Top**

      Display the **top 3 (three) members** in the community based on their **follower count**. Below, show threads posted within the community, sorted in **descending order** by the number of likes to highlight the most engaging content.
    - **Latest**

      Display threads posted within the community, sorted in **descending order** by **creation time**, with the newest threads appearing first.
    - **Media**

      Display **media content** (images, GIFs, and videos) from threads posted within the community in **a responsive grid with 3 (three) media items per row** and **infinite scrolling**. Clicking on any media item will navigate the user to the **corresponding Thread Detail Page**.
    - **About**

      Display key information about the community, including the **creator's name**, **creation date**, **community rules**, and **a list of current moderators**.
    - **Manage Members**

      Exclusive to moderators. On this tab, **moderator** can **accept** or **reject join requests**.

➢ **Create Community Page**

- This page is accessible to **authorized users** only (logged-in-users).
- The user must fill out the **form** with the following required details: **community name, description, icon, categories, banner and rules**.

- Upon submission, the **request** will enter a **pending** state and will **require admin approval before the community is created**.

➢ **Premium Page**

- This page is accessible to **authorized users** only (logged-in-users).
- Display the main benefit of being a premium member is receiving a **blue checkmark next to the user's name** for increased recognition.
- To become a premium user, the user must provide their **national identity card number**, a **reason** for wanting to be verified, and a picture of their **face** for verification.
- Once submitted, the **admin** will **review and verify the request**.
- Consider the **security** when **storing the identity card number**.

➢ **Profile Page (Own Profile)**

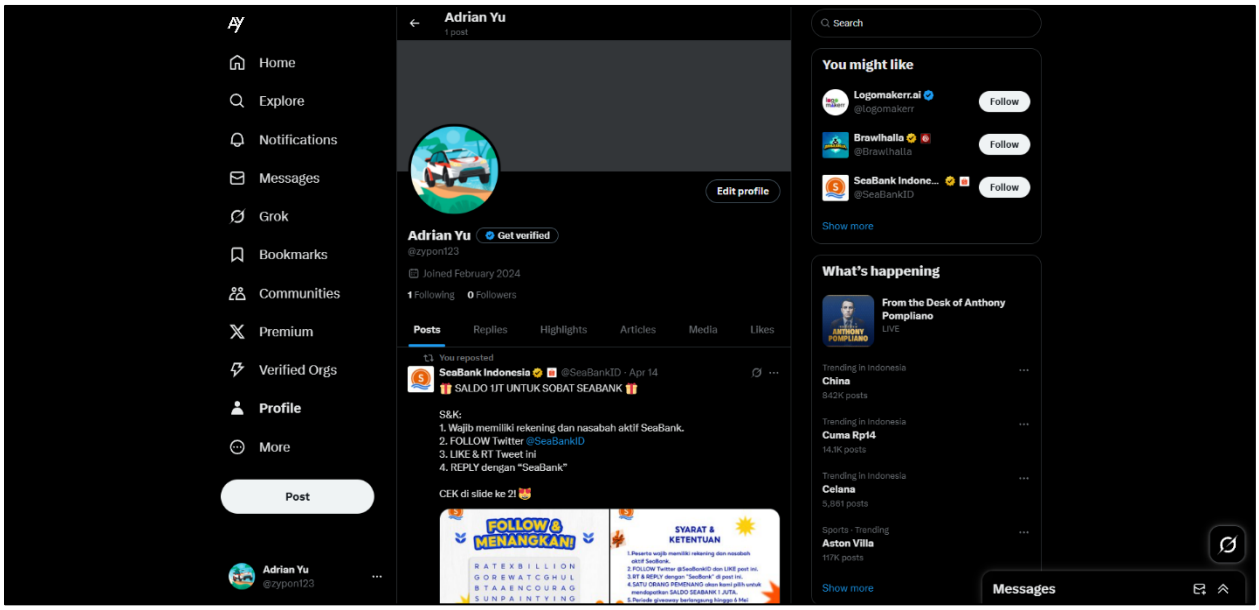

Figure 21. Profile Page (Own Profile)

- This page is accessible to **authorized users** only (logged-in-users).
- Display the **user's name**, **username**, **joined when**, **bio**, **following count**, **follower count**, **profile picture** and **background banner**.

- Allow users to **update** their **personal information**, including **name**, **username**, **email**, **password**, **gender**, **profile picture**, **date of birth**, and **background banner**. A modal is provided for editing, and **validation** is the **same as in the registration process**.
- When the **profile picture** is clicked, a **preview modal** is shown to display the image in a larger view.
- There are 4 (four) tabs:
    - **Posts**

      Display **a list of threads** the user **has posted** from the **newest**. Each thread will be clickable, leading to the **Thread Detail Page**. Users can **pin a thread** to their profile so that it appears at the top of the **Posts** section.
    - **Replies**

      Displays **the content of the user's replies**. Each reply is clickable and will navigate to the **specific thread where the user replied**. Users can also **pin a reply** to their profile to highlight it at the top of the **Replies** section.
    - **Likes**

      Display **a list of threads** that the user **has liked** from the newest. Each thread will be clickable, leading to the **Thread Detail Page**.
    - **Media**

      Displays **media content** (images, GIFs, and videos) from user's threads across the platform in a **responsive grid with 3 (three) media items per row**. Clicking on any media item will navigate the user to the **corresponding Thread Detail Page**.

➢ Profile Page (Other's Profile)

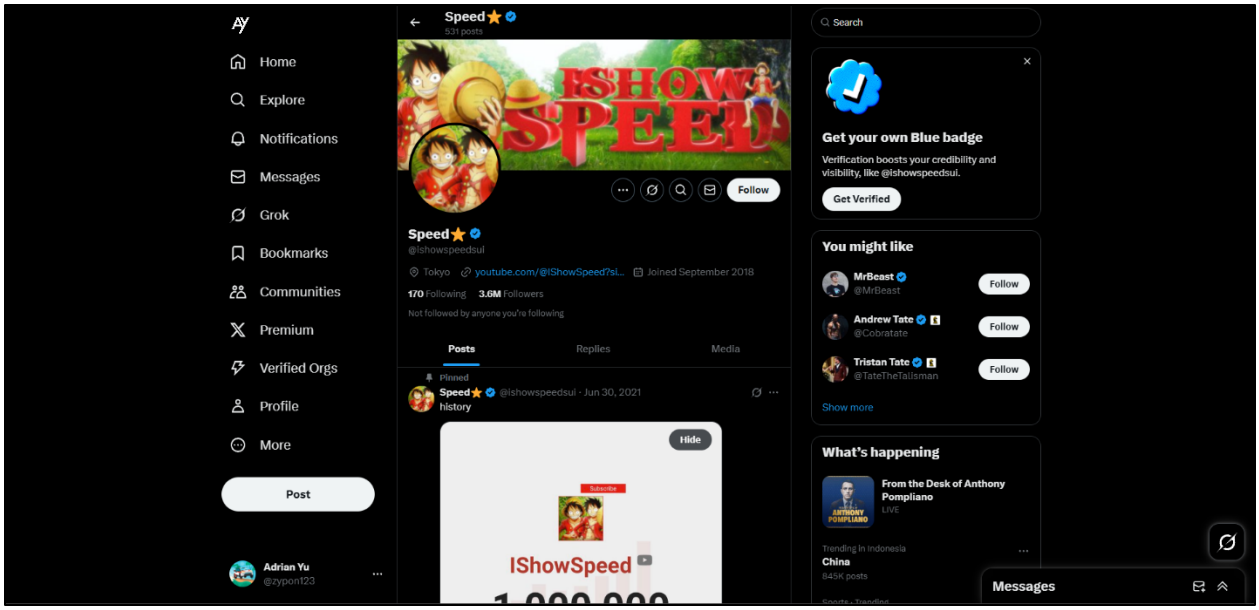

Figure 22. Profile Page (Other's Profile)

- This page is accessible to **authorized users** only (logged-in-users).
- Display the **user's name**, **username**, **joined when**, **bio**, **following count**, **follower count**, **profile picture** and **background banner**.
- Include a "**Follow"** button that allow users to **follow** the profile directly.
- Include a "**Report**" button that allows users to **report** the account. When a report is submitted, users **must give a reason** for the report. It will be **sent to the admin** for **review**.
- Include a **"Block"** button that allows users to **block** the account. Once an account is blocked, the user will **no longer see any actions** from the blocked account, and the blocked account will **no longer see the user's actions** on the application.
- Any **private user's actions** can't be seen by **users** who **don't follow** them.
- There are 3 (three) tabs:
    - **Posts**

      Display **a list of threads** the user **has posted** from the **newest**. Each thread will be clickable, leading to the **Thread Detail Page**.
    - **Replies**

Displays **the content of the user's replies**. Each reply is clickable and will navigate to the **specific thread where the user replied**.

- **Media**

  Displays **media content** (images, GIFs, and videos) from user's threads across the platform in a **responsive grid with 3 (three) media items per row**. Clicking on any media item will navigate the user to the **corresponding Thread Detail Page**.

➢ **Settings Page**

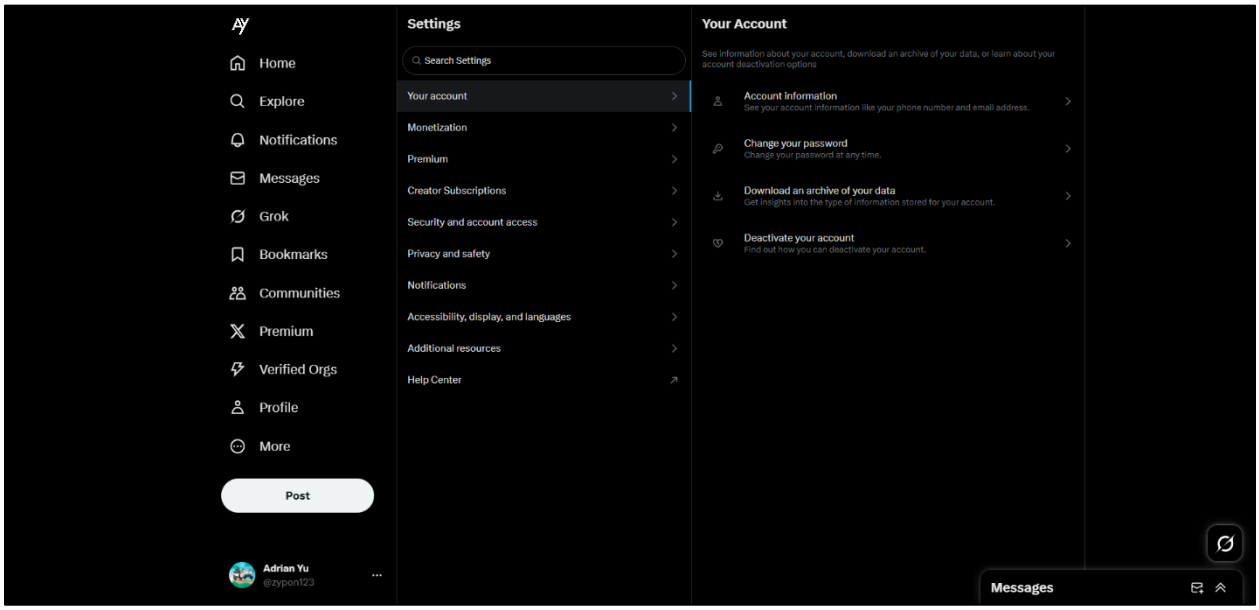

Figure 23. Settings Page

- This page is accessible to **authorized users** only (logged-in-users).
- There are 5 (five) tabs:
  - **Security**

    Users can choose to set their account to either **private** or **public**, controlling who can view their content.
  - **Display**

    Users have the option to customize their viewing experience by adjusting the **font size** and **font color** on the website.
  - **Your Account**

    Users can **deactivate** their account at any time from the settings.

- **Blocked Accounts**

  Users can **view their blocked accounts** and **unblock** them at any time if they choose to do so.

- **Notification Preferences**

  Users can **customize** which notifications they **receive**. Each notification type includes a **toggle** to **enable** or **disable** it based on preference: **Like**, **Repost**, **Follow**, **Mentions**, and **Community**.

➢ **Thread Component**

Figure 24. Thread Component

- Each **thread** should display the **user's name**, **username**, **profile picture**, a **relative timestamp** indicating when it was posted (ex: “2h ago”, “1d ago”), and **categories**, along with the **thread's content**. If the thread contains **more than one media item**, display the media in a **grid layout**.
- Below the content, include **interactive buttons** so user can **reply**, **repost**, **like**, and **bookmark** the thread.
- Use **different icon colors** to **visually** indicate if the user has already **replied**, **reposted**, **liked**, and **bookmarked**.

- Users can **repost content** from other threads. Reposted threads will include **reposting text above the original thread content**.

Figure 25. Thread Component (Reposted Thread)

- Users can **undo** their actions, such as **un-liking**, **un-reposting**, or **un-bookmarking** a thread.
- Users can **delete** the **threads** they have posted by clicking the **three-dot icon**, which will present an **option to delete** the thread.
- **Clicking anywhere** on the thread will **navigate** the user to the **Thread Detail Page**.
- Users can **pin** a **thread** to their profile by **clicking** the **pin icon**.
- **Pinning** is only **available** when **viewing** the **thread** from the **user's own profile page**. For more details, please refer to the **Profile Page (Own Profile)** section.
- Users can **send threads** in **messages** by clicking the **send icon**. After clicking, users need to **choose** a **recipient** and confirm before the thread is sent.

➢ Thread Detail Page

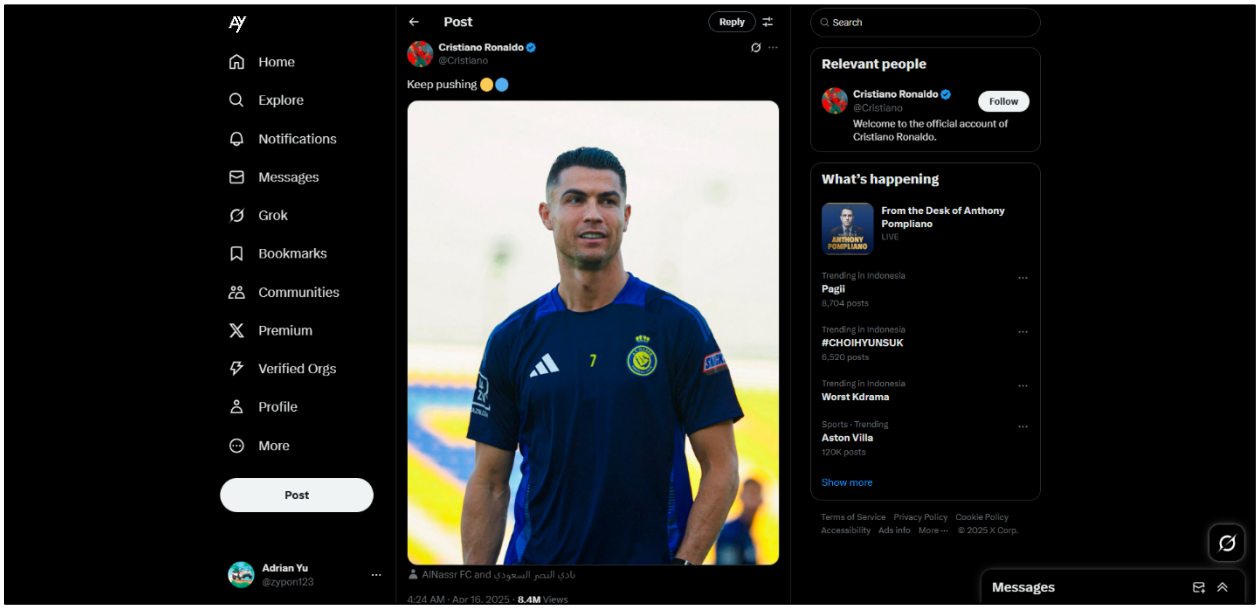

Figure 26. Thread Detail Page

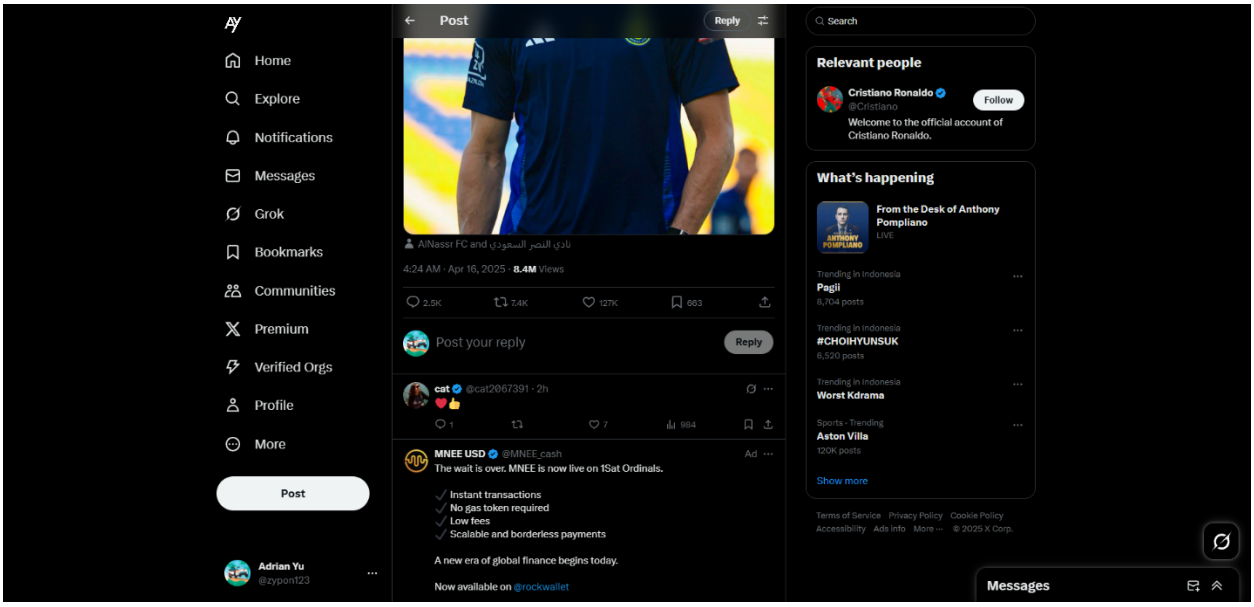

Figure 27. Thread Detail Page (Reply Section)

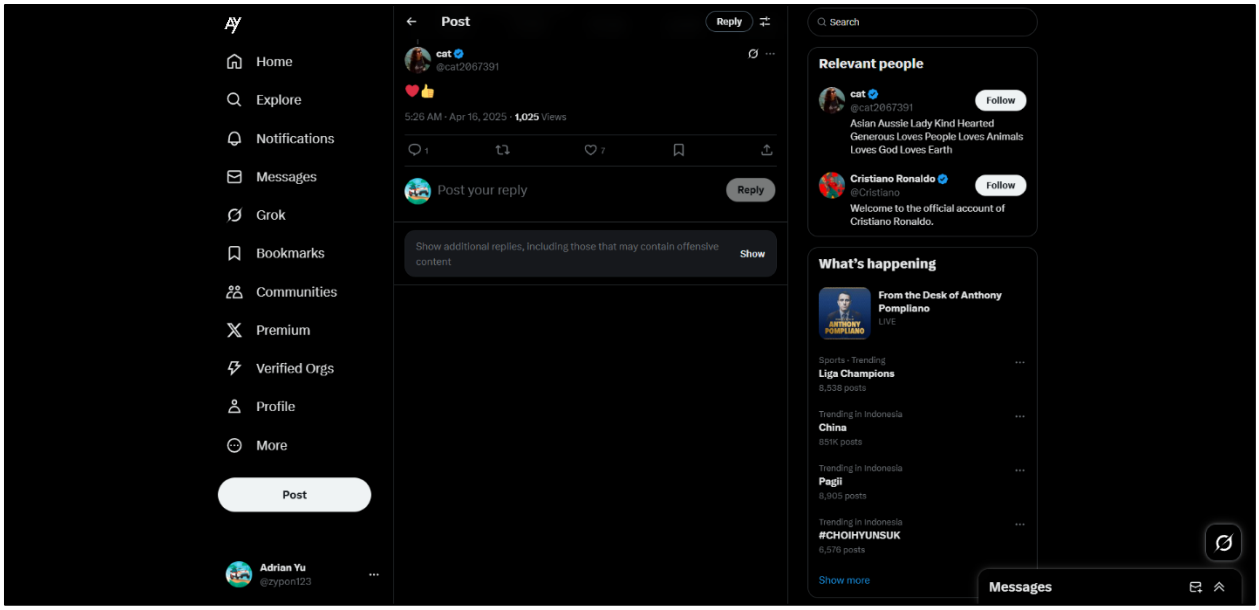

Figure 28. Thread Detail Page (Thread Reply)

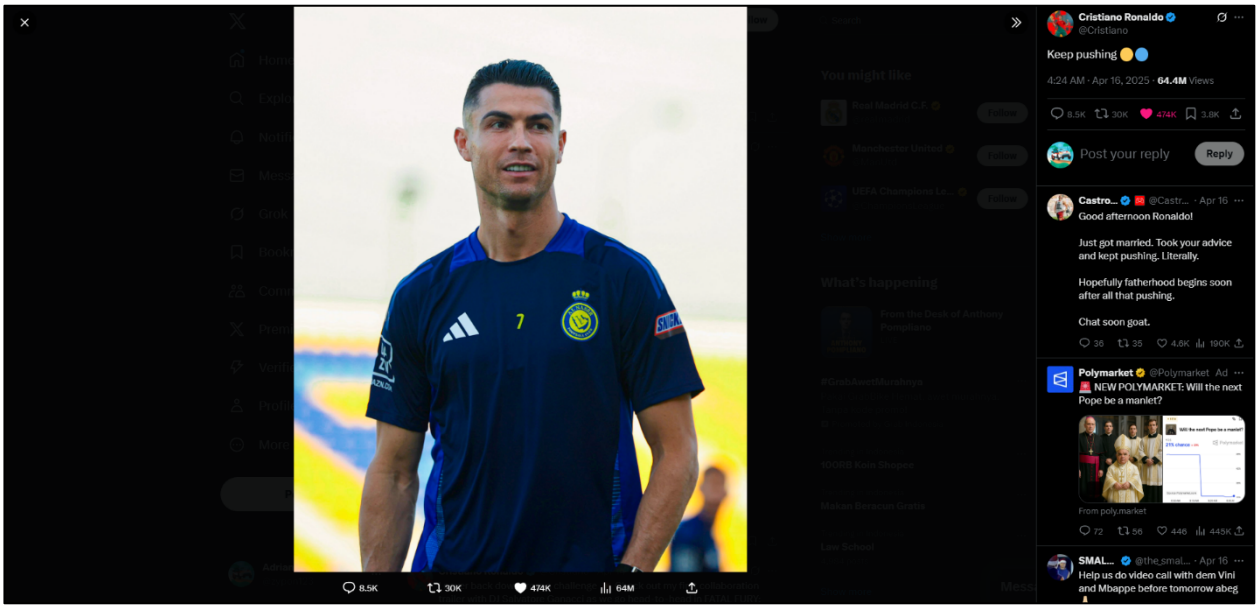

Figure 29. Thread Detail Page (Media Clicked Overlay)

- Display the **full thread component**.
- Below the thread, provide a **reply section** where users can **write** their replies. The reply form should have the **same specification** as the "**Create New Thread**".
- Shows a **list of existing replies** from other users, creating a **conversation thread** under the **original thread**.

- Users can **delete** the **replies** they have posted by clicking the **three-dot icon**, which will present an **option to delete** the reply.
- Users can also **reply**, **repost**, **like**, and **bookmark** individual replies.
- Clicking on a reply will **navigate** to a dedicated **Thread Detail Page** for that **specific reply**.
- When the media is clicked, display a **Media Overlay View**. While in this overlay, users **can still perform all regular interactions** such as replying, reposting, liking, bookmarking, and replying **without leaving the overlay**. If there is **more than one media item**, display **navigation arrows** within the overlay to allow users to **navigate between media items**.

➢ **Admin Page**

- This page is accessible to **admin users** only.
- Admin can **view all users** and able to **ban** or **unban** the **user**.
- Admin can do **send newsletters** to users that **subscribed**.
- Admin can **review** and **accept or reject requests** for creating **new communities**.
- Admin can **review** and **accept** or **reject requests** from users who want to become **premium**.
- Admin can **review** and **accept** or **reject report requests** from users. If the report is **accepted**, the reported user will be **banned**.
- If a request is successfully **accepted**, the **system will send an email** to the user **notifying** them about the **acceptance**.
- Admin can **view**, **create**, **update**, and **delete thread categories**.
- Admin can **view**, **create**, **update**, and **delete community categories**.

➢ **Notes**

- All features that have been made that are not listed will not be graded.

Good Luck