

NETS 2120

InstaLite Final Report

Team 2Y2B

Pulkith Paruchuri, Pragya
Singh, Claire Wu, Jingyi Zhao

pulkith@wharton.upenn.edu, pragya7@wharton.upenn.edu,
clwu@wharton.upenn.edu, zhao3@seas.upenn.edu

Report Link:

<https://docs.google.com/document/d/1Hg8uWulvVMRUz1AGRhktCV0mYB-5qPMkprkmnNVLG1Y/edit?usp=sharing>

Overview

InstaLite is an Instagram-like social media application developed as part of the NETS 2120 final project. At a high level, it provides users with a platform to share image and text-based posts, connect with friends, chat, and discover content through a ranking and recommendation system based on adsorption. Services include [Node.js](#), JavaScript, RDS, DynamoDB, S3, EC2, EBS, EMR, Apache Kafka, Apache Spark, React, Python, ChromaDB, and more, versioned using Git. Natural Language Processing and embeddings provide a level of customization, and real-time updates, error handling, push notifications, and UI / UX design enables a level of sophistication. Security measures and auto-scaling ensures for a (almost) production-ready system.

Overview	1
(1) Key Design Decisions	3
Secondary Screens & Features	4
Security and Scalability	6
(2) Technical Implementation Details	6
Technology Stack	6
Key Design Decisions	7
Changes & Lessons	11
(3) Extra Credit Features Implemented	12
Major	12
Moderate	13
Minor	14
Never Realized (but some code is there)	15
(4) Source Files	15
(5) Application Images	16

(1) Key Design Decisions

Our team implemented the following core features.

User Management

- **User Signup:** New users can create accounts by providing a login name, password, first/last name, email, affiliation, and birthday. Passwords are salted and encrypted. Logging in sends the user to the profile page to upload an image and choose tags.
- **Profile Photo Upload:** Users can upload a profile photo which is sent to S3 (with a signed URL because of learner lab security restrictions). The profile photo is embedded and compared with the IMDB actor embedding to find the 5 closest matches, which the user can select from and link to. These matches are stored and can be changed at any time, or unlinked at any time.
- **Hashtag Interests:** Users can specify interest hashtags in their profile, with suggestions for the top-10 most popular tags, as well as search for other hashtags. These hashtags can also be added throughout posts as well. These hashtags are used to rank posts.
- **IMDB Actor Linking:** Users can link their profile to an IMDB actor by matching their selfie/uploaded photo embedding with the 5 most similar actors from a ChromaDB database. Changing the linked actor will make a public post.
- **User Login:** Secure login with user ID and password. We implemented most of the functionality for the reset password option, but had trouble with the email verification and never got it finished :(
-

Main Content Feed & User Page

- **Instagram-Style Feed:** Displays status updates and posts from friends as well as other people. Posts made on another user's feed appear on both. This shows new posts (via a short circuit) before the reranking runs, as well as after the reranking runs.
- **User Posts:** Users can create posts with optional images and text. Posts can also include a list of hashtags.
- **Feed Content Sources:**
 - Posts from friends and self.
 - Posts from other sites (Federated Posts).
 - Posts related to user's hashtag interests.
 - Highly ranked posts from non-friends (based on SocialRank and federated posts).
 - Posts from the Bluesky Feed (via Apache Kafka).
- **Commenting:** Users can comment on any visible post in a threaded manner, which persist for the entire lifetime of the post. Commenting updates in real-time and shows profile and usernames. Clicking on the username goes to the profile page of the user.
- **User Actions:** Users can "like" posts (and unlike them) with a nice animation. They can also bookmark posts (which goes into the profile page to browse later). Users can also share posts via a button that copies a public link to the clipboard.
- **Feed Updates & Ranking:**
 - The feed data is refreshed hourly, fetching new content from Kafka and recent platform activity.
 - An asynchronous Spark job runs hourly to compute post weights using the adsorption algorithm. Logging in/out, and visiting some pages (i.e. the home page or publishing a new post) triggers a script that checks the last update and runs the job if needed.
 - The graph for adsorption includes users, movies, hashtags, and posts, with defined edge types and weights.

-
- **Navigation:** Top menu bar for access to profile, search, chat, and friends pages. Main feed has infinite scrolling, with new posts loading as the user reaches the bottom of their current page.

Secondary Screens & Features

- **Profile Page:**
 - Users can change their linked actor (from top-5 similar) from a modal or unlink overall.
 - Users can update their email and password (with confirmation)
 - Users can update their interest hashtags based on suggestions (top 10), and search for other hashtags.
 - Users can view their bookmarked posts.
 - You can view your posts and comment/interact with them still
 - If you are viewing someone else's profile, you can see send them a friend request (or accept/revoke it). If you are already friends with them, you can chat with them here, or unfriend them.
- **Add/Remove Friends:**
 - Users can manage their friend list (undirected "friend" relationships).
 - Users get a notification when a friend request is sent, accepted, declined, or a friend unfriends them
 - You can see which of your friends are online / offline
 - Users can get up to 10 friend recommendations based on their friends friend's and our ranking algorithm graph structure.
- **Chat Mode:**
 - Friends can chat with each other.
 - You can send invites even when your friend is offline
 - You can create group chats at once (sends invites to all of them), or by adding them to an existing chat (one by one or even in batches)
 - Solo chats are supported as notes to self.
 - Optimistic Messaging for low-network situations
 - Persistent chat history, with new users seeing all previous messages
 - System messages for when a user leaves / joins

-
- Users can leave chats, with notifications for when they leave, as well as the name and icon updating in real-time
 - Creating groups that would have equivalent members of another group (or even batch inviting that would have the same members) is rejected.
 - Lots of error handling for chat invites, chat creation, and chat joining
 - Notifications when an invite is rejected, accepted, received, etc...
 - Invites Persist over time
 - Cannot send duplicate invites
 - Can send both DM's invites and group invites
 - Chat shows profile pictures as well
- **Natural Language Search (Chatbot):**
 - Users can search for people and posts.
 - AI Summary uses RAG over movies as well as SQL over posts and users.
 - Always returns valid links to posts and users.
 - Can send/revoke friend requests from the search page
 - Can filter by only posts, only users or all.

Security and Scalability

- **Security:**
 - Password salting and encryption using Bcrypt
 - Session management to prevent impersonation.
 - Protection against unauthorized modification/deletion of content.
 - All routes are protected.
- **Scalability:**
 - Designed with scalable cloud services (e.g., EC2, RDS/DynamoDB, S3, Kafka, Spark).

(2) Technical Implementation Details

Technology Stack

- **Backend:** Node.js server with express with /api endpoints. Routes have protection. Most routes utilize async functions to allow for multiple connections at once. Hosted on an EC2 instance (with security protocols/groups) that the frontend connects to.
- **Database:** Most data stored in ~20 tables with RDS with a little bit of data stored in DynamoDB, and all images stored in S3 with signed URLs. ChromaDB hosted in docker on a EC2 instance for face embeddings and actor/movie embedding data. Stored with metadata that connects to RDS tables..
- **Large Object Storage:** Amazon S3 for profile photos and post images.
- **Natural Language Search:** GPT 4o with structured outputs and RAG (ChromaDB) used for semantic search and database queries. Data was pre-indexed to provide context (except for SQL queries, of course). Embeddings generated via OpenAI small models.
- **Social News Streaming:** Apache Kafka for the Bluesky movie feed and Federated Posts.
- **Adsorption Ranking:** Apache Spark, with the job invoked via Livy. The Spark job processes data from 3 tables (Federated, Bluesky, local Posts) to build the graph and writes ranked results back to a ranked post table in RDS that the feed service then queries. Comments / links are then written to a separate table that references the IDs in these ranked tables (works even if the posts are reshuffled in the rank table since it's based on the post ID, not the primary key).
- **Frontend:** React and JavaScript, served locally with Node.js.
- **Version Control:** Git and GitHub.

Key Design Decisions

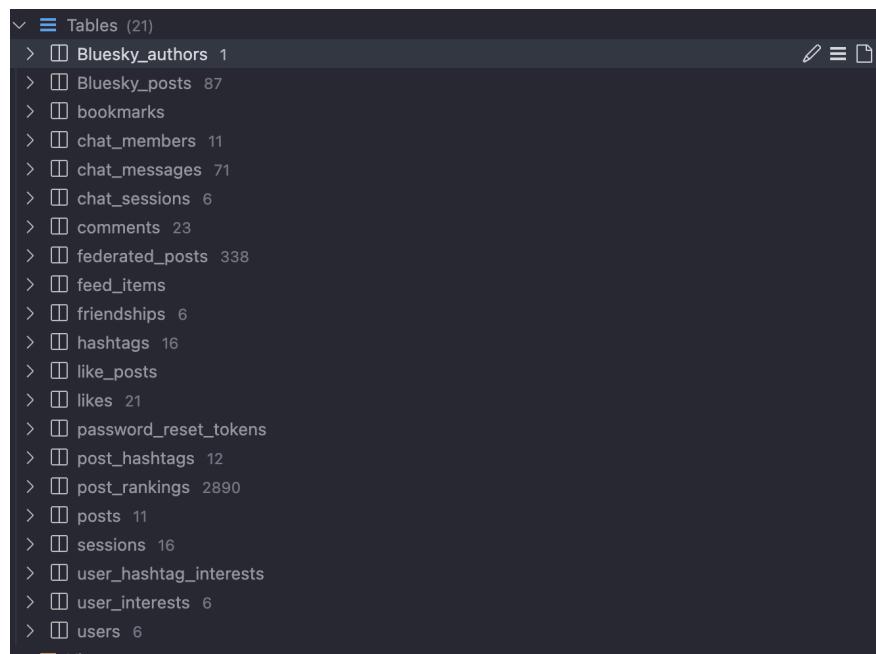
- **Database Schema:** We chose to structure most of our information into tables/SQL, since it allows for efficient and easy joins to get data and run analytics (for things like friend recommendations). We chose to include many tables since it

allows for updates to only need to happen in one place, and we used joins to get the relevant information.

Hosting images in S3 made sense since they're meant for large files and storing them on the server itself is not scalable. We create presigned URLs (because of learner lab restrictions) that we store the link to. Because of the restriction for time limits with signed URLs, we have validation for images to not have the default 'no-image' pictures, and instead show clean initials to replace the image when an image expires.

We chose 2 collections in Chroma, one for facial embeddings, and one for the movie/actor embeddings. For the movie/actor embeddings we created text summaries including the titles, actor name, year, etc... and stored the rest of the data in the metadata. Most tables have primary keys, allowing for indexing and efficient joins.

All posts are first put into separate tables, local posts, Bluesky posts, and federated posts, and then combined during the ranking process and stored in one table. This main table references the primary keys in the other tables, and also contains the adsorption ranking information for each user.



- **Security:** We use local storage and cookies to store session tokens, which are also stored in the database. These tokens are passed with API requests for authentication, and to retrieve things like username/user info on requests easily. These sessions are created on login and last some predetermined amount of time (a few hours) before they expire. All exposed routes require authentication before they can proceed (otherwise rerouted to the login page). All secrets like API keys are stored in a .env file.
- **Chat Implementation:** Chat uses web sockets and optimistic messaging for real-time communication, with all information being stored in an RDS database for persistence. Invites are persisted short-term on the server itself, and expire / are pruned regularly. All updates related to invites or users joining/leaving go through these web sockets, with toasts for alerts. We split messaging into two types: DMs and group chats. DMs display the username / profile picture, while group messages show the users involved and the number of users as the icon. We check for duplicate chats with all current members plus all pending invites plus all new invites, and also check against DMs. Since we have invite persistence, we allow requests to be sent to offline users as well, and just load it when the user comes online. We have system messages for when a user joins/leaves a chat, as well as the ability to send batch invites to users at once. You can create group chats all at once, or by inviting users to an existing chat, as well as having the ability to leave a chat. There is an ‘invite center’ with all pending invites, and once an invite is accepted it automatically navigates to the chat.
- **RAG for Search:** We used OpenAI embeddings and ChromaDB to find and search for relevant pre-index movie and actor information to answer questions. We also have the Chatbot create SQL queries over our post and user databases to provide more relevant information, that is combined with more hard SQL queries that split the query into words and check for subsequences in posts / usernames and first and last names. All posts and users have structured URLs so we can ensure that all URLs are correct. We use JSON output for easy parsing, and attach ‘citations’ with each query that we can display on the frontend. We add some prompt engineering to help reduce hallucinations.
- **Post Streaming:** We regularly poll the federated posts, as well as post to it whenever a user makes a new post. We then consume our own new post, so we can ignore our local posts table (except when we want some more information on

then post beyond the schema of federated posts. We also stream information from Bluesky via Kafka where we poll regularly.

- **Hashtags**

Users can upload a comma-separated lists of hashtags when they create a post, that show up with the post itself. When a user joins the platform (or anytime in profile), they can search from a list of all hashtags to choose from, as well as a list of the top-10 on the platform. We also parse out hashtags from federated posts / Bluesky posts. This is used in the recommendation system to help recommend more apt posts.

- **Recommendations / Ranking**

Every hour (triggered by regular API requests and a timer), a Spark job to run adsorption is sent to Livy to run, and all results are written to an RDS table. The home feed consumes this table to show for each user, filtered by the user. All likes/comments are indexed to the ID, so they maintain even after reranking. We add a recency bias, and also bias larger posts with more information and posts with pictures. We also add a small bias for posts on our own page. This all happens before normalization. We run adsorption for a fixed number of iterations or until convergence below a threshold is reached. We add edges for tags, likes, and comments, as well as friends. When posts (i.e. mainly Bluesky posts) have no links/edges to anyone else, we assign base links to everyone, so they are included in the final ranking and not isolated

- **Friend Recommendations**

Using an adaptation of friends through a very large SQL / join statement, we create friend recommendations whenever a user visits the friends page (updates constantly). We weight a candidate more based on the number of friends that are also friends with them. We initially also had a tiebreaker based on how similar the adsorption results are, but removed this since we felt it was unnecessary and slowed down the script.

- **Hosting**

We host everything on AWS. We chose to host every service separately (like backend, chroma, etc...) to have better isolation and easier management of

resources. We run items in docker with volumes for memory, and we use systemd to run in the background, auto restart on failure, and run on login. Services communicate to each other through the exposed APIs and VPC (although we couldn't get this configured for everything). Security policy and inbound rules (with some CORS) is used to limit requests.

- **Toasts and Push Notifications**

The entire web app is wrapped in a websocket controller (different namespace from chat to ensure no conflicts). This wrapper also contains a context for toasts for error/success/info messages. The server can send toasts (push notifications / SSE) through this wrapper websocket, and the frontend can also send toasts itself by invoking its context.

Changes & Lessons

- We initially just stored everything in DynamoDB since we felt it was the easiest to use and the easiest to change we had more fields to add / wanted to change the schema. However, we realized that it was hard to run analysis on this, or to join data from multiple collections, and was just harder to work with as our service grew. As a result, we should to RDS, which was more structured and rigid, but ultimately made a lot of other things easier to do. We learned how effective these structured databases can be, even if they seem harder to work with initially.
- We also initially stored multiple copies of information everywhere. For example, us profile picture was stored in both the user table and the post table. However, this meant when the picture was changed, we needed to make multiple changes, and our code became more and more messy. We eventually changed to making as many tables as necessary and instead joining data together. We learned that even though this was more work when we needed to query or get information, it was much, much easier to work with when needing to ensure one source of ground truth data and when needing to update data.
- We did not store invites and many things related to the chat initially. We would just broadcast it over the websocket. However, this led to a lot of issues like duplicate

invites or losing information that we needed to join between different events. As a result, we switched to storing things in the database / server, and we learned the importance of persistence of everything, even if it seems redundant at the start.

- Our chatbot initially was not as adaptive. We gave it strict restrictions on what it can query from the database, which made most requests to it not provide any useful results. Later on, we allowed it to query its own search results and generate the query itself, so it can be more dynamic with filters and the like. We also had to later add heads alongside the schema since we realized the schema was a bit vague. We also had it initially be recursive / edit itself when it realized its query was wrong in a Chain-of-thought / agenetic behavior, but it just self-looped often (likely because its own context was messing it up) and we realized it was just better to throw an error and ask the user to try again (we think a bigger model like o1 or o4-mini would've been more effective, albeit more expensive, at this).
- We also made all our AWS services use the same security group since it was just easier to do, and we thought they all really needed the same security anyway. However, we realized this led to a lot of issues, especially with cross-service communication, and messed up our services a lot. We ended up spending a good amount of time with custom security groups which made it a lot easier to work with afterward, and we had a lot less errors with communication between services.

(3) Extra Credit Features Implemented

We included a lot of small extra credit, as well as major extra credit features, many of which were listed in the spec, and many which we came up with ourselves.

We're listing everything that we think may be defined as extra credit, but we're not sure if they're all included.

Major

- Web sockets for the chat: chat namespace websocket for real-time chat updates. The message sent to the server is then sent to all clients in the chat in almost

real-time without long polling. Updates like chat requests, invites, declined invites, leaving chat, etc... are also sent through the websocket.

- Web sockets for main application: presence namespace websocket maintains a list of active users (with counting for duplicate sessions) for push notifications and active friend management. A new session updates immediately, and a force closes session (window close) updates after at most 45 seconds when a periodic heartbeat fails.
- Users can create unique links to posts to share (automatically copied to clipboard via the share button).
- Users can bookmark posts, and all bookmarks show up in your profile page.
- Infinite scrolling: reaching 3/4 to the bottom of the page, loads the next 10 posts (with debouncing), that looks almost seamless.
- LLM always returns valid URLs to posts and users.
- LinkedIn-style / two-way friend requests.
- Toasts: all notifications come through a custom toast notification system that is in the right corner and comes in varieties (success, error, warning, info). You can attach custom components for actions like page visits, etc.
- Server-side notifications (SSE) / push notifications. Obviously we can't do proper push notifications since it's a web app, but no matter what page you're on, you can get a notification (across all open windows). The server can also send notifications to the frontend (like when friend request).
- Auto reconnection. When Wi-Fi drops, the server displays a bad connection error, and retries every few seconds until it reconnects.
- LOTS of (nice) error handling for almost everything. We haven't had the server ever crash, or anything fail silently without an error message (toast) that is displayed nicely. Users are notified of every error, and some failed actions have 'retry' messages when it fails.
- We also spent a lot of time on making the UI look really nice! All custom styles and UI that is mostly consistent across pages and displays information as nicely as possible (e.g. modals, custom reusable components, shadows). We tried to make all information as digestible and pleasant to the eye as possible!
- The LLM generates and runs SQL over the user/posts database to get relevant results.

-
- (Almost) Everything updates in real-time without needed to refresh the page (or automated refreshing). Adding a comment/liking a post, changing your username, linking an actor, sending a friend request, a user going offline, creating a post, etc... all show their changes in real-time using react hooks and useEffect. The one thing that doesn't update in real-time is when you upload a profile picture it doesn't show the updated picture in the profile page (because of some consequences of our design choice; but it does show a preview)
 - Preemptive post showing. Shows posts to friends when first made, before the reranking algorithm runs (since it won't run for at most another hour). Still shows in the feed and in your profile.

Moderate

- Optimistic messaging for chat. Sending a message (with slow network) shows up on your side (but slightly gray). Once sent to the server with confirmation / acknowledged by the server (through the websocket), it is then confirmed (no longer grayed out).
- Rejection Logic: Chat invites are persisted, and sending extra invites will compare with all current invites as well as current group members. If the max potential group equals a current group (in members), the invites are rejected (as in the spec, with a bit more complexity).
- Top 5 embedding matches are always stored, so you can switch to one at any time (as well as unlink at any time)
- Profile page also includes all your posts (and you can still interact with them), ordered by time
- Can send friend requests / accept them from the search page.
- You can unfriend someone :(
- Can create group chats at once: creates an empty chat with just you and sends invites to everyone else.
- All pages and routes have authentication wrappers to (a) prevent unauthorized access (b) send the users to the login page if not logged in, and (c) get the user data easily in requests
- Auto homepage redirection when already logged in (using cookies)

-
- Loading text (like when opening a page) a disabled button to prevent issues when already pressed (until ready again).
 - Notifications when someone unfriends/friends you or sends a request
 - Notifications when someone sends you a message
 - Friend Recommendations recompute on every page load, not just every hour
 - Auto restart services in AWS on crash, and run in background and on login.

Minor

- Special system messages for when a user joins and leaves a chat that persists.
- Chats with just yourself are considered notes to self and can still be used.
- Duplicated Invite handling in chat
- Persistent Invite handling
- Can send multiple invites to the same user for DMs and chats. Chat invites include all other members in the chat in the description.
- Commenting includes profile picture, name (with link to profile)
- background tasks and auto reloads in AWS if the server crashes
- Nice little animation when you like a post, lol.
- Can visit profile from post
- Can unlink an actor from your profile.
- Little error handling when your profile image isn't valid -- even if the URL exists but isn't valid (i.e. a separate fetch to test) -- (if in profile and it's your image, it says please upload image), otherwise it shows the first letter of your name in a nice font.
- Can chat and unfriend/friend/accept/decline/remove friend requests from the profile page
- Natural timestamps: Shows how long you've been friends with someone (since today, 3 hours ago, etc...). Same type of natural language timestamp for posts.
- Chatbot lets you know when it doesn't know something rather than give overly confident answers.

Never Realized (but some code is there)

- Stories, add the route to post a story, but didn't have time to implement the UI

-
- Multi-image posts, all routes support multiple images (but didn't have time to implement the UI again)
 - Forgot Password, have all the UI and routes, but the emails were never sending for some reason.
 - View trending posts and be able to visit all posts related to a tag (wrote some UI, didn't finish backend in time)
 - Can view profile of users not in database (federated posts), kind of works but very buggy.

(4) Source Files

- `backend/`: Contains all Node.js/Java backend server code and RDS communication logic.
 - `server.js / MainApplication.java`: Main entry point and routing.
 - `routes/`: API endpoint definitions.
 - `models/`: Facial Detection models.
 - `services/`: Presence service for online/offline detection with heartbeat.
- `frontend/`: Contains all React frontend code.
 - `src/components/`: Reusable UI components and page components
 - `src/context/`: Toast Context for push notifications.
 - `src/App.js`: Main application component including topbar and routing.
- `Indexing/` and `movies/`: Contains all code to join databases, index, and upload movie/actor data with custom batch embedding logic.
- `src/main/`: Contains the Livy and Spark application code
 - `src/main/java/` or `src/main/scala/`: Spark job source.
 - `pom.xml` or `build.sbt`: Build configuration.

(5) Application Images

Login Page

InstaLite

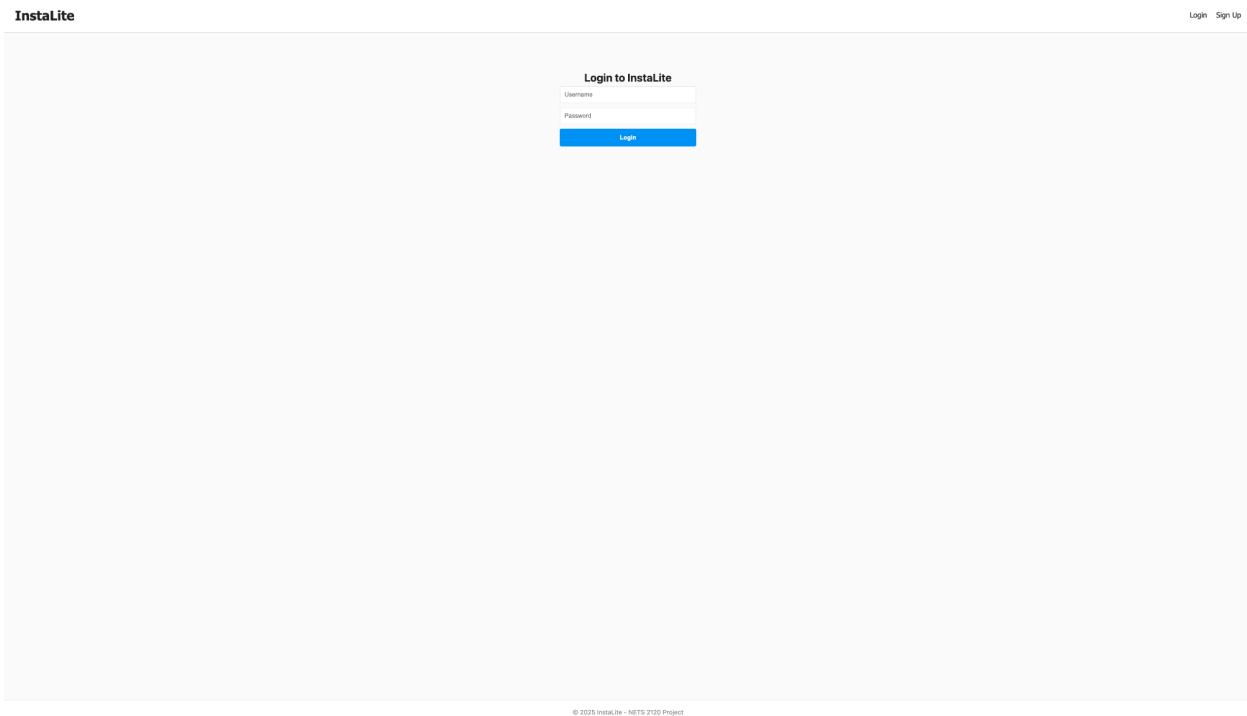
Login to InstaLite

Username

Password

Log In Sign Up

© 2025 instalite - NETS 2120 Project



Sign Up Page

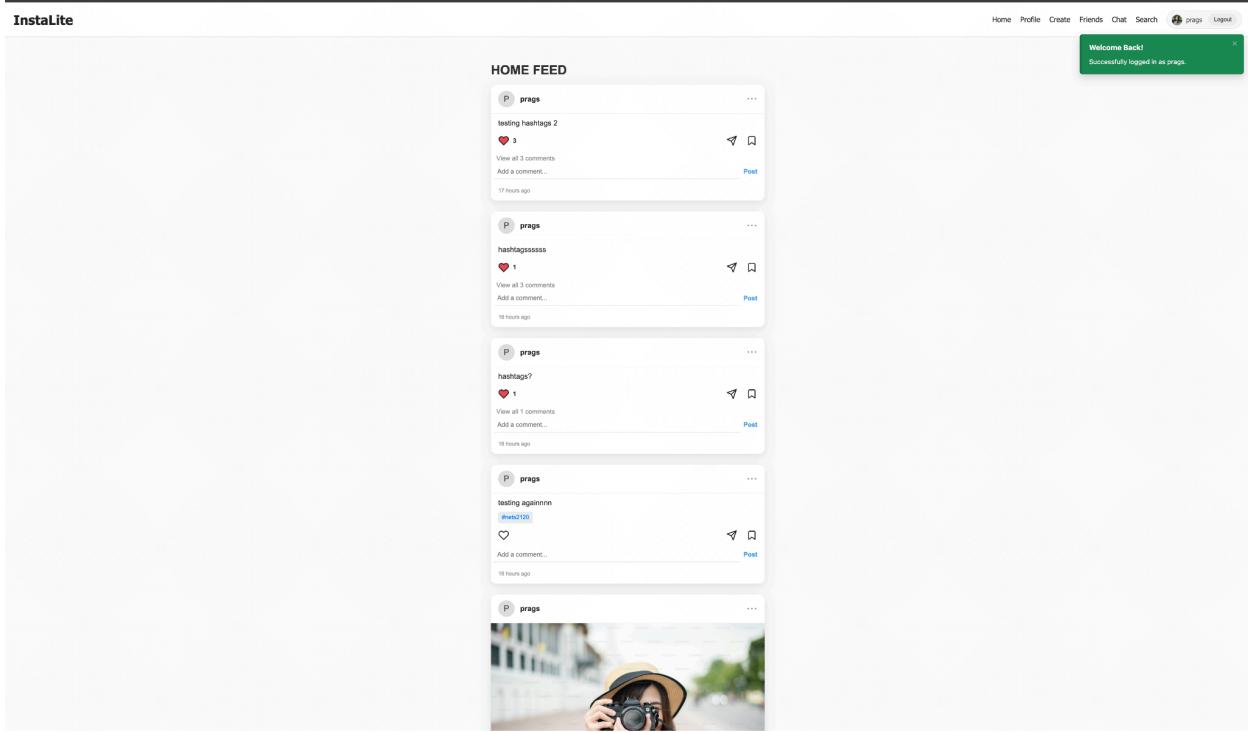
The screenshot shows the 'Create your InstaLite Account' sign-up form. The form fields are as follows:

- Username* (text input field)
- Email* (text input field)
- Password* (Min. 8 characters) (text input field)
- Confirm Password* (text input field)
- First Name* (text input field)
- Last Name* (text input field)
- Affiliation (Optional) (text input field)
- Birthday (Optional)
mm/dd/yyyy (date input field)

A large green 'Sign Up' button is centered at the bottom of the form. Below the button, a small link reads 'Already have an account? [Log In](#)'.

© 2025 instaLite - NETS 2120 Project

Login (Toast) and Home Page



Comments (Folded and Unfolded) + Liking Messages

HOME FEED

P prags

...

testing hashtags 2



2



[View all 4 comments](#)



prags wow that's so cool!



alice sup



pulks hey



prags wassup!

Add a comment...

[Post](#)

17 hours ago

P prags

...

hashtagssssss



1



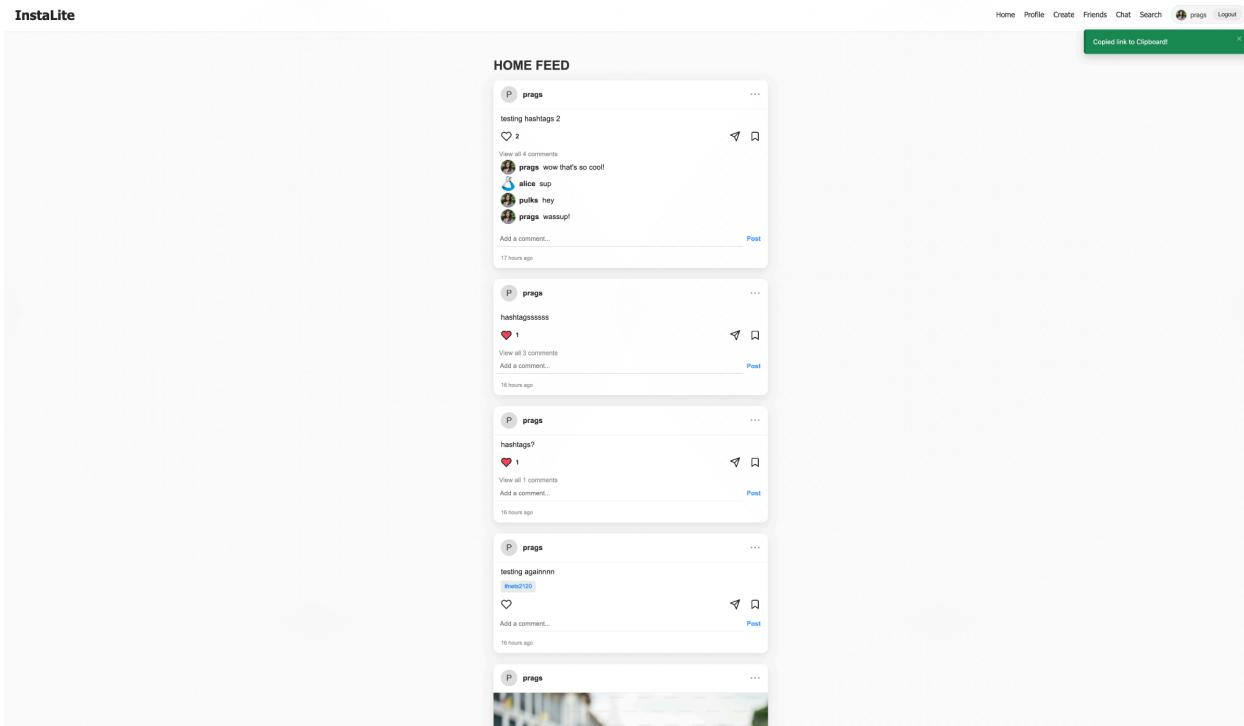
[View all 3 comments](#)

Add a comment...

[Post](#)

16 hours ago

Pressing “Share” Button



Shared / Single Post View

InstaLite

Home Profile Create Friends Chat Search  prags Logout

P prags ...



testing 234

Image

1

Add a comment...

Post

10 hours ago

[e- Back to Home](#)

© 2025 InstaLite - NETS 2120 Project

Post with Tags

A alice



Latest Post!

monk monke monkey

1 Add a comment... [Post](#)

6 hours ago

Bluesky + Federated Posts + Infinite Scrolling

The image displays three federated posts from the Bluesky platform, illustrating the concept of infinite scrolling. Each post is contained within a rounded rectangular card.

- Post 1:** Author: dwg (represented by a 'D' icon). Content: "hog". Interactions: 1 heart icon. Action buttons: a paper airplane icon (Share), a bookmark icon, and a blue "Post" button.
- Post 2:** Author: meg (represented by an 'M' icon). Content: "hello". Interactions: 1 heart icon. Action buttons: a paper airplane icon, a bookmark icon, and a blue "Post" button.
- Post 3:** Author: Rotten Tomatoes (represented by a red circular logo with 'RT'). Content: "Taron Egerton, Jurnee Smollett, Greg Kinnear, and John Leguizamo in first look images of #Smoke. The new Apple TV+ series premieres June 27." Interactions: 6 hearts (indicated by the number '6' next to the heart icon). Action buttons: a paper airplane icon, a bookmark icon, and a blue "Post" button.

The cards are arranged vertically, demonstrating the continuous nature of the feed. The interface includes a thin horizontal bar at the top and bottom of the main content area.

Profile Page

InstaLite

Home Profile Create Friends Chat Search  prags 

**prags**
Pragya Singh
Email: pragyasingh7@gmail.com
Affiliation: Penn
Birthday: 11/22/2005

Actor Identity Link
 Mary Pickford
Currently Linked to this actor.

[Manage Link](#)

Your Hashtag Interests [Manage Interests](#)

Update Profile Photo & Find Similar Actors
[Choose a Photo](#) [Upload & Match Actors](#)

Posts by prags

 prags
Hi I want to make a post

   
[Add a comment...](#) [Post](#)
4 hours ago

 prags


Profile Page (Different Account)

InstaLite

Home Profile Create Friends Chat Search puls Logout

**prags**
Pragya Singh
Email: pragya singh7@gmail.com
Affiliation: Penn
Birthday: 11/22/2005

 Actor Linked: Mary Pickford

Friendship with prags

-- Friends Since 3 days ago  

Posts by prags

 prags
Hi i want to make a post 


Manage / Unlink Actors Modal



prags

Pragya Singh
Email: pragyasingh7@gmail.com
Affiliation: Penn
Birthday: 11/22/2005

Actor Identity Link

Manage Actor Link

Currently Linked Actor



Mary Pickford
This actor is linked to your profile.

Unlink

Choose an Actor to Link

- **Marc McDermott**
Link Profile
- **Mary Pickford**
Link Profile
- **Theda Bara**
Link Profile
- **Geraldine Farrar**
Link Profile
- **Dell Henderson**
Link Profile

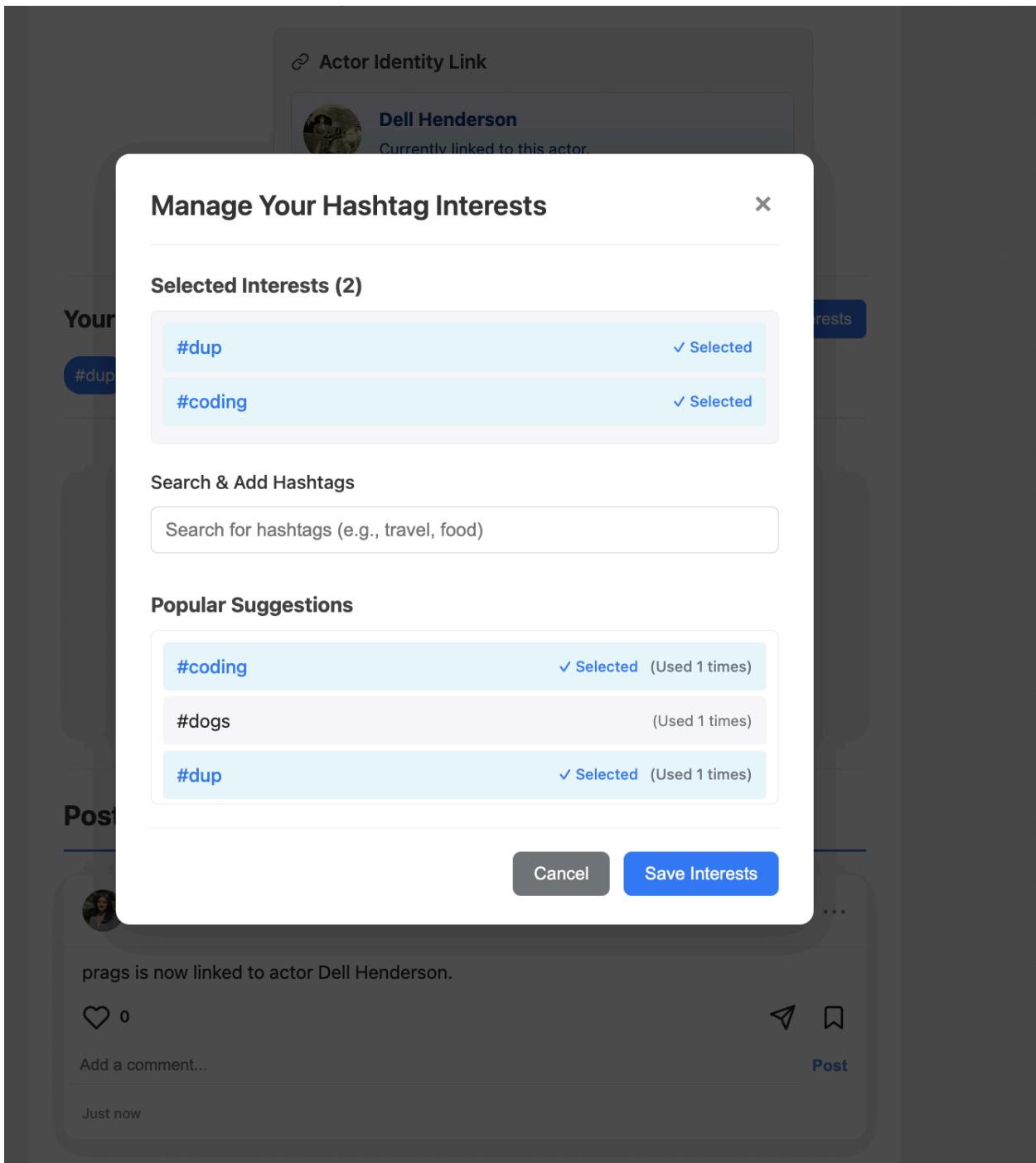
Close

Changing Actor + Auto Post

The screenshot shows a user profile on the InstaLite platform. At the top, there's a navigation bar with links for Home, Profile, Create, Friends, Chat, Search, and Logout. A green success message box says "Successfully linked to Dell Henderson". The main profile area for "prags" (Pragya Singh) includes her photo, name, email (pragyaingh7@gmail.com), affiliation (Penn), and birthday (1/22/2005). Below this is an "Actor Identity Link" section showing a link to "Dell Henderson" with the note "Currently linked to this actor." A "Manage Link" button is present. There's also a "Your Hashtag Interests" section with a "Manage Interests" button. A large central box is titled "Update Profile Photo & Find Similar Actors" with a "Choose a Photo" input field and a "Upload & Match Actors" button. Below this is a "Posts by prags" section displaying two recent posts:

- The first post shows a link to Dell Henderson with the message "prags is now linked to actor Dell Henderson." It has a "Post" button.
- The second post shows a message "Hi i want to make a post" and a "Post" button.

Changing Tags Modal



Uploading Profile Picture + Choose Initial Matches

The screenshot shows the InstaLite application interface. At the top, there's a navigation bar with links for Home, Profile, Create, Friends, Chat, Search, and Logout. A green notification bar on the right says "Profile image uploaded and processed." On the left, there's a sidebar labeled "InstaLite". The main content area has a header "Your Hashtag Interests" with a "Manage Interests" button. Below it is a section titled "Update Profile Photo & Find Similar Actors" showing a preview of the uploaded profile photo (Profile Image IMG 2093.jpeg) and a "Preview:" button. A blue button labeled "Upload & Match Actors" is present. A message below the preview says "Your new profile photo is active." Underneath, there's a section titled "Top Actor Matches" listing five actors with their similarity scores and "Link to Profile" buttons:

Actor Name	Similarity	Action
Marc McDermott	Similarity: 79.9%	Link to Profile
Mary Pickford	Similarity: 77.4%	Link to Profile
Theda Bara	Similarity: 76.0%	Link to Profile
Geraldine Farrar	Similarity: 75.0%	Link to Profile
Dell Henderson	Similarity: 74.9%	Link to Profile

At the bottom, there's a section titled "Posts by prags".

No linked actor / unlink result



prags

Pragya Singh

Email: pragyasingh7@gmail.com

Affiliation: Penn

Birthday: 11/22/2005

 **Actor Identity Link**

You have potential actor matches. Link your profile to an actor.

 **Link to Actor**

Your Hashtag Interests

#dup

Manage Interests

Profile with Pending Request



alice

Alice Smith
Email: alice@example.com
Affiliation: Penn
Birthday: 3/15/1995

Friendship with alice

alice sent you a friend request!
Sent Yesterday

Accept Reject

You need to be friends with alice to chat with them.

Posts by alice



Friendship with pulks

Friend Request Pending

Cancel Request

You need to be friends with pulks to chat with them.

Change Username / Password

 **Pragya Singh**
Email: pragyasingh7@gmail.com
Affiliation: Penn
Birthday: 11/22/2005

Actor Identity Link
You have potential actor matches. Link your profile to an actor.

Account Settings ×

Change Email

New Email Address

Update Email

Change Password

Current Password

New Password

Confirm New Password

Update Password

Your Hashtags #dup Manage Interests

Posts by prags

prags is now  0 comments ...

Heart 0 Post

Add a comment... Post

Just now

Create Post Page

InstaLite

Home Profile Create Friends Chat Search  Logout

Create a New Post

What's on your mind?

Image URL:

Hashtags (comma-separated)

© 2020 InstaLite - NETS 2120 Project

Friends Page + Friend Recommendations + Pending Friend Request + Offline / Online Friends + Unfriend + Invalid Profile Picture - (Charlie; shows first letter of their name as icon)

The screenshot shows the 'Manage Friends' section of the InstaLite application. At the top, there's a navigation bar with links for Home, Profile, Create, Friends, Chat, Search, a user icon labeled 'pauls', and Logout.

Pending Friend Requests (1)

- bob (Bob Jones) with 'Accept' and 'Reject' buttons

People You May Know (1)

- alice (Alice Smith) with an 'Add Friend' button

Your Friends (2)

Online (1)

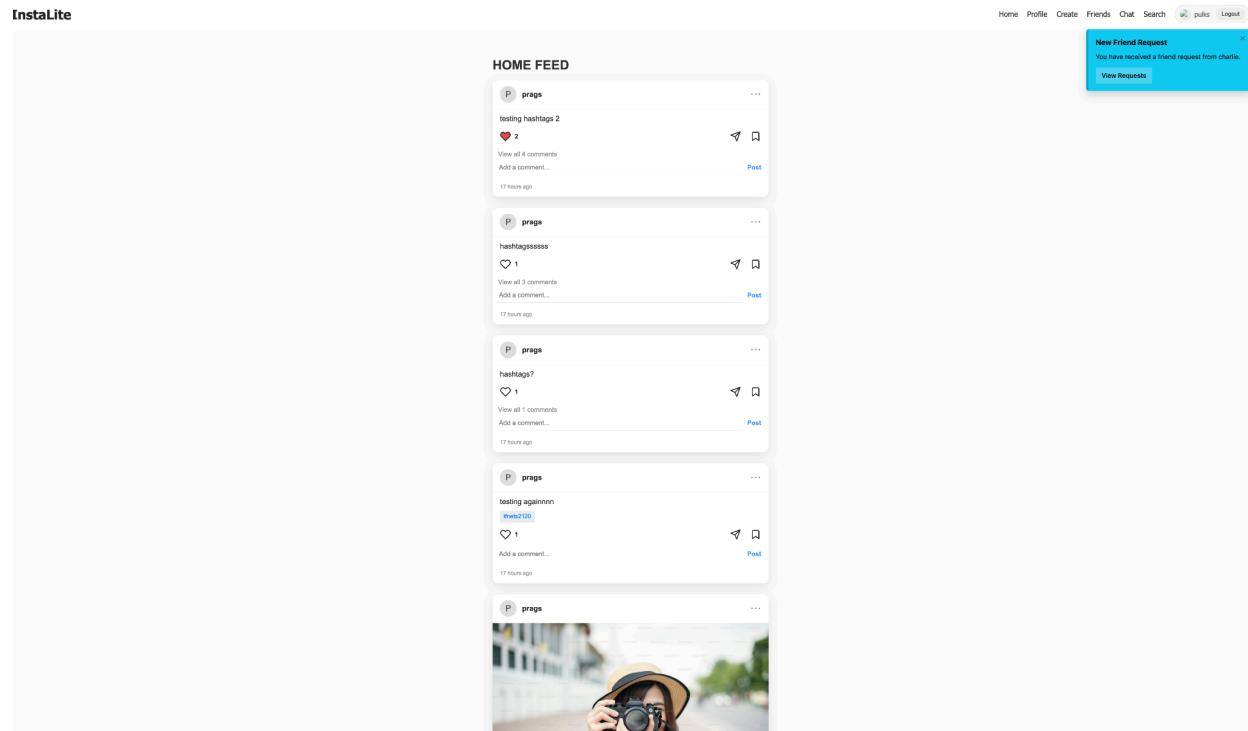
- praga (Pragya Singh) with an 'Unfriend' button

Offline (1)

- charlie2 (Charlie bobbins) with an 'Unfriend' button

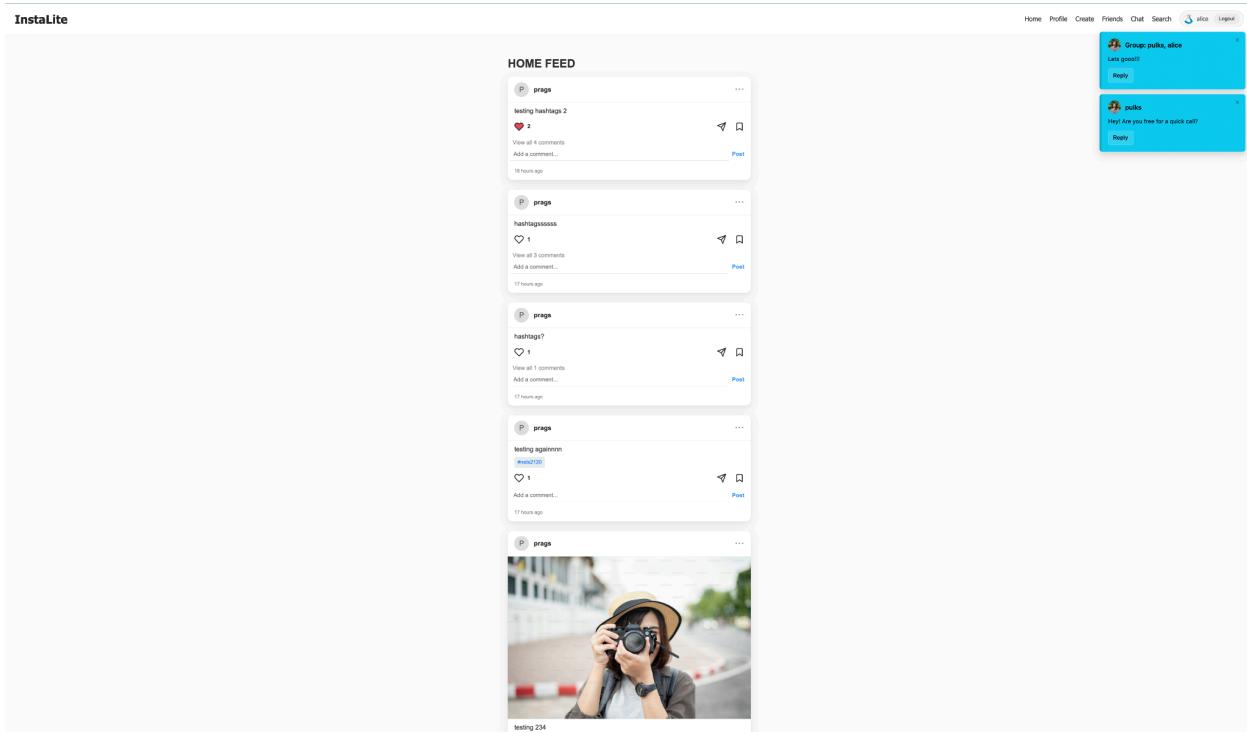
At the bottom of the page, a small copyright notice reads: © 2020 insta.lite - NETS 2120 Project.

Push Notifications (friend request + interaction)



```
User 3 (charlie) sent friend request to 5
[Toast Service] Sending toast to user 5 (sockets: 1) {
  duration: 5000,
  viewText: 'View Requests',
  title: 'New Friend Request',
  content: 'You have received a friend request from charlie.',
  type: 'info',
  viewLink: '/friends'
}
```

Push Notifications (Chat Message + Group Chats)



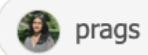
Friend Request Accepted

alice accepted your friend request!

[View Profile](#)



Home Profile Create Friends Chat Search



prags

[Logout](#)

Friend Removed

alice removed you as a friend.



Ranking + Infinite Scroll

```
Fetching page 1 with limit 20, offset 0 for user 5
Top ranked posts: [
  {
    post_id: 280,
    rank_score: 0.21194840595380915,
    source_site: '2y2b',
    post_uuid_within_site: '9bd1777a-a8e7-4589-987e-28e0663b656d'
  },
  {
    post_id: 281,
    rank_score: 0.21194840595380915,
    source_site: '2y2b',
    post_uuid_within_site: 'e7e11a06-4f4c-4b56-9919-6795b6692245'
  },
  {
    post_id: 279,
    rank_score: 0.21194840595380915,
    source_site: '2y2b',
    post_uuid_within_site: 'f35c5961-255b-4fa1-a9b0-09b2eec10cd2'
  },
  {
    post_id: 274,
    rank_score: 0.21194840595380915,
    source_site: '2y2b',
    post_uuid_within_site: 'some-uuid'
  },
]
```

```
Fetching page 2 with limit 20, offset 20 for user 5
Top ranked posts: [
  {
    post_id: 185,
    rank_score: 0.00006623387686056536,
    source_site: 'instalite-wahoo',
    post_uuid_within_site: '1695'
  },
  {
    post_id: null,
    rank_score: 0.00006623387686056536,
    source_site: 'bluesky',
    post_uuid_within_site: 'at://did:plc:zimtt7q4ei2pcqupfr7s4alp/app.bsky.feed.post/3ln3y2gdn5s2z'
  },
]
```

Group Chat

InstaLite

Home Profile Create Friends Chat Search pulka

Chatterbox

Active Chats

- Group: bob, prags
- Group: bob
- Group: bob
- Group: prags

Group: bob, prags

pulka

Leave

prags has left the chat.
10:55 AM
prags has joined the chat.
10:55 AM
prags has left the chat.
10:55 AM
prags has joined the chat.
10:55 AM
prags has left the chat.
10:55 AM
prags has joined the chat.
10:55 AM
bob has joined the chat.
10:55 AM

prags

hello
7:31 AM

bob

prags

Friends (2)

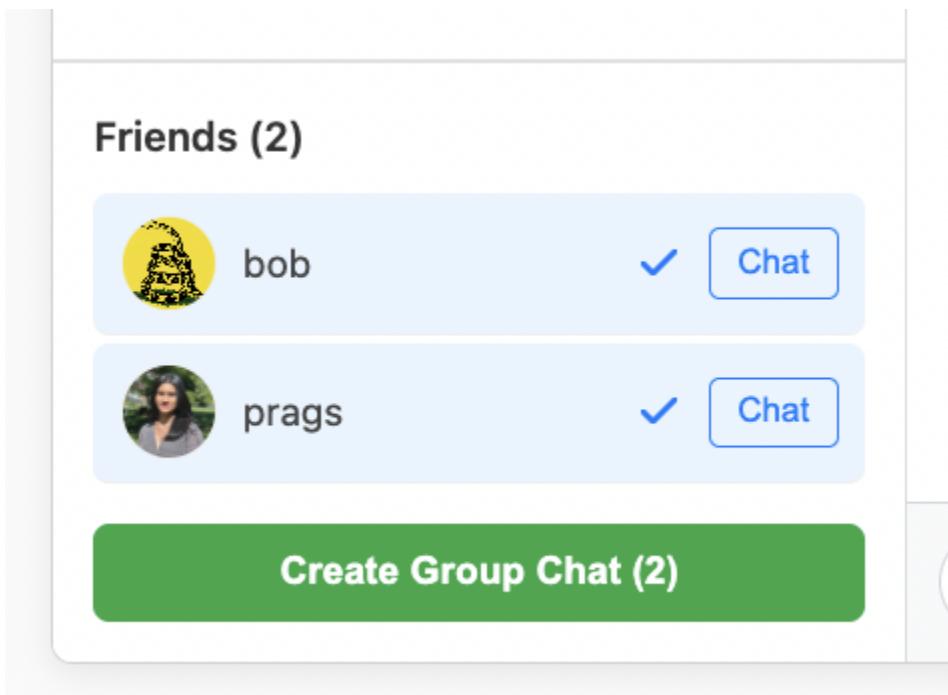
bob

prags

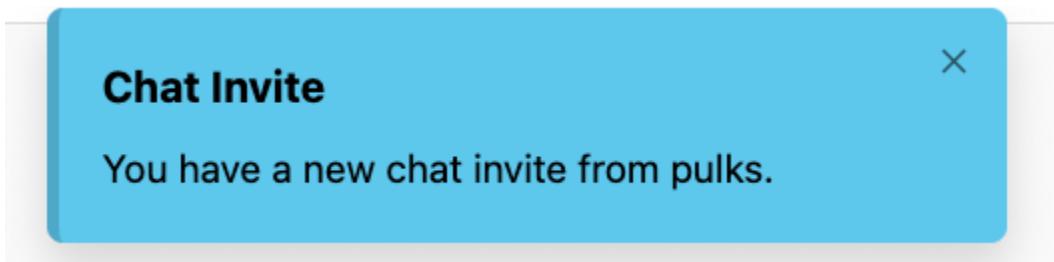
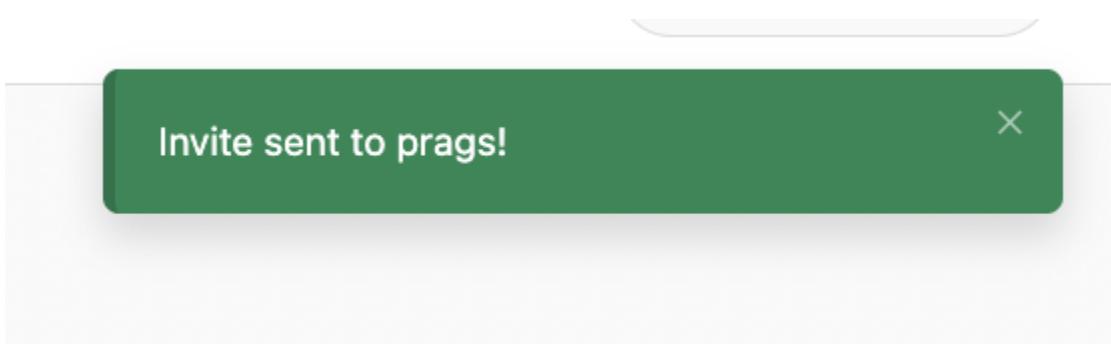
One

One

Group Chat Creation



Invite Notifications



Invite declined.



Invite Declined

prags declined your chat invite.



Invite Center

Pending Invites (2)



prags

Invites you to a new group wit

Accept

Reject

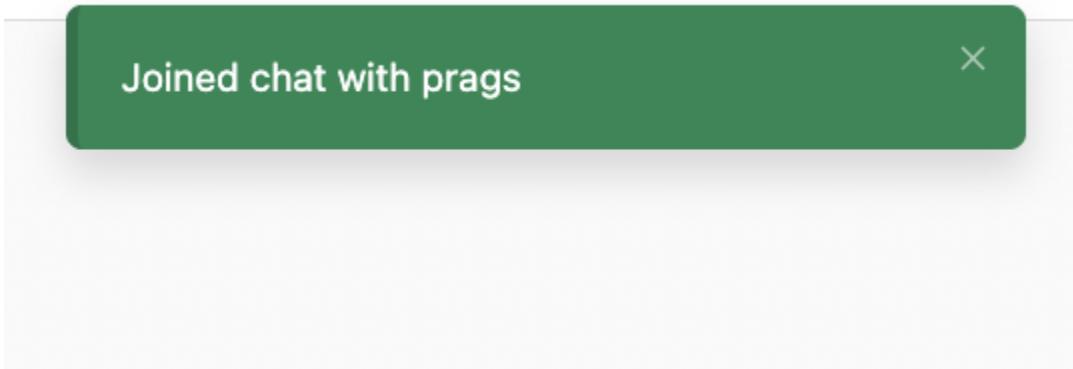


From: prags

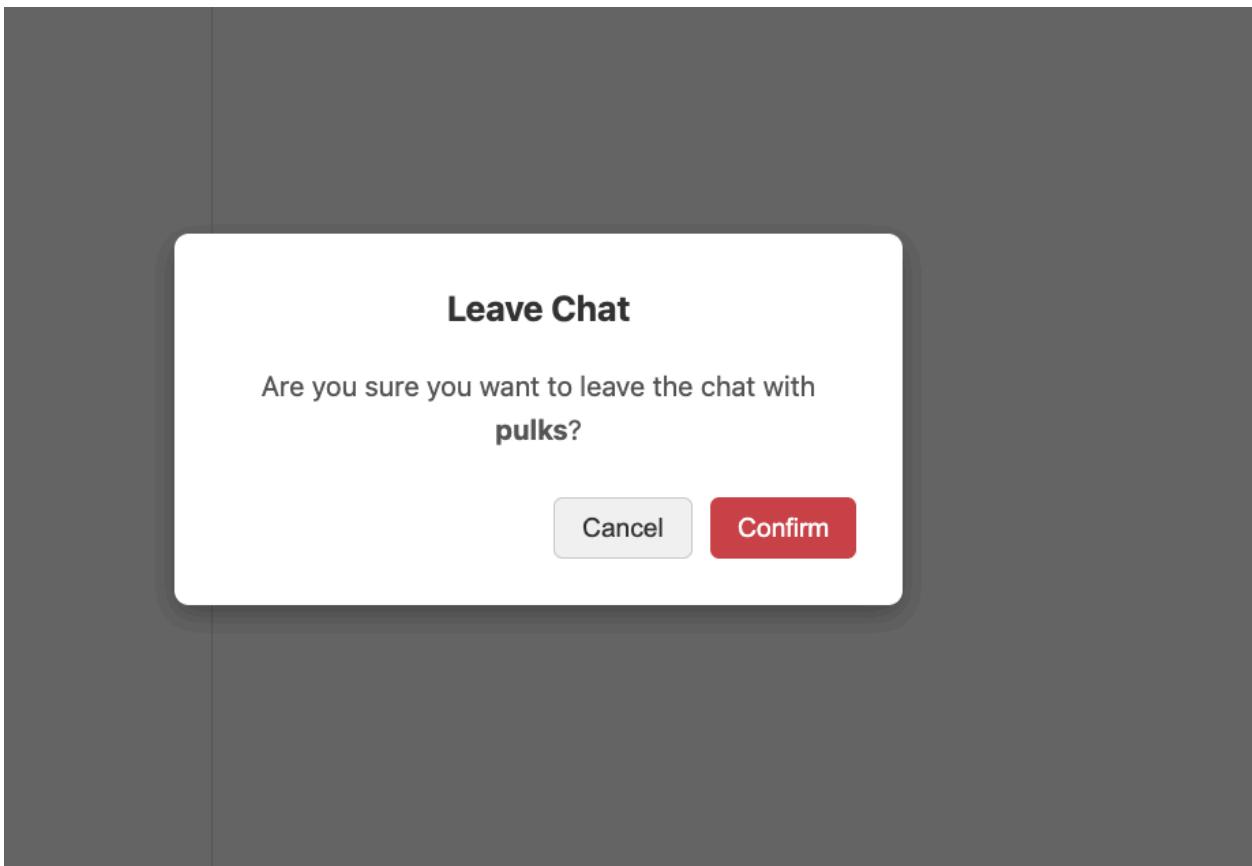
Accept

Reject

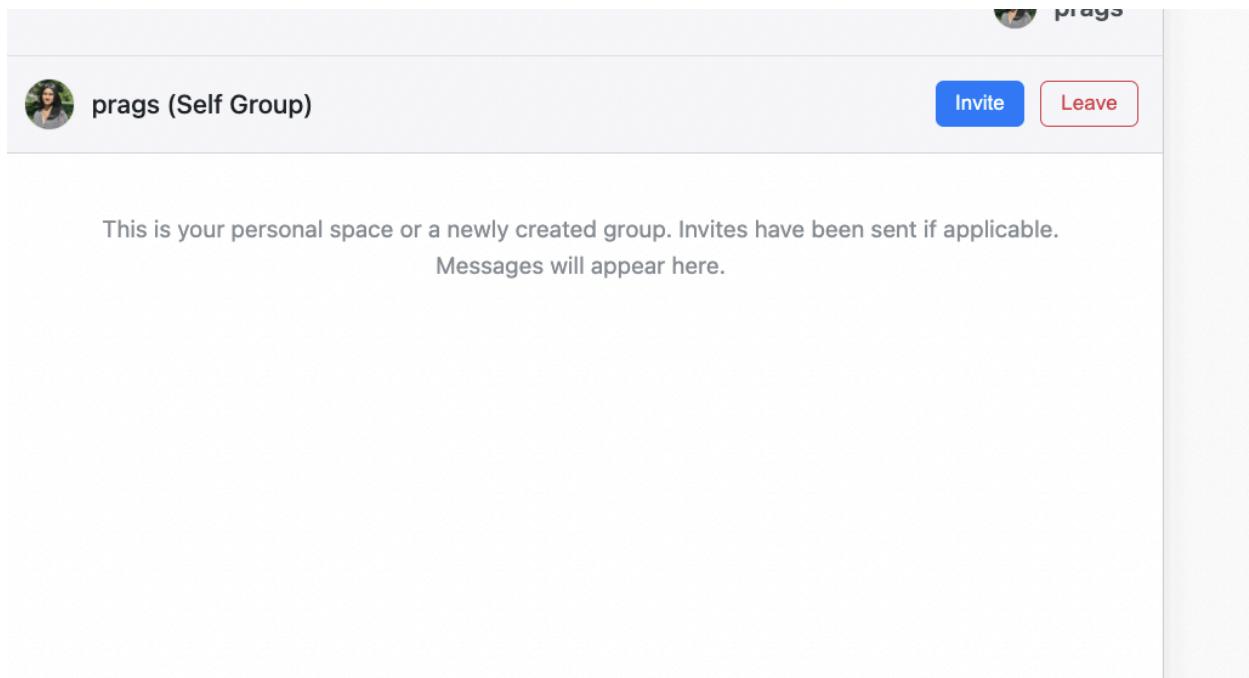
Accepted Invite



Leave Chat



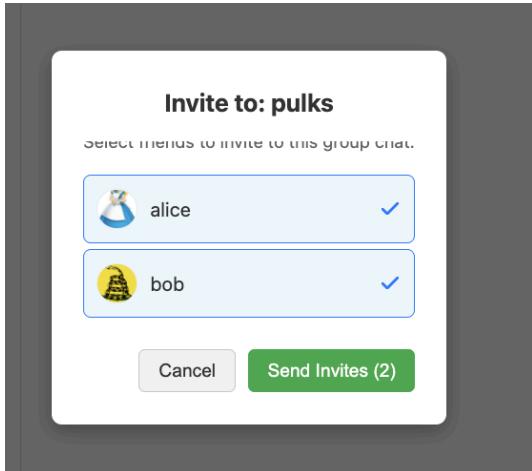
New Group Chat



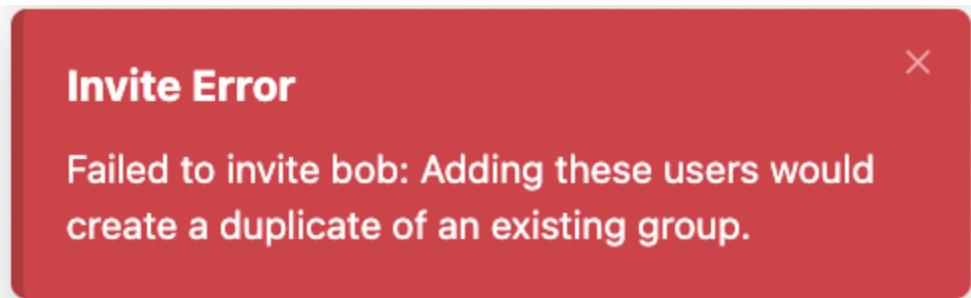
Active Chats

This screenshot shows a list of active chats on the left and a detailed view of a specific group chat on the right. On the left, under "Active Chats", there are four items: "Group: bob, pulks" (highlighted with a blue circle containing the number 3), "Group: pulks" (highlighted with a blue circle containing the number 2), a profile picture of a user named "pulks", and another "Group: pulks". On the right, the detailed view for "Group: bob, pulks" shows the following messages:
- "prags has left the chat." at 10:00 PM
- "prags has joined the chat." at 10:01 PM
- "prags has left the chat." at 10:01 PM
- "prags has joined the chat." at 10:01 PM
- "prags has left the chat." at 10:15 PM
- "prags has joined the chat." at 10:15 PM

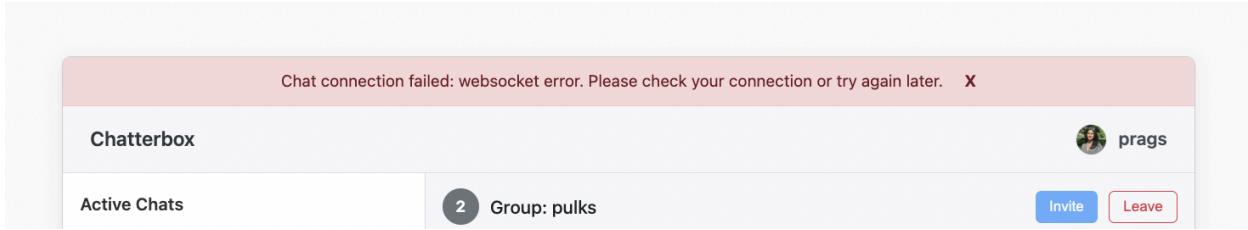
Invite to Existing Chat



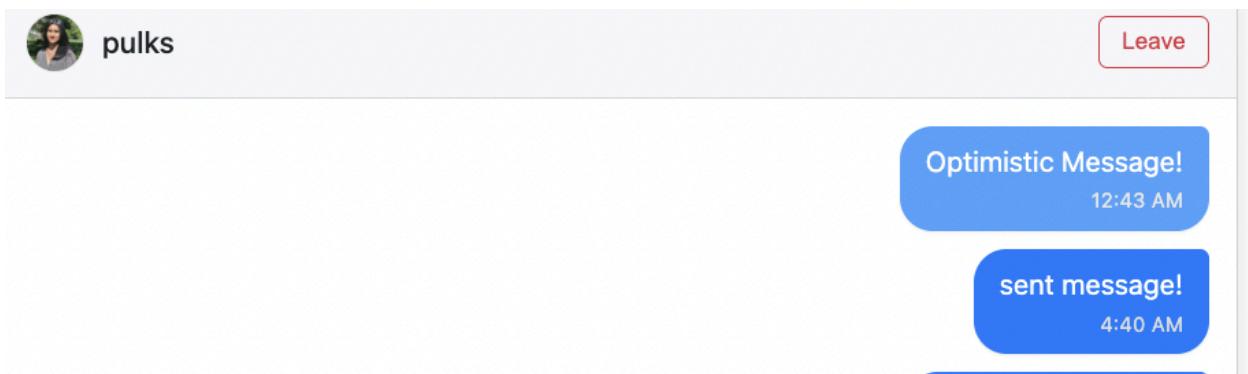
Duplicate Checking



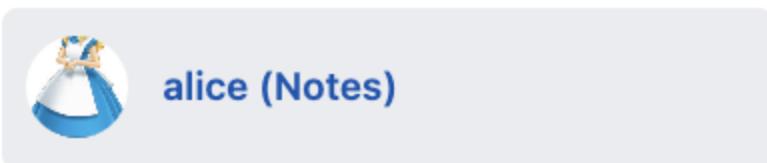
Auto-Reload Connection



Optimistic Messaging



Self Group (Notes)



Search for User (with valid links) + Not sending irrelevant info + SQL queries from LM

Universal Search Assistant

Explore actors, movies, users, and posts from across platforms.

Search

💡 AI Summary

Based on the provided information, there is a platform user named Bob Jones with the username 'bob' and a profile link, but no relevant actor or movie information was found. Additionally, no federated or Bluesky posts matched the query. No other relevant results are available.

👤 Platform Users



Bob Jones
@bob
Drexel

[Add Friend](#)



charlie bobkins
@charlie2

[Add Friend](#)

Grounding / Hallucinations

InstaLite

Home Profile Create Friends Chat Search Log in

Universal Search Assistant

Explore actors, movies, users, and posts from across platforms.

Tom Cruise

AI Summary

Based on the provided information, there were no results found. Tom Cruise in the actor or movie database, nor any platform users matching your query. However, multiple posts on the platform were found related to Tom Cruise's role in "The Final Frontier" starring Tom Cruise, including promotional posters and release details. Additionally, some posts mention other TV series and movies, but none specifically focus on Tom Cruise himself. No relevant federated posts were found. Overall, the most relevant information pertains to Tom Cruise's role in the upcoming film, as highlighted in Blusky posts.

Blusky Posts

didgciamnt7q4el2pcqupf7w4elp [View](#)

Pokerface: Season 2 is now Certified Fresh at 100% on the Tomatometer, with 22 reviews: [www.rottentomatoes.com/tv/poker...](#)

didgciamnt7q4el2pcqupf7w4elp [View](#)

Pokerface: Season 2 is now Certified Fresh at 100% on the Tomatometer, with 22 reviews: [www.rottentomatoes.com/tv/poker...](#)

Searching for question / actors / movies + actor info + AI summary + posts (liberal SQL query so we can see some results)

InstaLite

Home Profile Create Friends Chat Search Log in

Universal Search Assistant

Explore actors, movies, users, and posts from across platforms.

What movies was Tom Cameron in?

AI Summary

Based on the provided information, Tom Cameron was involved in the movie 'The Crisis' (1913). No other relevant movies or actors matching the name 'Tom Cameron' were found in the search results. There are no platform users or posts specifically related to your query about Tom Cameron.

Actors & Movies

Jim Cameron's Wife (1914) [View](#)

Actor: Tom Chatterton
Role: Jim Cameron
Date: 1914-01-01 | Studio: The Nickelodeon | Actor: Tom Chatterton | Role: Jim Cameron | Category: actor | Rating: 1/10 | Runtime: 20 min
Synopsis: Unknown | 1,000

The Crisis (1913) [View](#)

Actor: Tom Cameron
Role: actor
Date: 1913-01-01 | Studio: The Crisis | Actor: Tom Cameron | Role: Tom Cameron | Category: actor | Rating: 1/10 | Runtime: 250 min
Synopsis: Unknown | 1,000

Federated Posts

rodrigo 1/3/2023, 7:54:27 PM via team-netb210
what's up!!!!!!!

rodrigo 1/3/2023, 7:54:27 PM via team-netb210
what up

meq 1/3/2023, 4:16:37 PM via a-fuchs
what

Anonymous 1/3/2023, 4:16:39 PM via lastfmhttp://www.lastfmhttp://www.lastfm
Pete Alvin Luigi Mancuso was asked about the game on Guy High, and he said "But like seriously, how is there an entire game on this website? That should get some extra credit if this is a class project..."

EC2 Instances (Chroma, Backend, Kafka)

The screenshot shows the AWS EC2 Instances page. The left sidebar contains navigation links for EC2 services like Dashboard, Global View, Events, Instances, Images, Elastic Block Store, Network & Security, Load Balancing, Auto Scaling, and Settings. The main content area displays a table of 5 instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
finalprojecttest	i-00142eb53a01d835a	Running	t2.micro	Initializing	View alarms +	us-east-1d	ec2-52-91-114-121.co...
TunnelY	i-0d01381236560a9bd	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1d	ec2-54-209-107-196.co...
TunnelX	i-09b7940a14068bd4b	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1d	ec2-54-163-153-231.co...
chromadb	i-06b7afc7e7bal195b	Running	t3.medium	3/3 checks passed	View alarms +	us-east-1c	ec2-3-236-136-231.co...
backend	i-0ef7d0bbd203699eb	Running	m5.large	3/3 checks passed	View alarms +	us-east-1a	ec2-54-234-164-225.co...

At the bottom, there is a section titled "Select an instance" with a dropdown menu.

Chroma + Backend Hosted on AWS

```
ubuntu@ip-172-31-24-166:~/FinalProject/backend$ npm start
Dedicated Hosts
Capacity Reservations
Auto-assigned IP address
  □ 54.234.164.225 [Public IP]
m5.large
VPC ID
  □ vpc-0b65950e5c6adb11 □
AWS Compute
  □ Opt-in to AWS Lambda
  □ Learn more ...
Auto Scaling Groups
  □
  - Instance ARN
    □ arn:aws:ec2:us-east-1:65742538455:instance/i-0ef7d0bb2d
      03699eb
CPU Instructions
  - Managed
  - false
(node:2119) [MODULE_TYPELESS_PACKAGE_JSON] Warning: Module type of file:///home/ubuntu/FinalProject/backend/server.js is not specified and it doesn't parse as CommonJS.
Reparising as ES module because module syntax was detected. This incurs a performance overhead.
To eliminate this warning, add "type": "module" to /home/ubuntu/FinalProject/backend/package.json.
[Use 'node --trace-warnings ...' to show where the warning was created]
Creating MySQL connection pool...
(node:2119) [MODULE_TYPELESS_PACKAGE_JSON] Warning: Module type of file:///home/ubuntu/FinalProject/backend/server.js is not specified and it doesn't parse as CommonJS.
Reparising as ES module because module syntax was detected. This incurs a performance overhead.
To eliminate this warning, add "type": "module" to /home/ubuntu/FinalProject/backend/package.json.
[Use 'node --trace-warnings ...' to show where the warning was created]
2025-05-09 05:07:04.82310: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 AVX512F FMA IMDSV2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Attempting to load local actor names from: /home/ubuntu/FinalProject/backend/routes/names.csv
Parsing local CSV. Total lines: 6232
Snapshots
Finished parsing local CSV. Successfully loaded info for 6231 actors into map.
Operator
Encountered 1 lines with parsing issues in local CSV.
Initializing Facebox models...
Initialization of ImageClient with path: http://3.237.76.162:8000
Configuring static file serving for actor imaged from: /home/ubuntu/FinalProject/Indexing/images
  Directory found, Serving under URL path '/actor-images'.
Registering API routes.
Chat Socket Service initialized and attached to /chat namespace.
Global Presence Service initialized on /presence namespace.
Using TensorFlow backend: tensorflow
Loading models from: /home/ubuntu/FinalProject/backend/models
Server running on http://localhost:8080
Accepting connections from frontend at: http://localhost:3000
Database Host: instalte-attempt-2.chawsielycuon.us-east-1.rds.amazonaws.com
(node:2119) NOTE: The AWS SDK for JavaScript (v2) is in maintenance mode.
  - SDK releases are limited to address critical bug fixes and security issues only.
Target Groups
Please migrate your code to use AWS SDK for JavaScript (v3).
  - For more information, check the blog post at https://a.co/cUpNyil
  - >>> Successfully connected to RDS MySQL database via pool.
FaceAPI models initialized successfully.
  - Auto Scaling
Async backend initialization tasks complete.
  - Auto Scaling Groups
  - Platform details
    □ Linux/UNIX
  - Termination protection
    □ Disabled
  - AMI location
    □ amazon/ubuntu-2025
  - Instance details
    □ ami-0f9de6e2d2f067fca
      Monitoring
        disabled
      Name
        □ ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20250305
        Allowed image
        -
      Launch time
        □ Thu May 08 2025 23:44:52 GMT-0400 (Eastern Daylight Time) (about 1 hour)
  - Details
  - Status and alarms
  - Monitoring
  - Security
  - Networking
  - Storage
  - Tags
```

S3 Profile Pics / Images

The screenshot shows the AWS S3 console interface. The left sidebar includes sections for General purpose buckets, Storage Lens, Feature spotlight, and AWS Marketplace for S3. The main area displays a list of objects in a bucket named "profile-images_". The list includes various file types like jpg and png, with details such as name, type, last modified, size, and storage class. A search bar at the top allows filtering by prefix.

Name	Type	Last modified	Size	Storage class
043e0677-b3ca-4c3d-a8ea-5b379c1050d2.jpg	jpg	April 23, 2025, 11:40:19 (UTC-04:00)	356.2 KB	Standard
04cc34c-bd6f-4af8-f7f1-8740955303e2.jpeg	jpeg	May 6, 2025, 19:43:23 (UTC-04:00)	7.5 KB	Standard
062d13f4-e1b1-482b-8e54-d556b9a1fe93.png	png	April 23, 2025, 11:40:24 (UTC-04:00)	2.9 MB	Standard
09ce6108-8cf6-4528-81f8-198a5c929ca3.jpeg	jpeg	May 7, 2025, 23:20:35 (UTC-04:00)	7.5 KB	Standard
0cd9707d-ca5a-4841-af6e-1ebab8f99de.png	png	April 23, 2025, 11:59:57 (UTC-04:00)	2.9 MB	Standard
171f1e8f-20f0-49ad-8e4d-c9228ea3091.jpeg	jpeg	May 8, 2025, 01:03:47 (UTC-04:00)	2.8 MB	Standard
1788e0ad-6717-40de-96eb-3ad2d3315ca6.jpeg	jpeg	May 5, 2025, 23:37:51 (UTC-04:00)	7.5 KB	Standard
1effe58a-4930-4716-9c60-56c2c9b5d6e6.png	png	April 23, 2025, 12:35:08 (UTC-04:00)	2.9 MB	Standard
265bcd3f-d87c-4bc2-84fc-078d8a8429d.jpeg	jpeg	May 9, 2025, 00:13:57 (UTC-04:00)	2.8 MB	Standard
2e4c3002-f00c-4480-990e-77785a4a497.jpeg	jpeg	May 6, 2025, 17:51:42 (UTC-04:00)	7.5 KB	Standard
2f3ccffa-1a75-4e31-ba98-1f5056001494.jpeg	jpeg	May 8, 2025, 01:12:16 (UTC-04:00)	2.8 MB	Standard
42c4d206-6a84-4242-bd4c-f8d83bd9b70d.png	png	April 23, 2025, 12:08:53 (UTC-04:00)	2.9 MB	Standard
49c71f18-ee3c-4ec3-bb76-630d21878c3a.jpeg	jpeg	May 8, 2025, 11:00:37 (UTC-04:00)	7.5 KB	Standard
5038944d-6f6c-4d31-9b99-ba40cf1fc43.jpeg	jpeg	May 8, 2025, 16:05:48 (UTC-04:00)	2.8 MB	Standard
5259d481-876b-4b76-bc89-dd8531aedc4b.jpeg	jpeg	May 5, 2025, 23:34:58 (UTC-04:00)	2.8 MB	Standard
53303634-0606-4300-a081-e55a838039a1.png	png	April 23, 2025, 11:37:50 (UTC-04:00)	2.0 MB	Standard
607da9cf-bc54-4334-9411-04180e000000.jpeg	jpeg	May 8, 2025, 16:05:53 (UTC-04:00)	2.8 MB	Standard

EMR Rankings

The screenshot shows the AWS EMR console interface. The left sidebar indicates the user is on EC2: Clusters. The main area displays a list of clusters, each with a status of "Terminated". The columns include Cluster ID, Cluster name, Status, Creation time (UTC-04:00), Elapsed time, and Normalized insta. A "Create cluster" button is located at the top right.

Cluster ID	Cluster name	Status	Creation time (UTC-04:00)	Elapsed time	Normalized insta
j-3SIZ4BFAI30VX	post-ranking	Terminated Auto-terminate	May 08, 2025, 14:48	1 hour, 43 minutes	48
j-39NLG0I9458KT	post-ranking	Terminated with errors Instance failure	May 07, 2025, 16:39	2 hours, 56 minutes	72
j-2SYK2D5RGZHSH	post-ranking	Terminated with errors Instance failure	May 07, 2025, 00:24	3 hours, 5 minutes	72
j-5PRFLXT276Q	post-ranking	Terminated with errors Validation error	May 07, 2025, 00:22	11 seconds	0

VPCs

Your VPCs (1) Info							
<input type="text"/> Find VPCs by attribute or tag		Actions					
Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR	DHC	dopt
-	vpc-0b65950e5c6adbc11	Available	Off	172.31.0.0/16	-	-	dopt

RDS

Aurora and RDS < Databases							
Databases (3)							
<input type="text"/> Filter by databases							
DB identifier	Status	Role	Engine	Region ...	Size	Recommendations	CPU
imdb-basic	Available	Instance	MySQL Co...	us-east-1d	db.t4g.micro	2 Informational	
instalite-attempt-2	Available	Instance	MySQL Co...	us-east-1f	db.t4g.micro	2 Informational	
installite-2y2b-nets2120-project	Available	Instance	MySQL Co...	us-east-1d	db.t3.micro	2 Informational	