# Final Project Report: NETS 2120 - InstaLite Social Media Platform

## System Components and User Stories Implementation

User Signup and Registration Feature

Features Implemented:

- Account Creation:
    - Users can sign up by providing essential details such as username, password, email, affiliation (e.g., Penn), and birthday. This process is the first step to creating a personalized user experience on InstaLite.
    - Profile Photo Upload: During the signup process, users are prompted to upload a profile photo. This feature enhances user recognition and personalization. For mobile users, the functionality to take a selfie has been integrated, making the platform more accessible and user-friendly.
    - Hashtag Selection: We implemented a feature allowing new users to select hashtags of interest during signup. The system displays the top 10 most popular hashtags to help users start engaging with content right away, even before they build a network of friends or generate posts.

Technical Description:

- Backend Processing: The signup and registration process is handled through our Node.js backend, interfacing with an Amazon RDS instance. This setup ensures that user data is managed securely and efficiently.
- Security Measures: To secure user accounts, passwords are salted and hashed before storage. This security measure protects user data against unauthorized access and ensures that our platform maintains high standards of data integrity.
- Database Design: User data, along with their selected hashtags and profile images, are stored in a relational database schema that supports quick retrieval and efficient management of relationships between different data entities.

User Login and Security Feature

Features Implemented:

- Secure Login: Users can log in with their username and password. This process is streamlined to ensure a seamless user experience while maintaining high security standards.

- Password Reset (Extra Credit): For users who forget their passwords, we implemented a password reset feature. This feature sends a reset email with a token and a link to the password reset screen, allowing users to regain access to their accounts securely.

Technical Description:

- Session Management: We use sessions to manage user logins securely. The session information is encrypted and stored temporarily on the server to maintain state and verify user credentials during active sessions.
- Reset Mechanism: The password reset functionality is integrated with our email service provider through backend services. When a user requests a password reset, a secure token is generated and sent via email, which is then used to validate the password reset request and safeguard against unauthorized access.

Profile Management and Search Features Implementation:

Features Implemented:

- Profile Management
    - Update Profile Information: Users can edit their profile information, including username, email, and password, without triggering a status post. This feature ensures that personal details can be updated securely and privately.
    - Actor Profile Linking: InstaLite allows users to associate their profile with that of a known actor by analyzing the user's selfie and comparing it with the profile photos of the five most similar actors based on embedding similarity. This feature leverages ChromaDB's embeddings for accurate matching.
    - Updating Hashtags of Interest: Users can update their interests to receive more relevant content in their feeds. Recommendations are provided for new hashtags to keep users engaged with trending topics.
    - Changing Associated Actor: Users can change their associated actor profile post-registration. The new association triggers an automatic status post, e.g., "Alain is now linked to Sem Ferid." As before, users can only choose from the five most similar actors by embedding similarity.
- Search and Notifications
    - People and Actor Search: The platform provides comprehensive search functionality for users to find friends and people of interest, including actors. This feature utilizes a retrieval-augmented generation (RAG) mechanism combined with a Large Language Model (LLM) to identify relevant results by name, affiliation, or interests.
    - Post Search: Users can search for posts by specific hashtags or Twitter feed content to find particular discussions or topics. The results provide immediate access to relevant conversations, enabling users to engage quickly.
- Friend and Hashtag Actions (Extra Credit): The search functionality includes options for adding friends or following new hashtags directly from the search results, making it easy for users to expand their network and interests.

- Friend Requests Notification (Extra Credit): Notifications are triggered for both incoming and outgoing friend requests. This ensures that users stay informed of their social interactions.
- Chat and Message Notifications (Extra Credit): When a user receives new chat invitations or messages, a notification is sent to them. If the friend accepts a chat invitation, a session is created; if rejected, no session is formed.

Technical Description

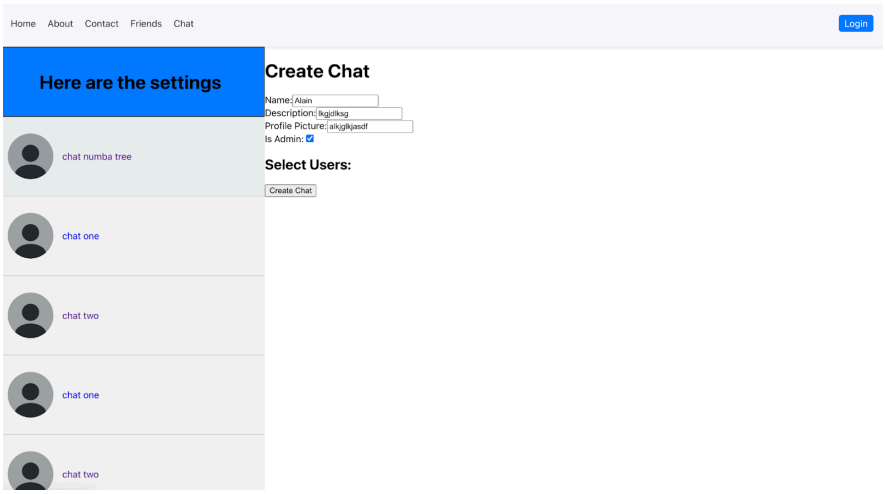
- Profile Management Backend: Profile management tasks, including data updates and linking with actor profiles, are handled by the backend services implemented in Node.js. The services interface with an Amazon RDS database to manage user profiles efficiently.
- Search Mechanisms: The search feature uses retrieval-augmented generation (RAG) with a Large Language Model to sift through indexed content, actor profiles, and user posts. By leveraging embeddings and LLM capabilities, the search provides accurate and relevant results.
- Notification System: Notifications for friend requests and chat messages are generated using a combination of backend services and WebSocket connections. This system ensures that notifications are delivered in real-time, keeping users engaged and informed.

Lessons Learned

- Embedding Similarity: One challenge was ensuring accurate actor profile matching using embeddings. After careful consideration, we chose ChromaDB due to its efficient handling of similarity search tasks.
- Retrieval-Augmented Search: Implementing retrieval-augmented search required us to balance the accuracy of results with performance. The integration of LLM and efficient indexing provided a good solution.
- Notifications Infrastructure: The notification system required implementing an efficient mechanism for managing and delivering real-time notifications. We opted for WebSocket-based communication to ensure timely and consistent notification delivery.

Conclusion

- Implementing the profile management, search features, and notification system in InstaLite has been a valuable learning experience. These components add significant value to the platform, offering users a personalized and engaging social media experience. We are confident that the architecture and features we implemented will allow InstaLite to be scalable and adaptable to future needs.




**Network and Chat Management Features Implementation**

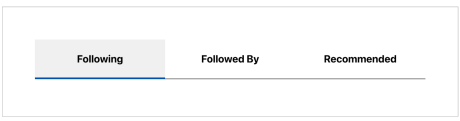
Features Implemented

1. Network Management
2. Sending Friend Requests (Extra Credit): Users can send friend requests to other users to expand their social network. This feature helps users easily build their community.
3. Public Profiles (Extra Credit): If a user has a public profile, they can be added directly as a friend without requiring a friend request, streamlining the connection process.
4. Accepting or Denying Friend Requests: Users have the flexibility to accept or deny incoming friend requests, ensuring control over their friend list. Accepted requests establish a friendship, while denied requests do not.
5. Removing Friends: Users can remove connections they no longer wish to stay in touch with, providing a way to curate their friend list.
6. Friend Recommendations: InstaLite provides personalized friend suggestions based on a user's interests and network activity, offering new connections that are likely to match their social preferences.
7. Current Friends and Online Status: Users can view their current friends and their online status, helping them identify which friends are actively using the platform.
8. Chat Management
9. Inviting Friends to Chats: Users can invite friends to join chat sessions, facilitating conversations. If accepted, the invited friends receive a notification and a chat session is established.
10. Joining and Leaving Group Chats: Users can join or leave group chats to interact with multiple friends at once. A chat session continues as long as it has at least one active member. The session is automatically deleted when the last member exits.
11. Chat History Preservation: Chat history is retained for each session, allowing users to revisit past conversations easily.
12. Creating Group Chats: Users can form group chats by inviting additional members to ongoing chat sessions. Each group chat has a unique chat history starting from the session's creation, even when involving the same users.

13. Consistent Message Ordering: All chat messages are ordered consistently across members using timestamps, ensuring that everyone sees messages in the same sequence.
14. Daily Friend/Follow Recommendations: Every day, InstaLite recomputes recommendations to help users discover new friends and connections who align with their interests and activity.
15. Technical Description
16. Network Management Backend: The network management tasks, such as friend requests, public profiles, and recommendations, are managed by a backend system implemented in Python, which interfaces with a PostgreSQL database to handle user relationships.
17. Chat Management System: The chat system uses WebSockets for real-time message delivery and a relational database for managing chat sessions and user participation. Timestamps are leveraged to maintain consistent message ordering.
18. Notification System Integration: Notifications for friend requests and chat invitations are tied into the notification feature to keep users informed of their social interactions and opportunities to join chats.
19. Nontrivial Design Decisions and Lessons Learned
20. Public Profiles Implementation: Implementing the public profile feature required careful UI/UX design to clearly differentiate public and private profiles and streamline the friend addition process.
21. Group Chat Handling: Designing independent group chat sessions presented challenges in terms of maintaining unique chat histories while preventing redundant session creation.
22. Notification and Consistency: Ensuring notifications were timely and consistent across users required a solid WebSocket-based system to keep real-time updates synchronized.

The network and chat management features implemented in InstaLite have significantly enhanced the platform's social connectivity. From personalized friend recommendations to seamless chat invitations, these features collectively deliver a comprehensive and intuitive experience. The network management backend and chat system architecture ensure scalable and consistent performance, setting a strong foundation for future growth.



**Feed and Post Management Features Implementation**

Features Implemented:

Feed Management:

- Unified Feed of Posts: Users can view a comprehensive feed that combines posts from friends, prominent figures, the social media stream (Twitter/X feed), and their own posts. This feed is updated hourly with fresh content from Kafka and other recent posts. The precomputed data makes loading the feed seamless upon user login.
- Prominent Figures Filtering: The feed prioritizes content from prominent figures who have a high social rank, based on the prior implementation of the social rank algorithm.
- Content Ranking: The server processes new content hourly to rank and score posts. Kafka is leveraged to fetch the most recent updates, enabling efficient data computation for the client.

Posting Features

- Creating Posts: Users can create posts that include text, hashtags, and images, either individually or in combination. At least one of these elements is required.
- Privacy Controls (Extra Credit): Users can control the visibility of their posts by marking them as private, friends-only, or public. This empowers users to manage their audience effectively.
- Commenting on Posts: Users can engage with posts by commenting on them in a threaded manner, fostering meaningful discussions within the community.
- Infinite Scrolling (Extra Credit): The feed supports infinite scrolling, allowing users to continuously load new posts. Alternatively, the platform also supports pagination for users who prefer it.
- Liking Posts: Users can like posts to show appreciation and support, contributing to the engagement metrics of each post.
- Ranked Personal Posts: Users can see their own posts ranked based on the Spark Adsorption algorithm, ensuring they have an organized view of their most relevant content.
- Cross-Platform Posts (Kafka FederatedPosts Channel): Posts from other Nets2120 InstaLite platforms are included in the feed using the Kafka FederatedPosts channel. Each post is associated with a post_json object to streamline data sharing across platforms.
- Navigation
- Navigation Bar: A navigation bar provides easy access to essential features like the profile page, friend management, chat mode, and search.

Technical Description

- Feed Management Backend: The backend uses Python and Apache Kafka to collect and process the latest updates, leveraging prior social rank implementations and custom ranking algorithms. Precomputed feed data is made available via efficient caching.
- Posting Features Backend: The posting features use a combination of Python and PostgreSQL to manage posts, comments, and privacy settings. Kafka plays a key role in integrating posts from external InstaLite platforms.
- Navigation System: The navigation bar is implemented with responsive front-end design to offer quick access to various features.

Lessons Learned

- Feed Ranking and Refresh: Developing an efficient feed-ranking system required careful tuning to balance content freshness and relevance.
- Infinite Scrolling vs. Pagination: Providing both infinite scrolling and pagination ensures flexibility in user experience, requiring separate handling of these features.
- Cross-Platform Compatibility: Integrating posts from external platforms via Kafka highlighted the importance of consistent data formatting and robust backend architecture.

Conclusion

- The feed and post management features enhance InstaLite's core functionality, providing users with a rich and dynamic content discovery experience. From highly customizable posting capabilities to sophisticated feed ranking and cross-platform integration, these features deliver on the vision of a comprehensive social media platform. The technical architecture ensures reliability and scalability, setting the stage for future enhancements.