Lemon Pie (g06): Eric Chen, Jason Gao, Aditya Sirohi, Lucas Wu
NETS2120
Professor Zachary Ives
May 8th, 2024

Final Report

System Overview

We built Lemonstagram, a Lime (Lite) version of Instagram, implementing the major functionalities that Instagram provides, including a dynamic user profile, live feed algorithm, posts, chats, and follow requests. Additional features implemented in this class were also integrated within our platform such as actor mapping. Lemonstagram was developed on a React frontend with a node.js/express.js backend.

Technical Features

Here is an overview of each of the technical feature we have developed:

- Posts/Feed Routes (feed_routes.js) + frontend (home.tsx, createPost.tsx, etc.)
    - Functionality to create posts and curate the user's home feed based on user posts, friend posts, public posts, and Twitter (X) Kafka posts
    - Comment and like functionality on posts allowing users to like/unlike and comment/delete comments
    - Database tables for comments and likes maintains persistence via user interactions
    - Front end formatting for the home feed, navigation menu, and posts
- SocialRank (ComputeRank.java + CreateGraph.java)
    - Implemented an adsorption algorithm using SparkJava and PairRDDs to rank users and posts for the purposes of curating a feed/recommendations
    - Creates a social graph of all the hashtags, users, and posts by querying through the RDS database
- Kafka Streaming
    - Set up Kafka consumers to listen to the Twitter-Kafka and FederatedPosts Topics by processing the json files to register other group's users in our database & publish posts
    - Set up a Kafka producer to allow our group to send our posts to the FederatedPosts Topic using our group source: g06
- Chat Routes (chat_routes.js) + frontend (ChatHome.tsx, Chat.tsx)
    - Chats features include chat invites, messaging, joining/leaving chats, and creating chats
    - Designed specifically to ensure persistence of chats, chat_messages, and chat_members across user sessions
    - Timestamps utilized to maintain proper ordering of the chat messages when displaying them on the front-end for users
- User Routes (user_routes.js)
    - Handled functionality related to the user, including basics used during startup like registration and login - UI/UX designed to facilitate flow similar to Instagram.

    - Routes that allow users to follow, unfollow, send requests, etc. with everything stored in our follower database and enables users to accept
    - Handles profile changes and image uploads to S3 - functionality reused for posts
- Face API / Face Matching with ChromaDB (actor_routes.js)
    - chroma-access.js: connect to ChromaDB, generates embeddings for images, upload embeddings to database, and contains other set-up functions
    - actor_routes.js: generates embedding for user profile picture by querying image table for image URL (image upload via s3 to table is completed in user_routes.js), return closest actor image recommendations and let user choose from actor image recommendations which links between the user and actor IDs

Design Decisions

Lemonstagram's architectural strategy was designed to allow for modular development such that elements of both the frontend, backend, and additional features could be developed, tested, and integrated seamlessly.

Frontend: Diving into the frontend structure, the approach was to mirror the existing Instagram UI/UX structure with a constant navigation menu and utilize that page as a central hub. There were separate pages for each feature (Home, ChatHome, Profile, EditProfile, Login, etc.) which contained the design and integration with the backend routes which will be discussed later in this report.

- Login/Signup Screen: this landing page is mirrored to Instagram's approach
- Home: after registering and logging into a user session, this is the main feed display
- Friends: friend recommendations, friend invites, requests, and mutual friends
- Chat: chat features include inviting friends, accepting and creating chats
- Profile: ability to follow, unfollow, link actors, edit profile (bio, photo, etc.)

Backend - The backend consisted of the routes for each feature implemented in addition to the Apache Spark social rank computation, Face Matching API integration for actor mapping, and models with files for creating the RDS tables, ChromaDB setup, and Kafka utilization.

- Routes: actor_routes.js, user_routes.js, chat_routes.js, feed_routes.js, etc.
- Models: chroma_access.js, db_access.js, create_tables.js, kafka,.js
- Other files: ComputeRank.java, CreateGraph.java, chroma_data, etc.

Challenges & Learnings

1.) *Collaboration*: For most of our members, this was the first time developing a full stack project with so many technology stacks and within a team setting. As a result, we learned the importance of collaboration in terms of segmenting software development, unit testing, and clear communication to facilitate successful merges.

2.) *Integration:* This project incorporated many topics that were covered in NETS2120; however, the application of the features and technologies into one comprehensive application was initially difficult to understand. After breaking down each component, we were better able to understand how to integrate each component into a cohesive product: Lemonstagram.

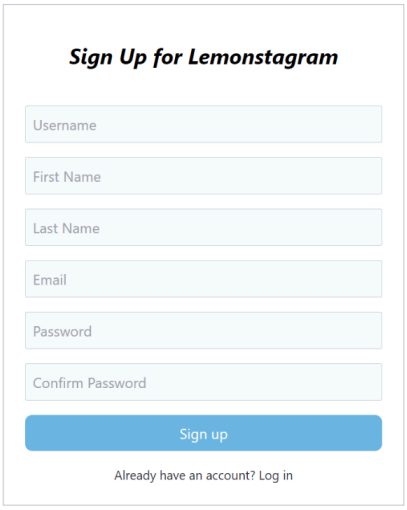
3.) *Project Management:* Given that we were all working on various components and different stages, understanding the importance of hitting milestones and unit testing were essential to avoid bottlenecks. The majority of the features were developed in isolation (e.g. getting Kafka to work in a vacuum and then working to link it to the home feed functionality via routes). This project helped us develop a sense of project management and how to successfully complete future projects.

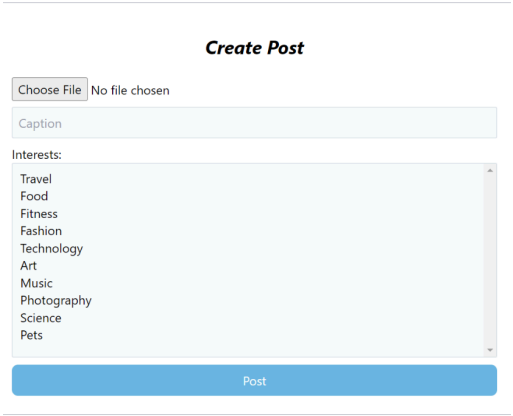
Extra Credit Features

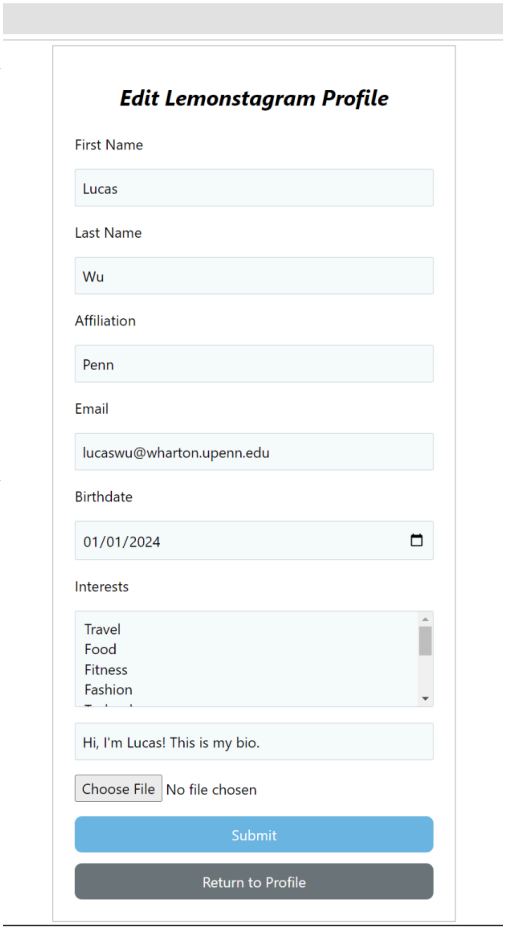
- LinkedIn style connect and decline requests where users can "confirm" or "delete" follow requests. If confirmed, the user is then able to request to "follow back."
- Privacy controls were implemented building off our follow/follow-back system such that a user's home feed is only composed of their posts, their friends' posts, and public posts such as Twitter (X) and federated posts. This privacy control system was essential in creating the privacy controls that Instagram utilizes.
- File handling between the frontend and backend utilized Multer, a node.js middleware. Multer was used also for the S3 container/actor mapping endpoints.

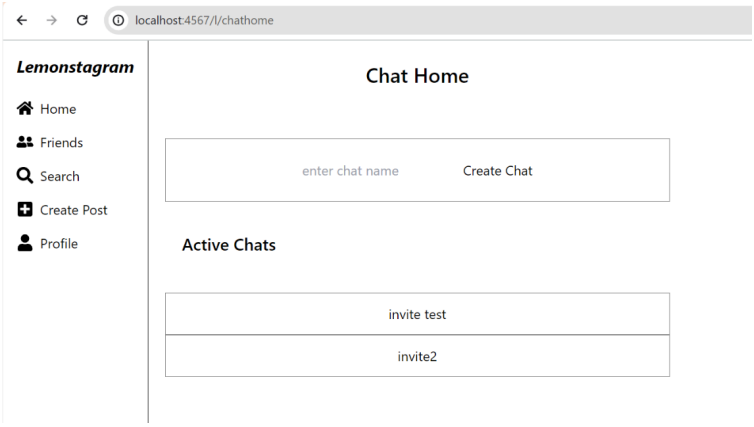
System Applications Images:

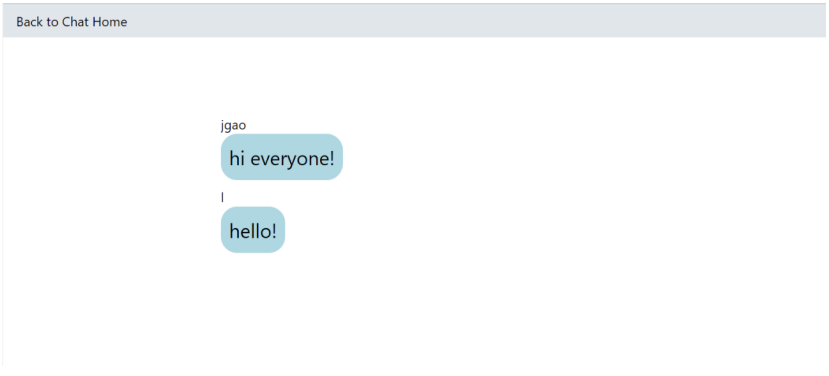
System Application Images

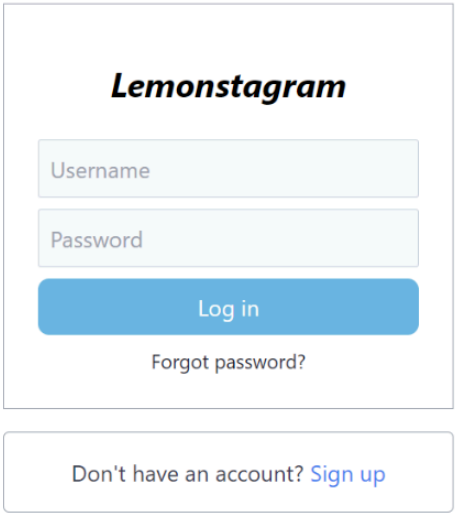


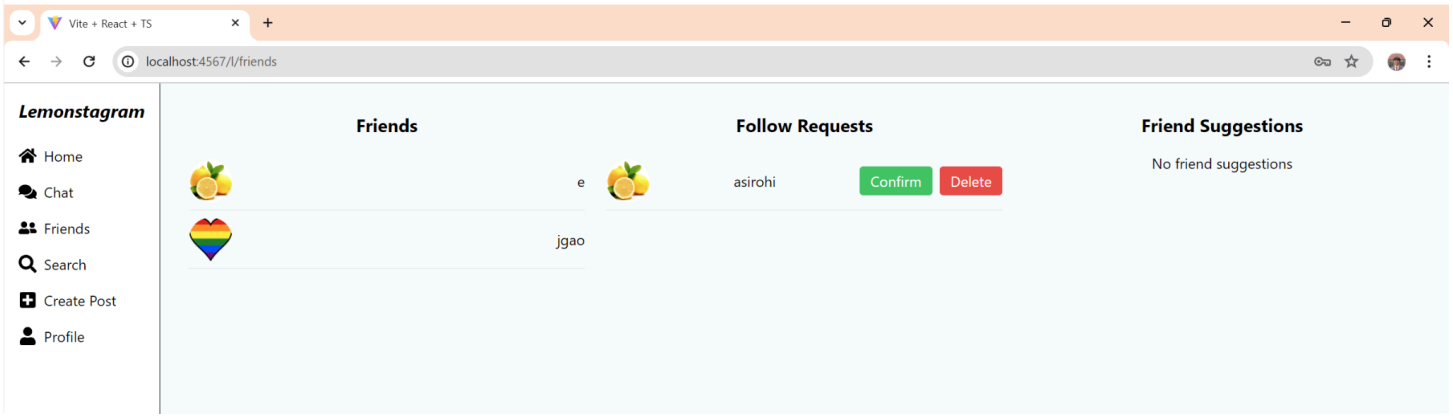


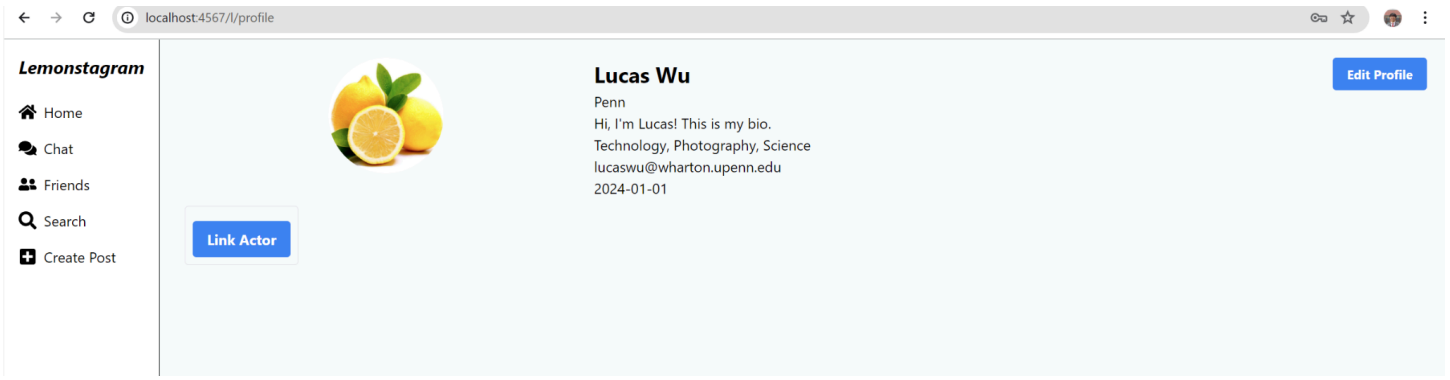


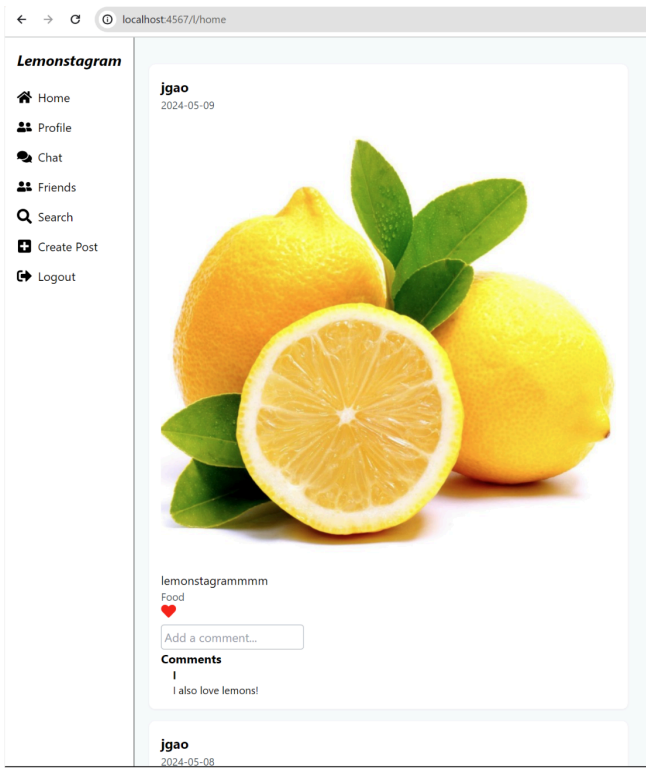







Vite + React + TS
localhost:4567/l/friends
Lemonstagram
Home
Chat
Friends
Search
Create Post
Profile
Friends
e
jgao
Follow Requests
asirohi
Confirm
Delete
Friend Suggestions
No friend suggestions


localhost:4567/l/profile
Lemonstagram
Home
Chat
Friends
Search
Create Post
Lucas Wu
Penn
Hi, I'm Lucas! This is my bio.
Technology, Photography, Science
lucaswu@wharton.upenn.edu
2024-01-01
Edit Profile
Link Actor


localhost:4567/l/home
Lemonstagram
Home
Profile
Chat
Friends
Search
Create Post
Logout
jgao
2024-05-09
lemonstagrammmm
Food
Add a comment...
Comments
I
I also love lemons!
jgao
2024-05-08