Twitter Engine v2.0

COP5615 – Distributed Operating Systems

Submitted By

Sreenivasa Sai Bhasanth Lakkaraju – 41602287 – slakkaraju@ufl.edu

Meghana Reddy Voladri – 43614999 – mvoladri@ufl.edu

Objective:

The goal of this project is to implement front-end for the backend functionalities of Twitter-like engine that we developed in project 4.1.

Our Approach:

We have created UI with HTML, Javascript and Bootstrap that acts as an interface to connect the database written in elixir to our phoenix channels.
The UI is a very simple one to display twitter's basic features.

Following functionalities have been implemented as a part of Project 4.1:

1. Register and delete account.
2. Send tweet. We have created random tweets (using alphanumeric characters) with hashtags (#) and mentions (@).
3. User can subscribe to tweets.
4. A user can re-tweet another user’s tweet.
5. Function to query and fetch tweets with specific hashtags, user mentions, etc.
6. Live display of all the tweets if the user is still connected.

Following functionalities have been implemented as a part of Project 4.2:

- Register: User registers from the UI. Every time a new user is registered, we add a user entry to our server-side ets table.
- Login: Through the login page, user logs in to the twitter clone. We keep the user logged in to get details correctly for a single browser. We also have encryption. Password is encrypted and server-side validation is done and user is authenticated.
- Tweet: User tweets here and posts it. It gets stored in the database.
- Query Tweets: Returns all the user’s tweets followed by the user.
- Subscribe: Subscribe to any registered user.
- Retweet: All the tweets that the user gets can be re-tweeted using this option.
- Query Mention: It provides all the tweets with the user's given name. Username should have a prefix @.
- Search Hashtag: It gives all the tweets listed by the user with the hashtag. Hashtag should have a prefix #.

All the above functionalities have been properly tested manually for each functionality.

Implementation:
1. JSON-based API - Phoenix channels have been used to create an API wrapper over the code written in part 1. We have made necessary changes to accommodate the UI requirements. For all the existing features we had in the previous portion, a wrapper was created.
2. Clients are connected using the websockets.
3. In the client's Javascript part, we used sockets to connect to the phoenix channel of the server.

To run the project,

1. Get dependencies: mix deps.get
2. Create database: mix ecto.create
3. Go into the assets folder and install node dependencies: cd assets && npm install
4. Start server mix phx.server

Output:

1. Fully functional end to end twitter clone has been developed.
2. We have already tested the simulation for 100 users and mentioned the metrics in the project report 4.1

Demo Video Link:

https://youtu.be/cLrO7_l1k7c