BITS Pilani
Pilani | Dubai | Goa | Hyderabad

“Streamer” Community App

SPA, Assignment Part 2

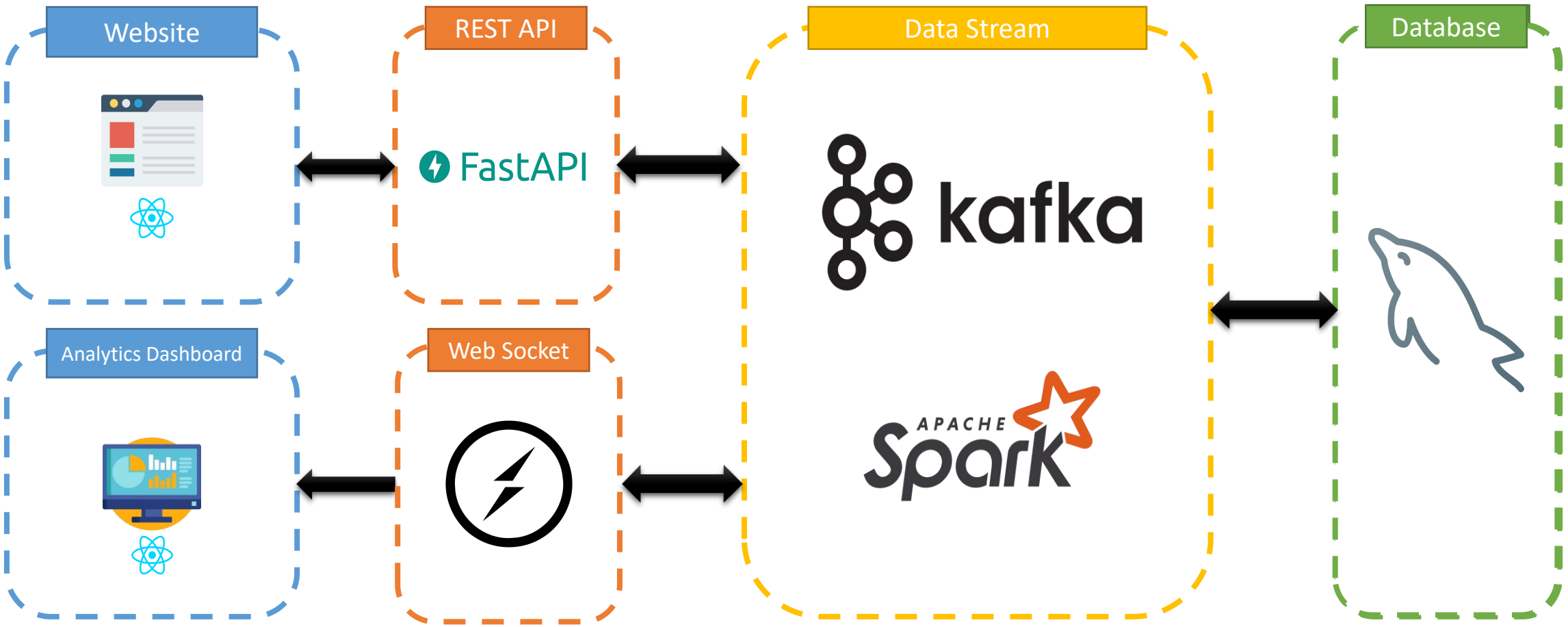
Rishav Saigal
2019AB04045

Mohd Tauseef Ansari
2019AB04067

Agenda

- High Level Architecture (modified)
- Overview of Project Structure
- Web Application (with Analytics Page)
- Fake Data Generator
- Backend API + WebSocket API
- Streaming with Real Time AI
- “Streamer” Application Demo

High Level Architecture (modified)



Overview of Project Structure

Web Application (with Analytics Page)

In this we have build an UI for user to login to respective communitee, view there posts as well as others post also and create posts for that particular group.
Even we have an Analytics Page for viewing the real time various insights what's hapenning behind the application, like how many users are currently logged in, no of posts created, no of likes, no of shares, no of comments and the emotion for the respective posts.

Fake Data Generator

We have created a simple python API request application that will randomly generate the data and feed into our *Backend API* application.

Backend API + WebSocket API

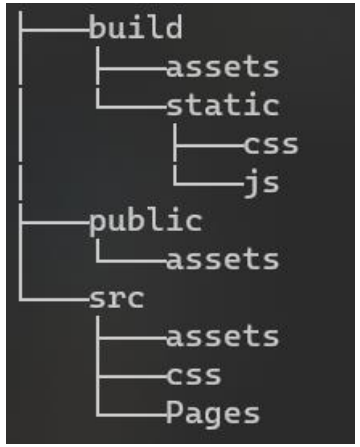
Here we are fetching the resuest from *Web Application* or *Fake Data Generator*. Once we fetch the request we forward it to the Streaming pipeline through kafka producer that consists of kafka and pyspark.
Here we have websocket also to fetch the results through kafka consumers and pass the websocket api, these results further be used by the UI to visualize it.

Streaming with Real Time AI

Here we are fetching the data in batches of 30 seconds time window from kafka producer and passing it to pyspark cluster for further processing of the data and ingesting into the database, once the data is processed from pyspark the result is passed to producer so that the *WebSocket API* can pass back to UI.

Web Application (with Analytics Page)

Folder Structure (web-ui)



Files Shared

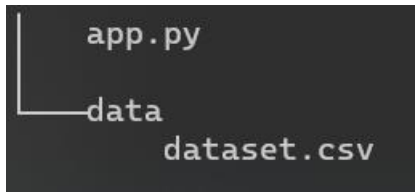
- build (this is a build version from the src folder most of the folder is auto generated using *yarn build* command)
  - assets - this folder consisits of all the ui required images and logos
  - static - within this we css and jss folders
    - css - this holds all the css files of the UI
    - js - this holds all the js files of the UI
  - this folder holds the index.html for entry point of our application
- public
  - assets - this is our working folder where we store all the required images and logos for index.html
  - this folder holds the index.html for entry point of our application
- src
  - assets - this is folder where we store all icons that we use in respective react scripts
  - css - this folder holds the css files which are required for our react scripts
  - Pages - this folder holds sub pages of the application like Analytics.js, CreatePost.js, Login.js, Main.js, and Posts.js
  - App.js, Constant.js, index.js, setupTest.js (these files are entry points of our application and some helper files are here also)
- package.json - this file holds all the configurations of the applications and install packages also.

Command Used

- *yarn install* - for installing all the packages
- *npm start* - for starting the application (in dev enviroment)
- *yarn build* - for building the application for production
- *serve -s build* - this for running the application in production enviroment

Fake Data Generator

Folder Structure (data-generator)



Files Shared

- **app.py - this is our application entry point that will randomly login the users, create posts, likes, shares, comments and this will happen every 30 seconds.**
- **data**
  - **dataset.csv - holds sample text data that is used for passing into our python Backend API**

Command Used

- ***python app.py* - for starting the application**

Backend API + WebSocket API

Folder Structure (backend-api)

```
api_models.py
app.py
database_models.py
```

Files Shared

- **app.py - this is our application entry point here all the topics are created which are required for our analytics, we also have respective routes to fetch the requests from *web-ui* and *data-generator***
- **api_models.py - this a helper class model that holds the structure of the database**
- **database_models.py - this is our database controller, for fetching the data and inserting into database.**

Command Used

- ***uvicorn app:app --reload* - for starting the application**

Folder Structure (node-kafka-websocket)

```
app.js
package-lock.json
package.json
```

Files Shared

- **app.js- this is our application entry point, here we are setting a consumer that will fetch all the json messages from the topics**
- **package.json - this file holds all the configurations of the applications and install packages also.**

Command Used

- ***node app.js* - for starting the application**

Streaming with Real Time AI

Folder Structure (pyspark-kafka)

```
app.py
mysql-connector-java-8.0.26.jar
process_message.py
```

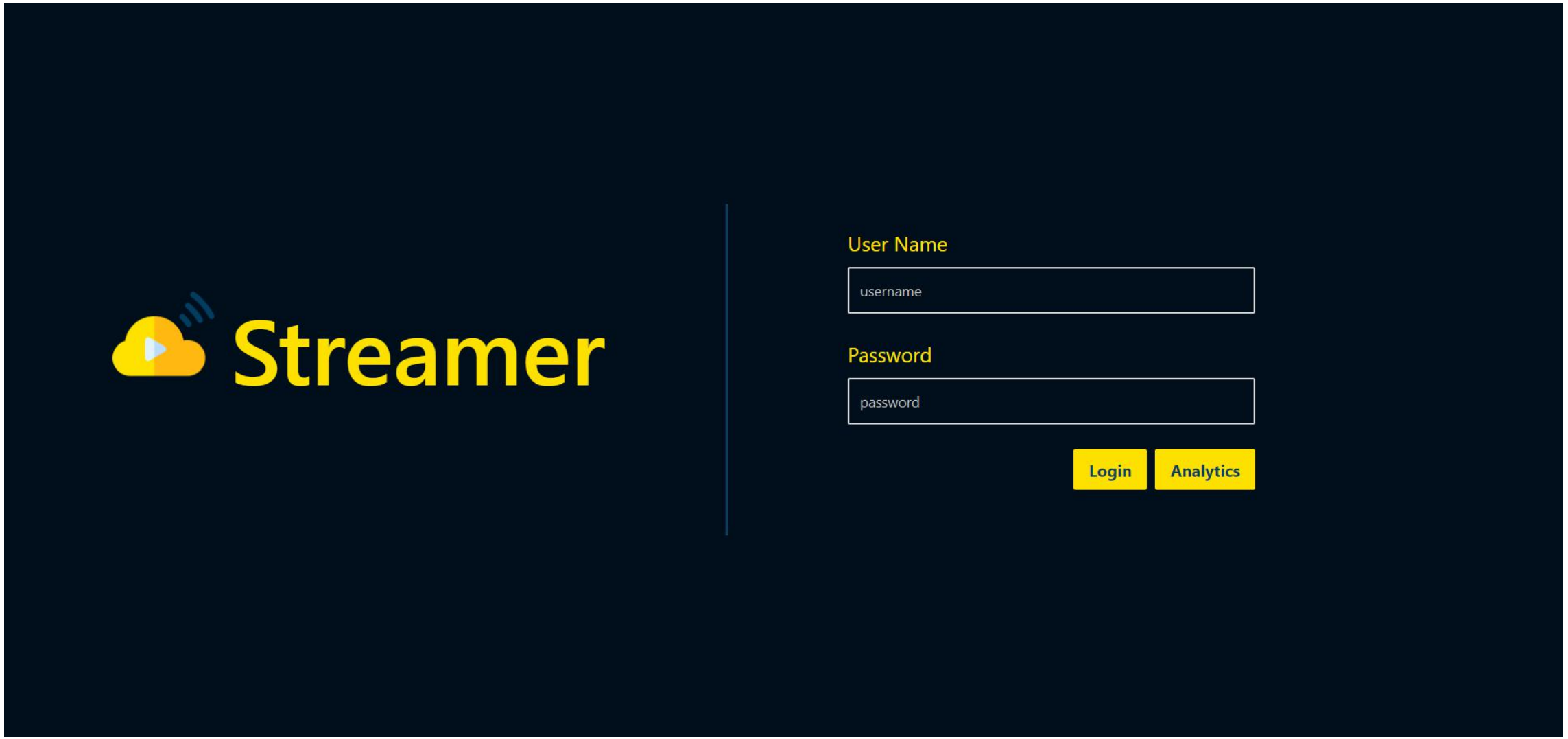
Files Shared

- **app.py - this is our application entry point, here we are creating consumers and fetching the json messages from kafka topics and laters in batches of 30 seconds window frame we are processing the results.**
- **process_message.py - this is a helper file for ingesting processed data to database and perform ML analytics.**
- **mysql-connector-java-8.0.26.jar - this is a jar file which is used for connecting to the mysql database.**

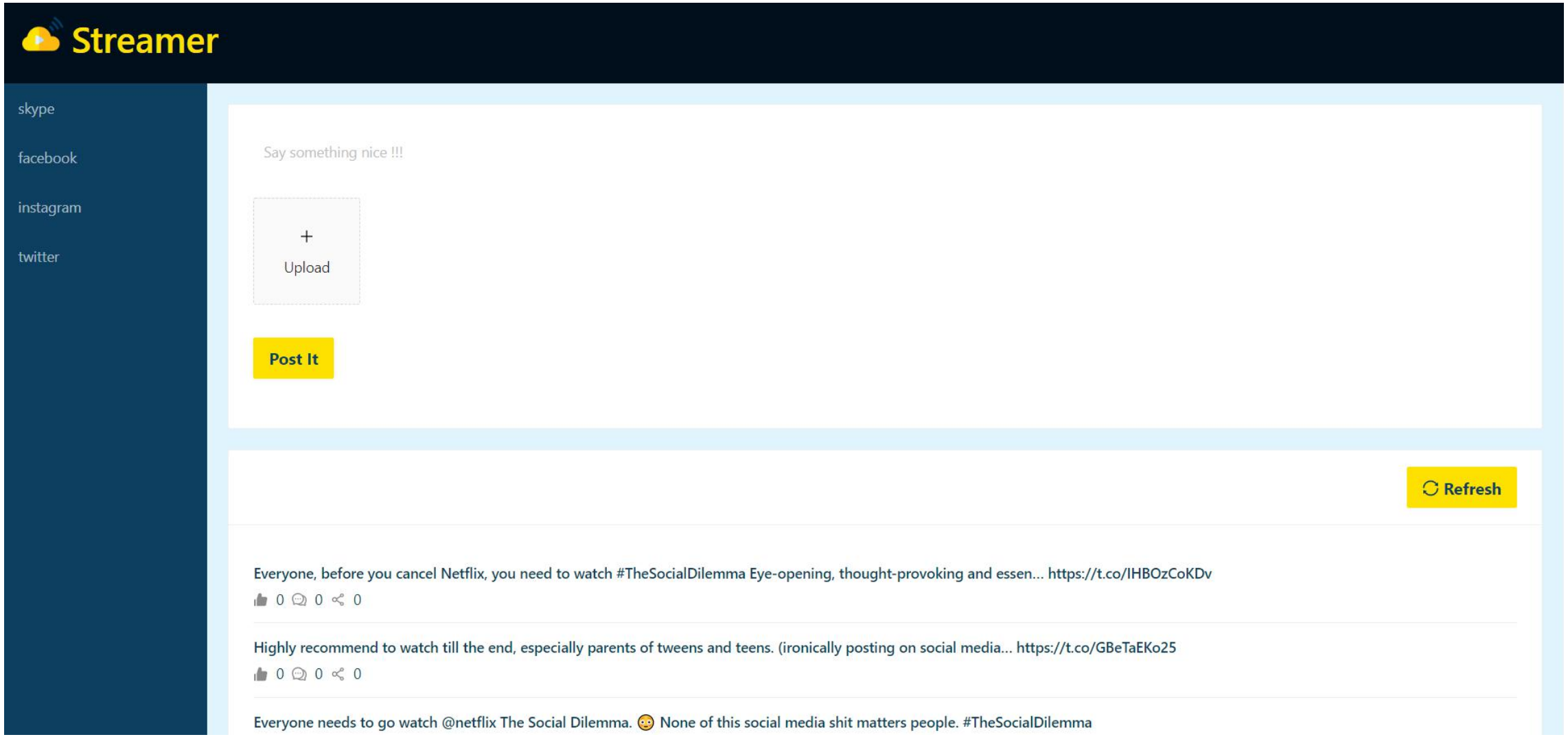
Command Used

- ***faust -A app worker -l info* - for starting the application**

Login Page


Streamer
User Name
username
Password
password
Login
Analytics

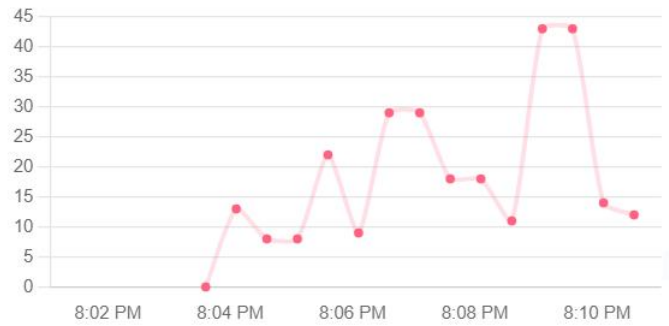
User Page


Streamer
skype
facebook
instagram
twitter
Say something nice !!!
+
Upload
Post It
Refresh
Everyone, before you cancel Netflix, you need to watch #TheSocialDilemma Eye-opening, thought-provoking and essen... https://t.co/IHBOzCoKDv
0 0 0
Highly recommend to watch till the end, especially parents of tweens and teens. (ironically posting on social media... https://t.co/GBeTaEKo25
0 0 0
Everyone needs to go watch @netflix The Social Dilemma. 😳 None of this social media shit matters people. #TheSocialDilemma

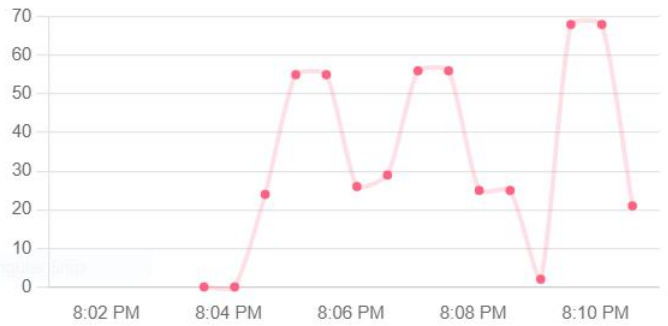
Analytics Page

Streamer Analytics

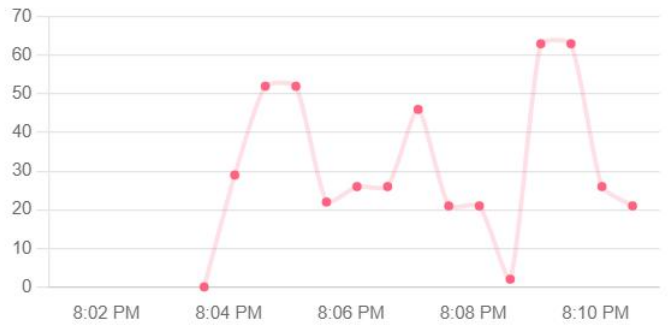
Active Users Trend



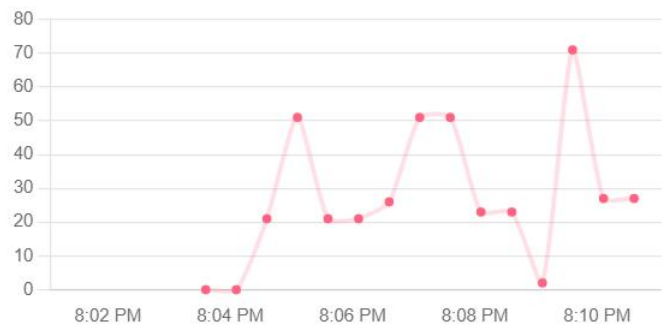
Likes Trend



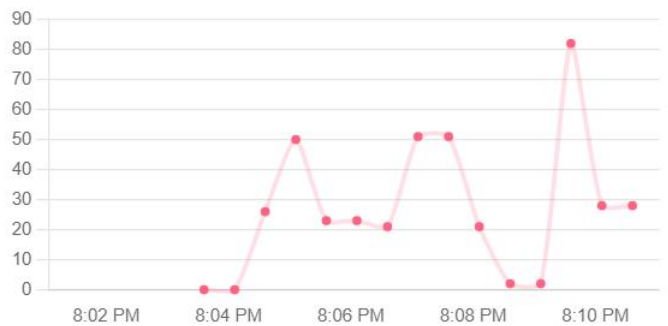
Posts Users Trend



Comments Users Trend



Shares Users Trend



User Emotion over posts