**COLLABORATIVE RECOMMENDER SYSTEM WITH RESTful API**

**Similarity Calculation Used:**

Cosine similarity: Measure of similarity between two documents (in our case two user) that measures the cosine of the angle in between them.

Why use cosine similarity:

* Performs very well when the data is in 0 to 1 range. I normalized the data to fall in zero to one range using min max scaler.
* Low time complexity, as an initial baseline and short deadline (6-8 hours), I chose to put more detail and thought in building and deploying the API with front end.
* Works well with sparse matrices.

What could be done next:

* Use a mix of supervised and unsupervised learning. i.e. cluster users into optimal clusters and then build models for such clusters.
* Experiment with dimensionality reduction methods like PCA and SVD.
* Based on time and trend select features that best represents the user’s current interests.

Working at scale:

* Data Engineering:
  + Set up timely jobs to load new data into the user tables
  + Index entries by time to make calls faster.
* ML pipeline:
  + Enable parallel processing and continuous updates to the scores table.
  + Enable sparsity reduction techniques.
  + Investigate and use the features that best represent the user behavior using metrics like course completion ration, continuous view streak, test scores, user activity and more.
  + For example, a user matches another user, but that user is inactive discount the similarity score.
* User Facing ML:
  + Optimize the suggestion display position on the screen by making use of usability metrics like eye tracking, intent to move vs. click conversion etc.

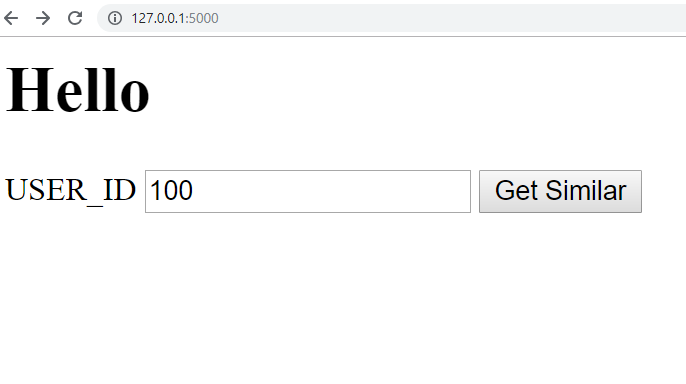
Other data that I would like to know:

* Number of active users’ trend.
* Does test score mean anything?
* Incentive to complete/ stay logged on.
* Course author ratings are very important as most people turn to the best instructor available.
* Are there any course paths in the system for example a sequential list of courses towards a goal, this can be useful to group people together and increase the similarity ranking?

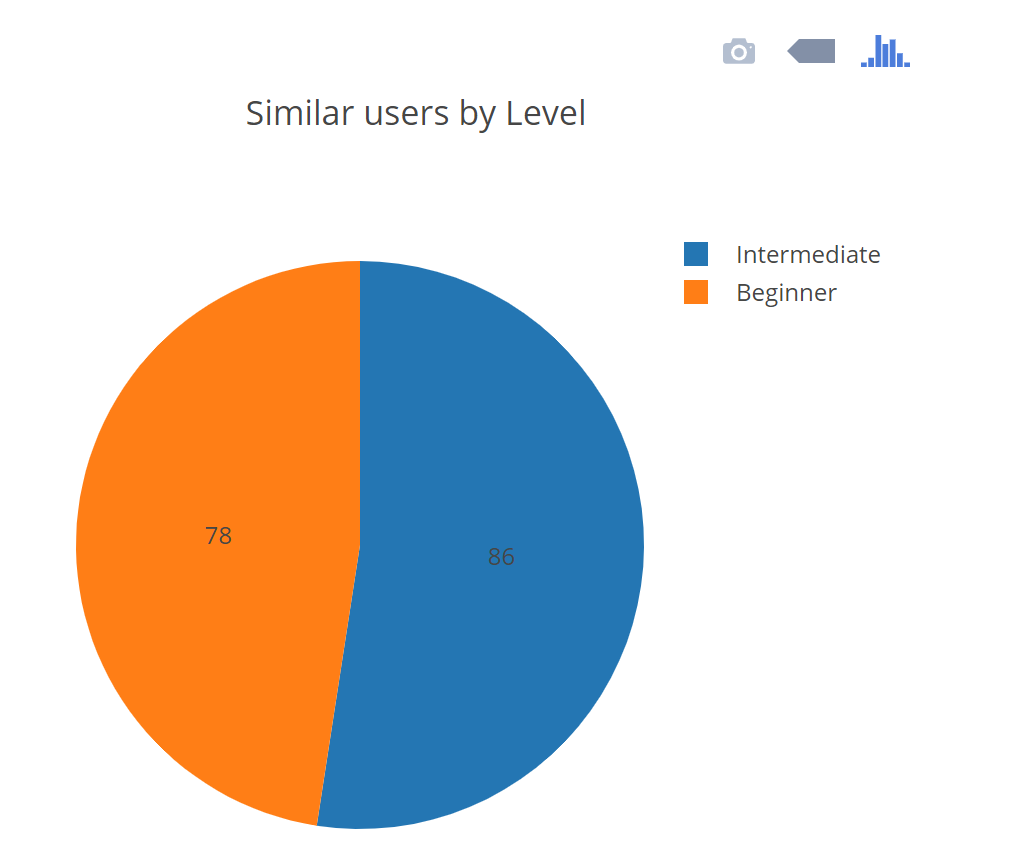
Application design steps:

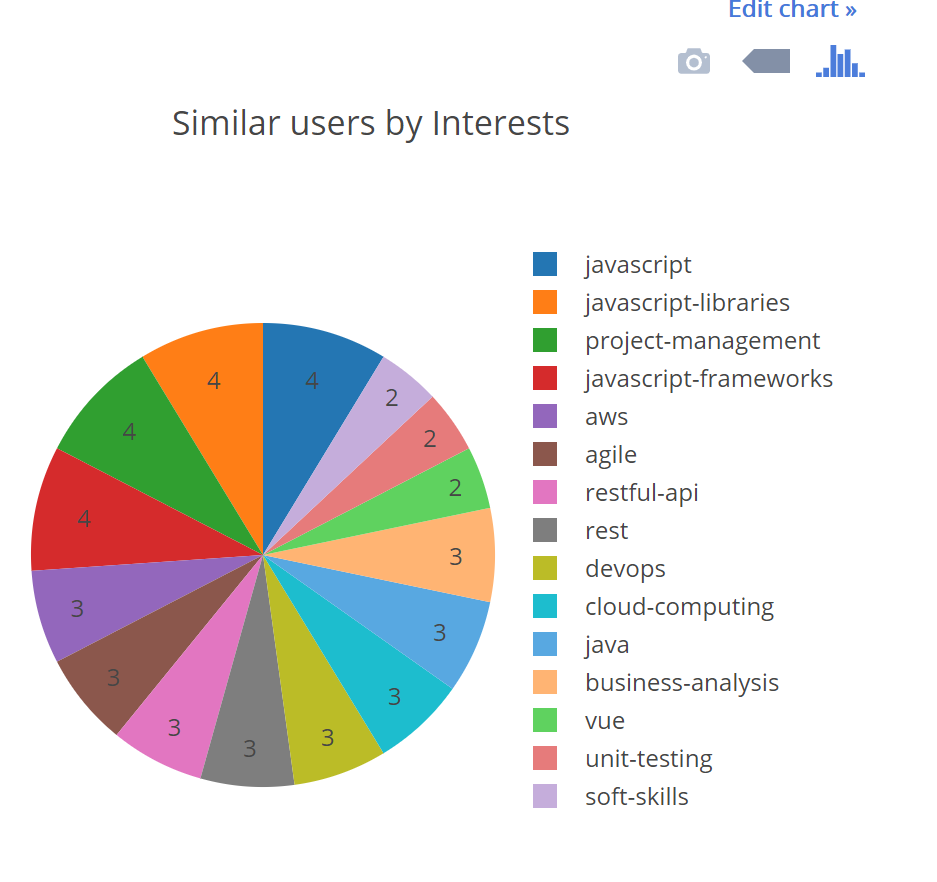
* Explore the data set (1-2 hours): Played around with the data set to notice that it would be sparse as expected. Also, test assessment is of no use as they do not account for at least 50% of the users. Using the information would only increase sparsity.
* Design and create database functionality: I integrated SQL backend into the application as I think that it is the right way going forward, holding data frames in memory might prove costly later.
* The above step also includes calculation of similar users. In this version only 10 most similar users are retained.
* Set up application and testing basic functionality
* Set up a landing page to input the user query id and output dynamic Plotly graphs showing the summary of the most similar users.
* Adding write ups, making build and committing final commit.

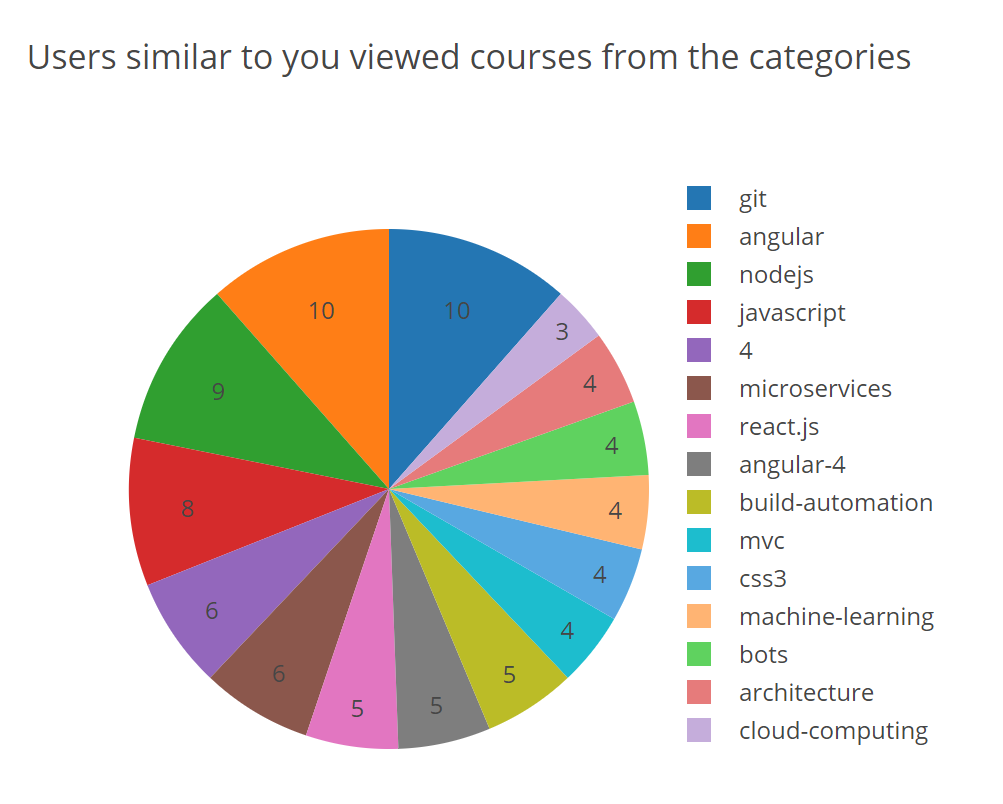
Landing page:



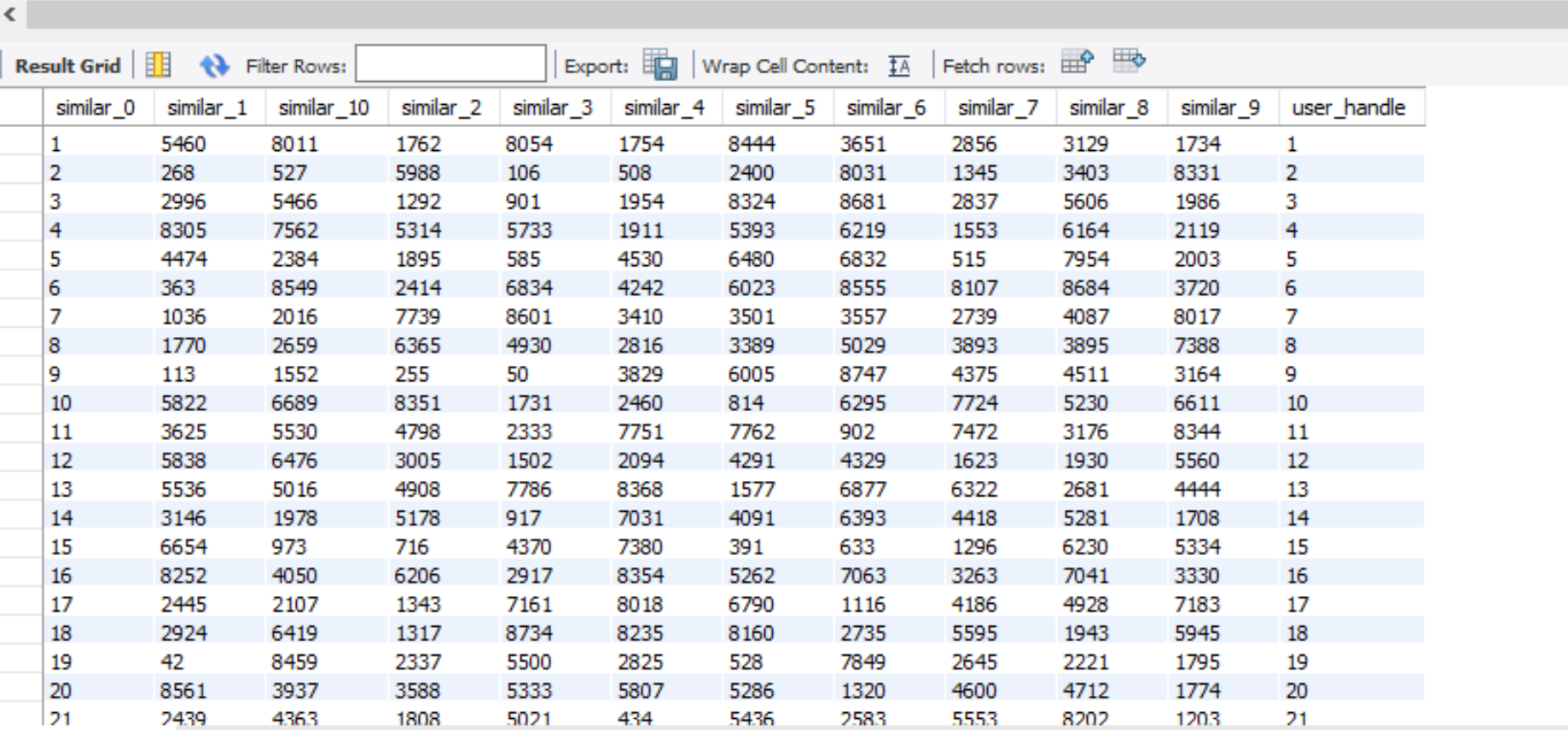
Dashboard page:







Database similar users:



Improvements/ Next steps:

* Better similarity metrics
* Improved testing suite
* Acquire more data and set up a pipeline
* Improvise front end using dask