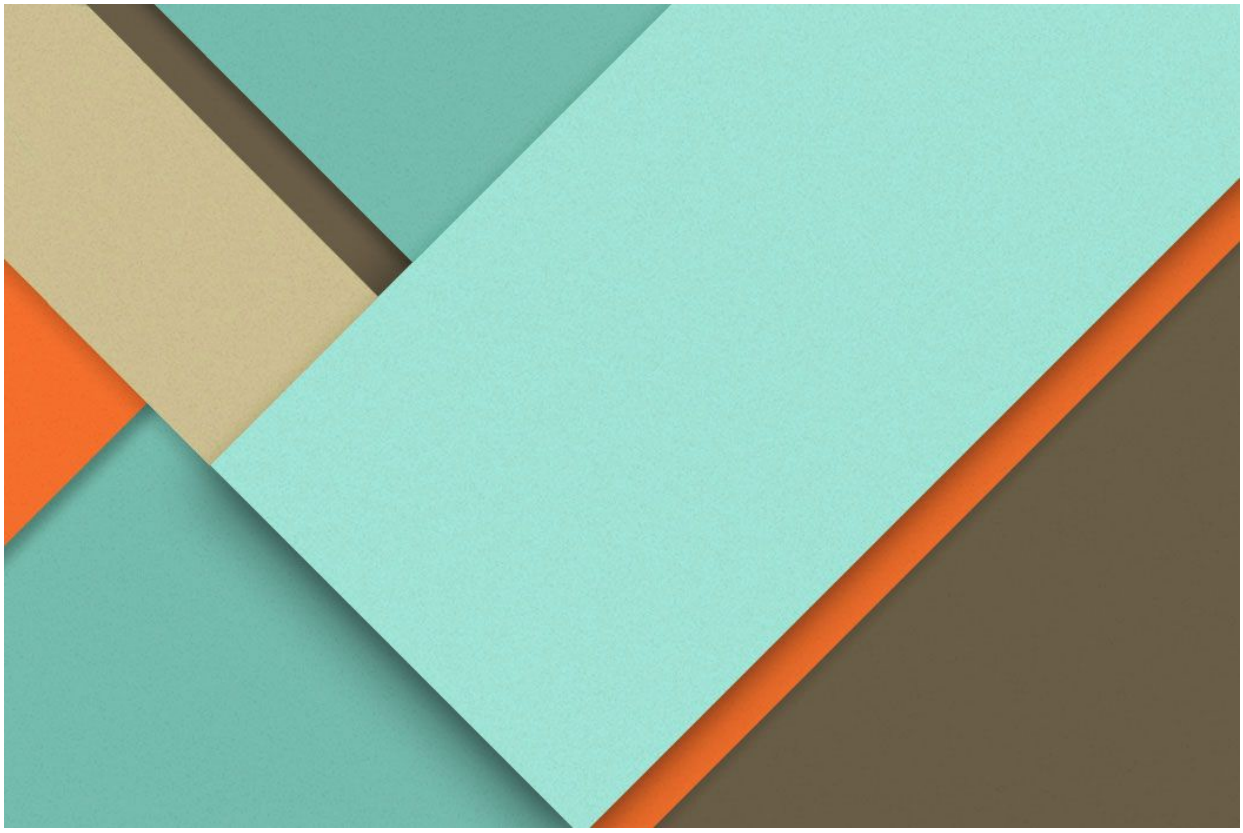


IDK - (A StackOverflow Clone)

24.11.2018

Parashara R	01FB15ECS202
Rahul Mohan	01FB15ECS222
Rahul Pillai	01FB15ECS224

Overview

This project is an attempt to make our own version of a question answering site like “StackOverflow” along with a few intelligent components as well.

Our frontend (written in node) and the backend (written in Django) are two separate servers where the front-end server sends API requests to the backend server.

Technologies Used

Front End

- Angular 7
- Angular Material
- TypeScript

Back End

- Django Framework
- Rest Framework
- Rest Auth
- Postgresql Database

Frontend Specifications

I. Components

The entire web app is divided into a number of Angular components. The root component <app-route> loads different child components according to the URL route.

The component structure is as follows:

```
app/
  feed/
    feed-item/
  nav/
    drawer/
    toolbar/
  login/
```

newquestion/
newanswer/
search/
qa/
  qa-content/
    qa-content-item/
  qa-similar/

II. Services

Services provide a way to share data between component classes and also allow creation of custom callbacks that can be called when data is changed.

Our app contains the following services:

- AuthService - Handles authentication and stores user login details
- RestService - Contains methods to query the REST API
- FeedService - Fetches and stores feed questions (featured and unanswered)
- QuestionService - Handles adding questions, answers and comments
- SearchService - Handles searching through questions in the database

Backend Specifications

https://github.com/ParasharaRamesh/IDK/

III. APIs Exposed

Question APIs

1. AddQuestion: add a question to DB
2. DeleteQuestion: delete a question from DB
3. GetQuestion: get a question and all its related answers and comments from DB
4. GetSimilarQuestions: given a question, it returns all the questions which are most similar to this question

Answer APIs

1. AddAnswer: add an answer to a question in the DB
2. DeleteAnswer: delete an answer from the DB
3. GetAnswer: get an answer and all its related comments from the DB

Comment APIs

1. AddComment: add a comment(with tagging users) to a question /answer in the DB
2. DeleteComment: delete a comment from the DB
3. GetComment: get a comment along with notified user(if any) from the DB
4. GetNotification: get a notification instance from the DB
5. DeleteNotification: delete a notification instance from the DB

User APIs

1. Login/Logout/Register: using django_rest_auth module for easy login/logout/register
2. GetUserDetails: get the user details and his related fields from the DB
3. Follow: follow another registered user
4. Unfollow: unfollow a followed user

Vote API

1. Vote: give an upvote or downvote to an question or answer

Search API

1. Search: given a search query (tags/normal text) search and show the most related questions

Feed API

1. Feed: for a specific user, show the “featured” and “unanswered” questions

* For detailed request, response JSON structure and datatypes structure refer:

https://github.com/ParasharaRamesh/IDK/blob/master/API-spec.md

IV. Database Table Initialization

We wrote a script to initialize the tables from a StackOverflow dataset available on kaggle. We added around 200 questions and 100 users in our database to simulate a working application.

V. Intelligence Components

We mainly have three intelligence components namely

(Integration has to be relooked at - there are some bugs)

1. **Automatic Tag Extractor:**

   Reference:

   https://github.com/E-tanok/NLTK_stackoverflow_tags_recommender

   Description:

   - This repository uses some advanced NLP techniques and is trained on existing StackOverflow data such that when given an input text it predicts programming tags related to it.
   - We use this tool to extract tags for every answer and question in our database which we then associate with a client user.
   - These stored tags are then used for implementing our “Search” and “Feed” APIs unique to every user.

2. **Offensive Language Detector:**

   Reference:

   https://github.com/adityagaydhani14/Toxic-Language-Detection-in-Online-Content

   Description:

   - This repository also uses some advanced NLP techniques and is trained on twitter data such that when given an input text it predicts whether it is “Offensive”, ”Hate” or a “Clean” text.
   - We use this tool for collapsing any piece of text typed by a user in the UI if it happens to be an “Offensive” or “Hate” message.

3. **QuestionSimilarity Checker:**

   Reference:

   https://github.com/rahul-1996/NLPService/blob/master/app/NLP/word_similarity.py

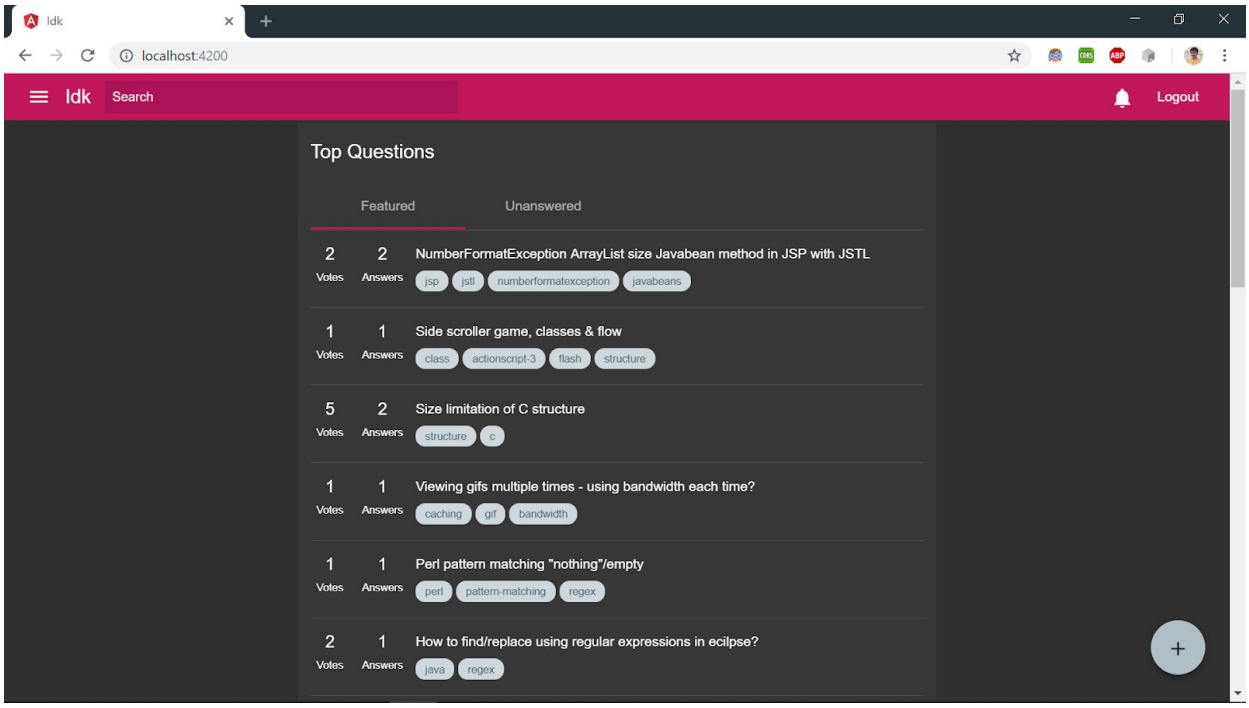
   Description:

   - We used a “wup_similarity” score to get the similarity between the two questions.

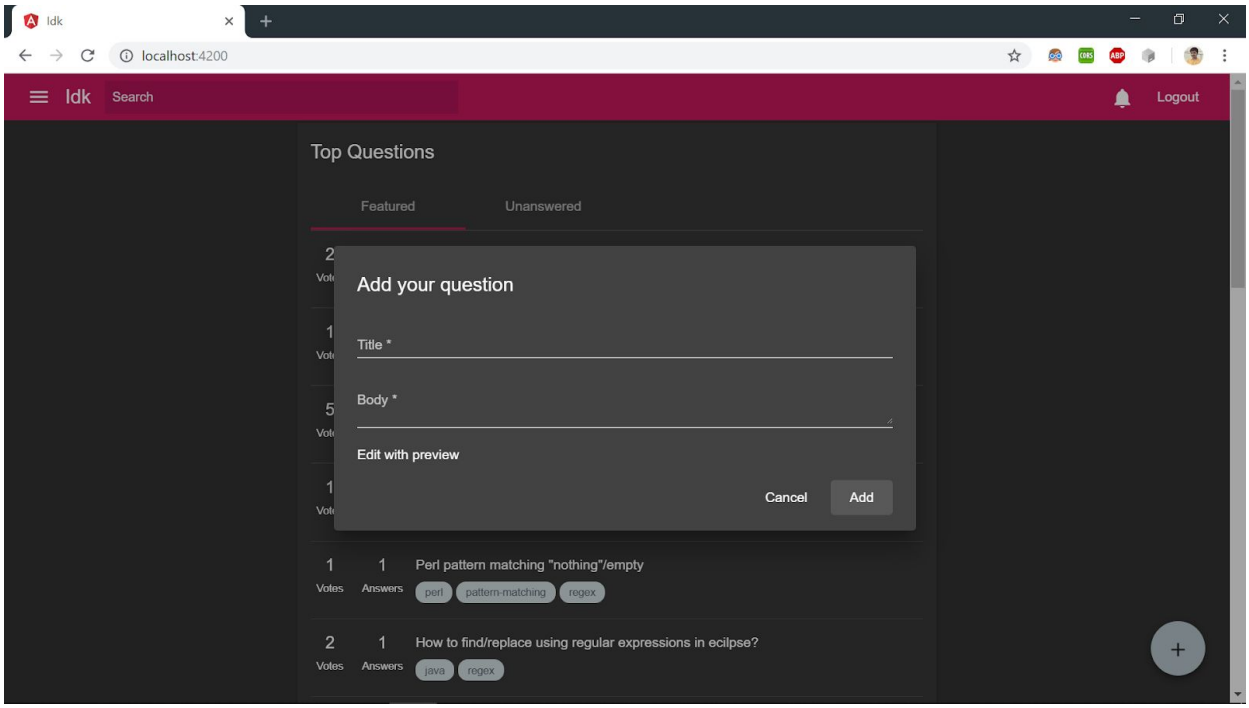
- We used this score to get all the similar questions for a particular question by sorting the top N questions based on the similarity score.
- Initially, we wanted to use a “Siamese LSTM” model but because of the slow predictions due to the large word embedding file, we decided to opt for a much simpler and faster similarity score.

Screenshots of the application

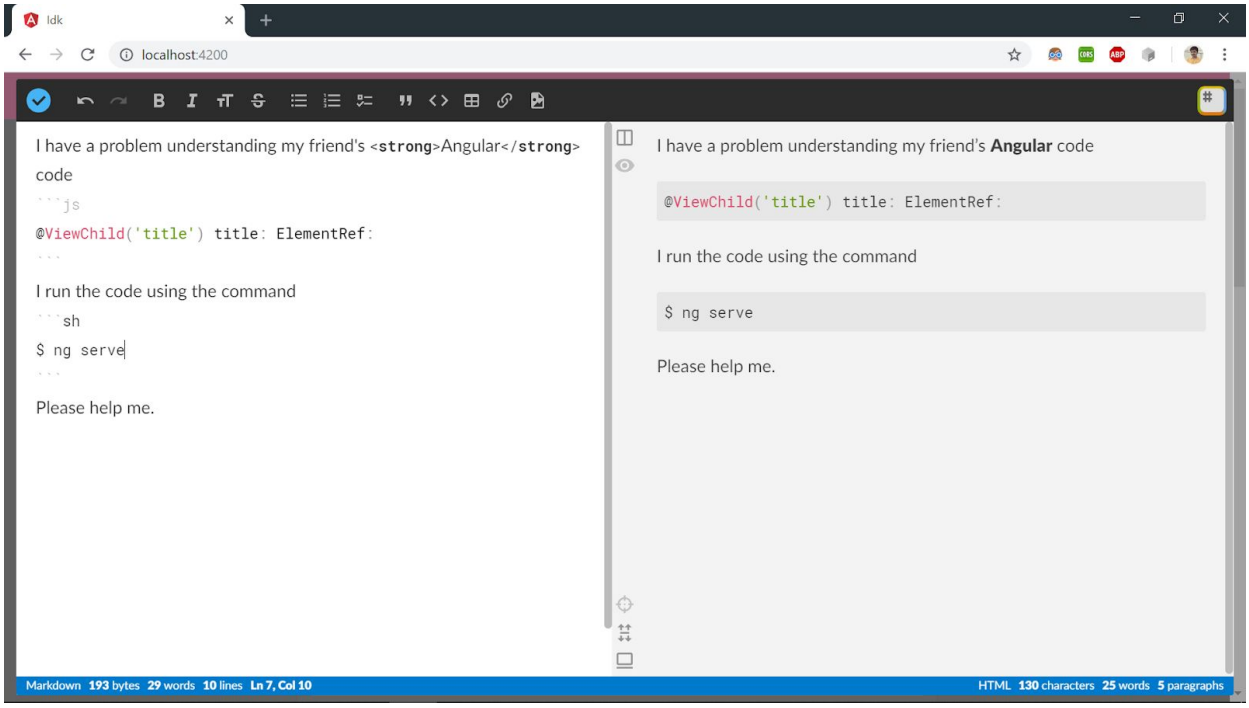
Index page after logging in



‘Add Question’ dialog



Question markdown editing with live preview



Question details page