Title: AcadOverflow

High Level Design

Abstract

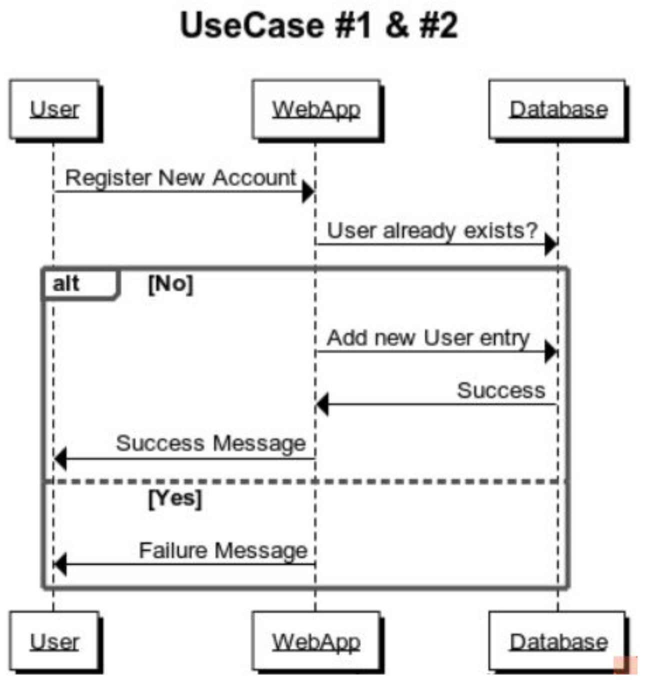
In today's world, people use the Internet to get answers to a wide range of questions. A person can ask a question being in one part of the world and get an answer from people present in other geographical locations. However, in our IIIT campus, we mostly confine our technical discussions and knowledge sharing to a small group of friends. Questions posted on moodle are mostly directed towards the TAs or the professor of the course.

As students we want to have a platform in the institute where technical questions can be posted/queried by any student, so that fellow peers can post responses to it or previously asked similar questions can be listed along with the answers.

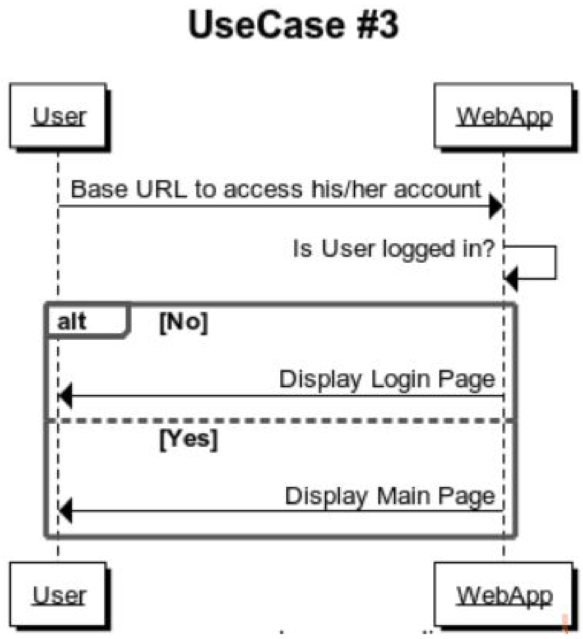
Use Cases

1. An account registered by a new user should be updated in the database.
   Given that a user wants to register a new account on the web app.
   and the user doesn't already have an account.
   when the user registers a new account
   then the total number of users accounts should increase by one and appropriate entries must be made in the database.

2. An account registered by an existing user, should result in an error message.
Given that a user wants to register a new account on the web app.
and the user already has an account.
when the user registers a new account
then an error message should be displayed to the user.



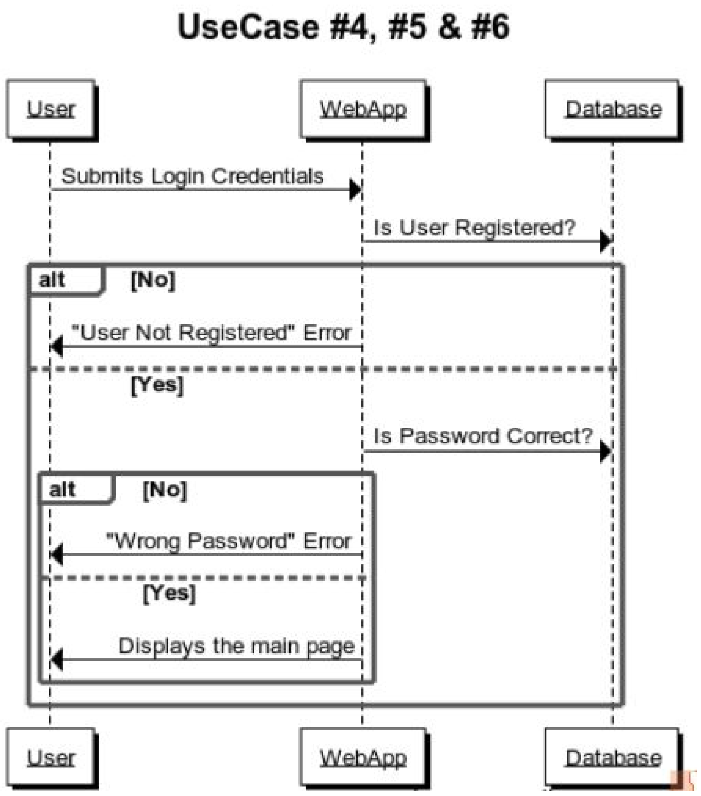
3. A logged out user should reach the login page from the base URL
Given that a user wants to access his/her account.
and the user is logged out of the account.
when the user enters the URL
then the login page should be displayed to the user.



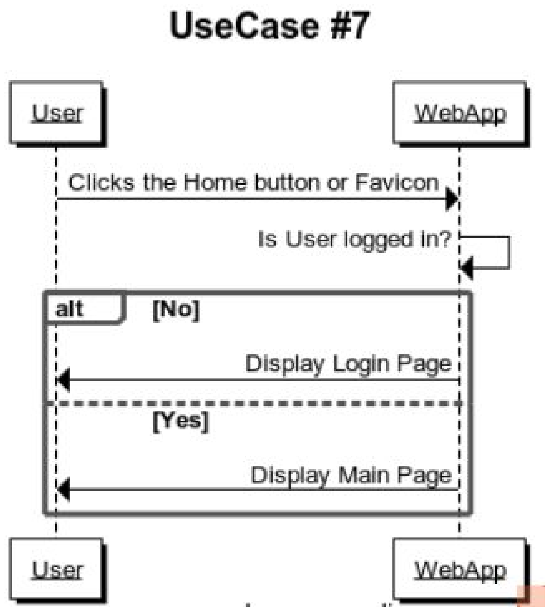
4. Unregistered username / EmaiL ID should get an error on login
   Given that a user wants to access his/her account.
   and the user has not registered an account.
   when the user submits the login credentials
   then an error message should be displayed to the user.

5. Registered username / EmaiL ID with wrong password should get an error on login
   Given that a registered user wants to access his/her account.
   and the user has entered a wrong password.
   when the user submits the login credentials.
   then an error message should be displayed to the user.

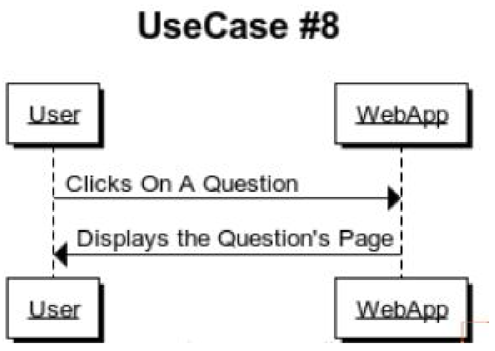
6. Successfully logged in users should be redirected to the main page.
   Given that a registered user wants to access his/her account.
   and the user has entered the correct password.
   when the user submits the login credentials
   then the user should be redirected to the main page.



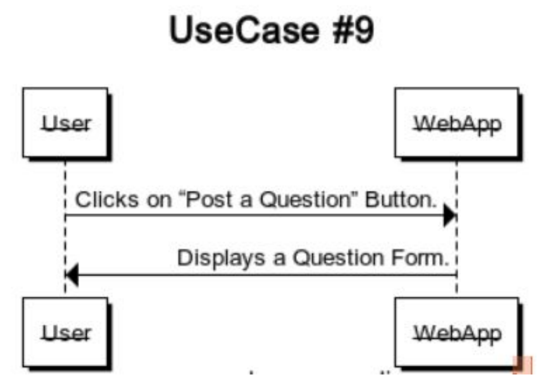
7. <u>A logged in user should reach the main page from the base URL or Home button.</u>
   Given that a user is logged in
   and there is a home button on the page
   when the user presses the Home button
   then the user should be redirected to the main page.



8. <u>A clicked question should redirect to that question's page</u>
   Given that the questions are listed as per the user's query history
   and displayed in descending order of vote count.
   when the user clicks on a question.
   then the user should be redirected to that question's page.

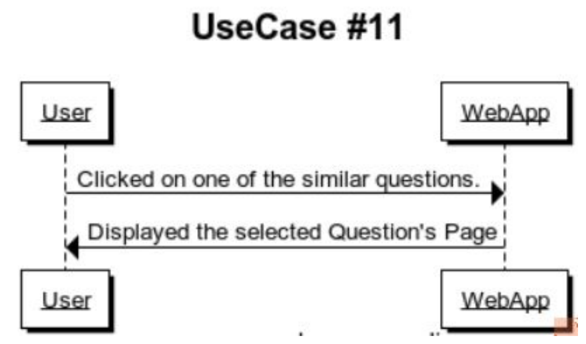


9. “Post a Question” button clicked should display the question form.
Given that a user wants to post a question
and has successfully logged in
when the user clicks on the “Post a Question” button.
then the user should be provided a question form.

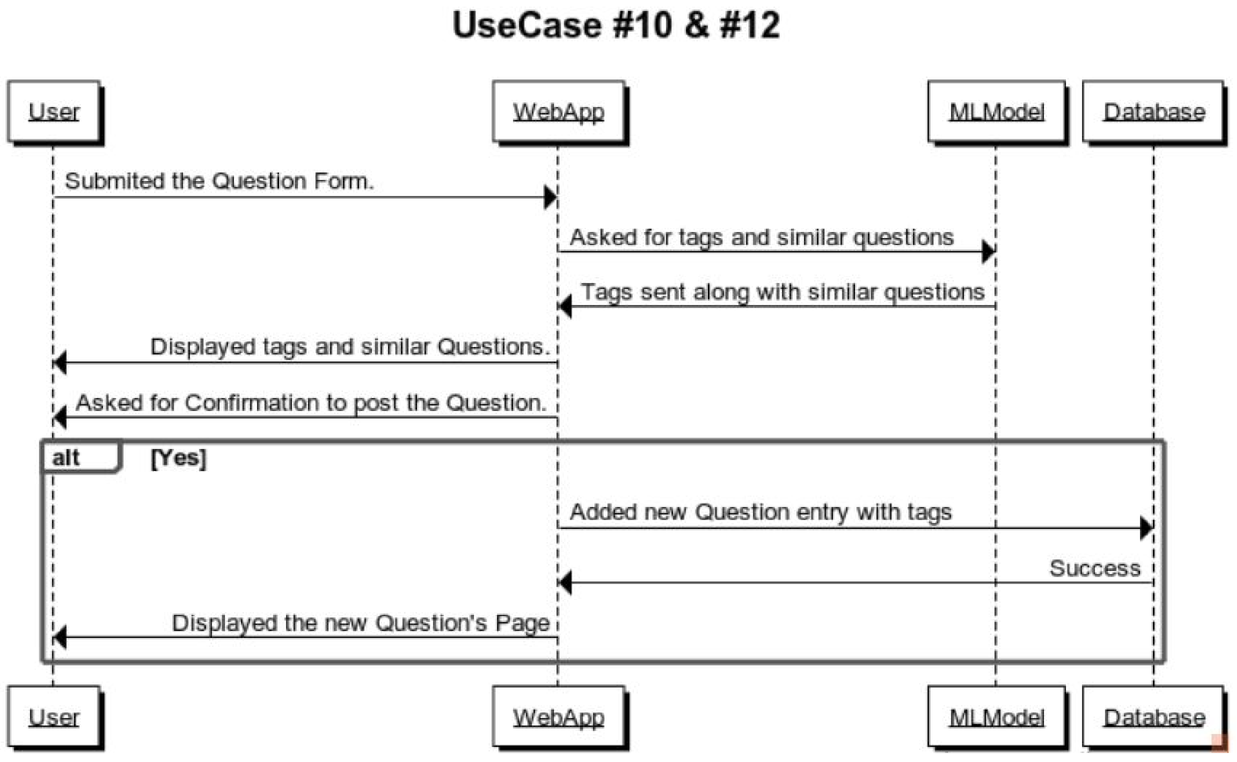

UseCase #9
User
WebApp
Clicks on “Post a Question” Button.
Displays a Question Form.
User
WebApp

10. Question Form submitted should result in tags assigned to the question
Given that the user has filled the question form.
and the ML model has been trained
when the user submits the form.
then tags should be assigned to the new question and displayed along with it.
The most similar questions are also displayed. Two options are given to the user, either to go to one of the similar questions if he finds one for which he is querying for, otherwise post the question.

11. User found the question he wanted to ask, so that question’s page is displayed to him.
Given the user found the question which he was querying for in the similar questions list
and he wants to go to that question’s page.
then the user should be redirected to that question’s page.

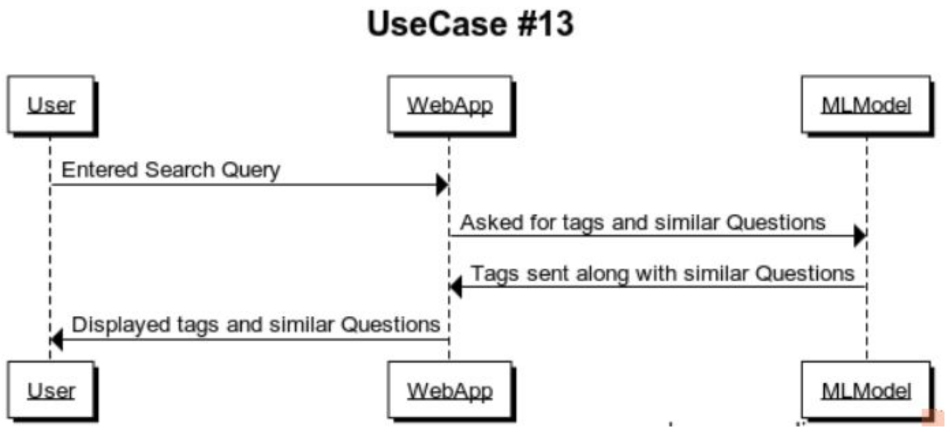

UseCase #11
User
WebApp
Clicked on one of the similar questions.
Displayed the selected Question's Page
User
WebApp

12. <u>User not able to find the question he wanted to ask, so he actually posts the question.</u>
Given the user did not find the question which he was querying for in the similar questions list
and he wants to actually post the question which he mentioned in the questions' form.
when the user confirmed to post the question
then should be redirected to that question's page.



UseCase #10 & #12
User
WebApp
MLModel
Database
Submited the Question Form.
Asked for tags and similar questions
Tags sent along with similar questions
Displayed tags and similar Questions.
Asked for Confirmation to post the Question.
alt
[Yes]
Added new Question entry with tags
Success
Displayed the new Question's Page

13. <u>Search query submitted should result in similar questions being listed</u>
Given that the user has successfully logged in
and entered the question in the search box
when the user clicks the search button.
then similar questions should be displayed.

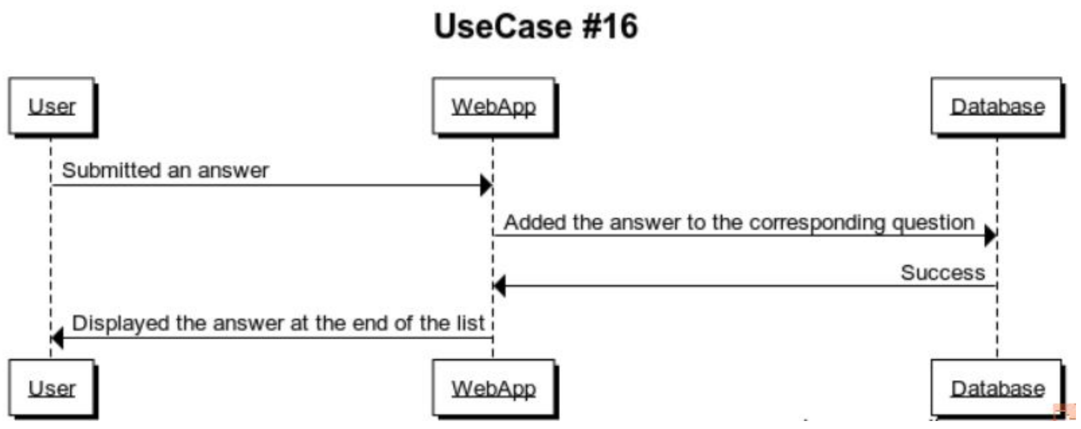


UseCase #13
User
WebApp
MLModel
Entered Search Query
Asked for tags and similar Questions
Tags sent along with similar Questions
Displayed tags and similar Questions

14. <u>Upvote / Downvote buttons result in incrementing / decrementing the vote count of a particular question</u>
    Given that a user wants to upvote / downvote a question
    and the vote count has been initialized with 0 in the database.
    when the user clicks upvote / downvote button
    then the vote count is updated accordingly on the screen and the database.

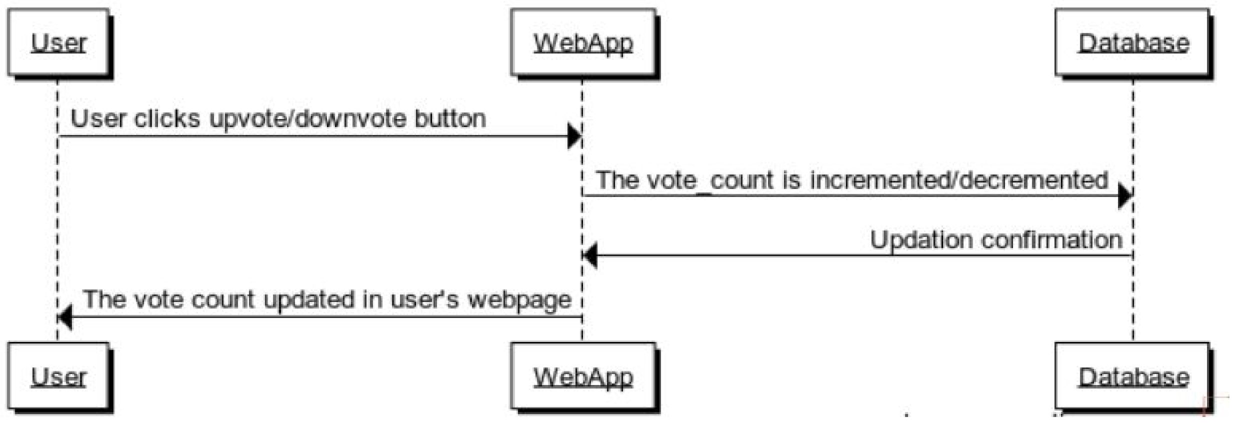
15. <u>Similar questions should be displayed in decreasing order of vote count</u>
    Given the user has entered the question to be searched or posted
    and vote count is maintained for each question in the database
    when the user clicks the “search” or “post question” button
    then the questions should be displayed in the decreasing order of vote count

16. <u>Submitted answer should be displayed at the end of Answer’s List</u>
    Given a question
    and an answer box on the Question’s page
    when user submits an answer
    then the submitted answer is displayed at the end with vote count as 0.

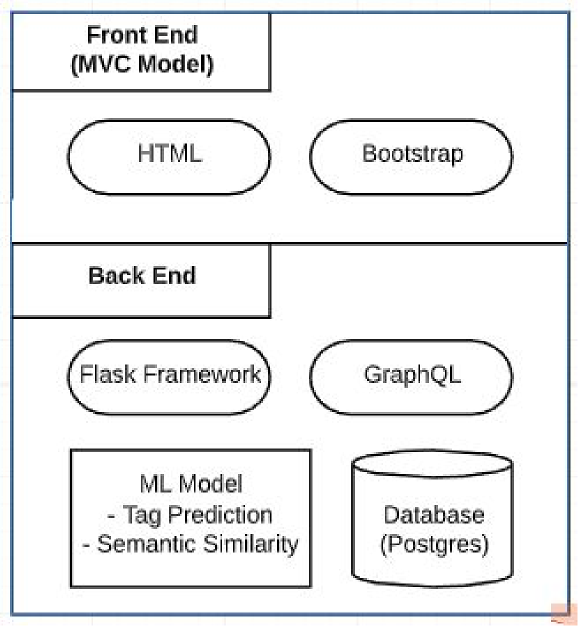

UseCase #16
User
WebApp
Database
Submitted an answer
Added the answer to the corresponding question
Success
Displayed the answer at the end of the list
User
WebApp
Database

17. <u>Clicking Upvote / Downvote buttons resulted in incrementing / decrementing the vote count of a particular answer</u>
    Given that a user wants to upvote / downvote an answer
    and the vote count has been initialized with 0 in the database.
    when the user clicked upvote / downvote button
    then the vote count is updated accordingly on the screen and the database.

UseCase #14 & #17

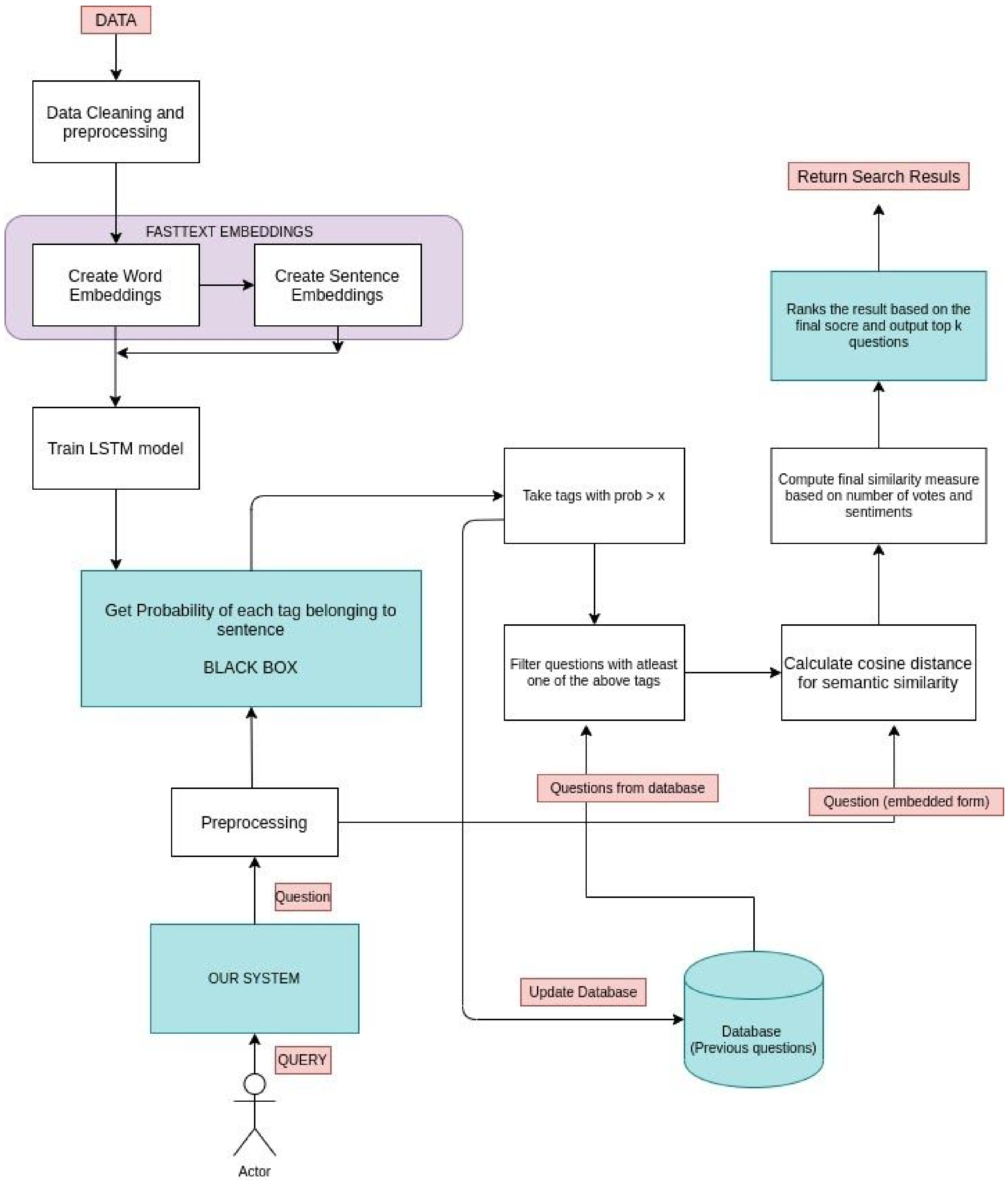

User
WebApp
Database
User clicks upvote/downvote button
The vote_count is incremented/decremented
Updation confirmation
The vote count updated in user's webpage
User
WebApp
Database

Web Application Architecture



- In **Model View Controller (MVC**) architecture, the server sends HTML pages to the browser, with the relevant data inside them. The pages are created on the server using template engines.
- **Bootstrap** is an open source toolkit for developing with HTML, CSS, and JS. It comprises a responsive grid system, extensive pre-built components, and plugins built on jQuery.
- **Flask** is a microframework" for Python, and is an excellent choice for building smaller applications, APIs, and web services.
- **GraphQL** allows smaller objects in the database transactions. It allows fetching different parts of different objects from the server in one call, and making calls leaner, which is better for possible smaller bandwidth that might occur if the internet is bad.
- Relational databases have tables and their schema is predefined. **PostgreSQL** is a popular choice in such cases.

Machine Learning Architecture


DATA
Data Cleaning and preprocessing
FASTTEXT EMBEDDINGS
Create Word Embeddings
Create Sentence Embeddings
Train LSTM model
Get Probability of each tag belonging to sentence
BLACK BOX
Preprocessing
Question
OUR SYSTEM
QUERY
Actor
Take tags with prob > x
Filter questions with atleast one of the above tags
Questions from database
Update Database
Database (Previous questions)
Calculate cosine distance for semantic similarity
Question (embedded form)
Compute final similarity measure based on number of votes and sentiments
Ranks the result based on the final socre and output top k questions
Return Search Resuls

We will collect Data of questions and answers from Stack Overflow. **Google BigQuery** dataset includes an archive of Stack Overflow content, including posts, votes, tags, and badge. This data will be a tuple of **{id, title, ques, tags, answers, votes}**.
Data Preprocessing involves cleaning the text in the following ways -

- Tokenization
- Conversion to Lowercase
- Remove Stop Words and punctuation
- Dealing with HTML, URLs, abbreviations, etc.

We will use nltk library for text cleaning and preprocessing. We extract the top **500** tags based on their occurrences in the dataset. In order for our model to understand the raw text data, we need to vectorize it for that we are using **fasttext** embeddings and then will train a Tag classifier using the **LSTM model**.
"Given the user query, we'd like to predict which tags the given query best belongs to"

Since we are dealing with a multilabel classification problem here, we cannot train the model using a binary cross-entropy loss. This is because binary cross-entropy loss would push our model to predict one or two tags which are included in the ground truth, and won't penalize it for missing out on the other ones. Thus **we are using log loss on each class separately** and then compute its average based on the number of classes.

After this model is ready, given any question we can predict the probabilities of each tag belonging to this question.
Finally, we will filter out top tags with probability greater than **threshold** value and hence this will reduce our set of questions which could be similar to this question.
Finally we will get the semantic similarity with the questions in this set by measuring **cosine distance** between the questions (i.e. between encodings of questions). For final similarity we are using following formula -

$$Similarity(q,\ Qi)\ =\ Cosinedistance(q,\ Qi)\ +\ 0.1\ *\ Normalized\ Vote\ Score(Qi)\ +\ 0.4\ *\ Sentiments(Qi)$$

where q = user query and Qi = existing question
Hence it consider -

- Cosine Distance as a base measure.
- Popularity of questions based on votes.
- Responses people have given to the existing question.

Finally, we will get all the related questions.

Team Members (Team #20)

Bhavi Dhingra
Aditi Shrivastava
Nikita Rungta
Surbhi Sharma
Samyak Agrawal
Kajal Mohan Sanklecha