ProbExpert - A novel learning aid platform for reducing time spent on learning and finding answers.

2021-155

Prob Expert

Team

Supervisor
Ms. Dinuka Wijendra

Co-Supervisor
Ms. Anjalie Gamage

Kamal Thennakoon
IT18004564

Dasun Ekanayake
IT1801360

Ashen Ranasinghe
IT18011012

Thanura Marapana
IT18078992

Introduction

- Focused on creating a better learning aid platform to the learners and contributors.
- One to one learning
- give newbies a solid idea of how they should shape their path
- Providing a platform to users regardless of their knowledge degree to interconnect with each other.

System overview

- Portfolio generation
- Automatic answer generation from public information
- Structured-type quizzes for knowledge checking
- Optimized answers
- Each user can earn ProbExpert coins through their contribution to the platform.

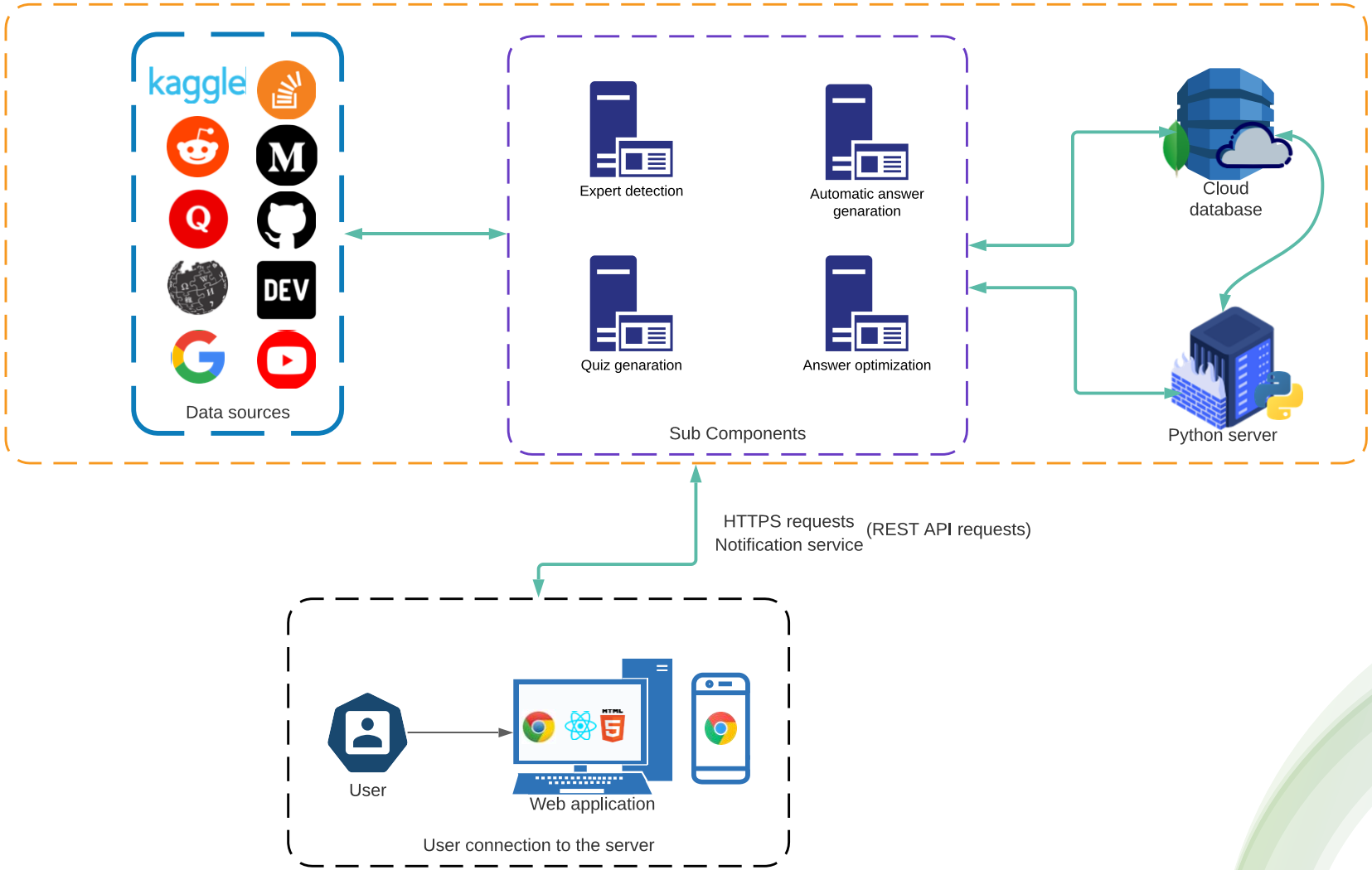
What is one to one learning in ProbExpert?

- Find a mentor for your requirement.
- Schedule a video call using ProbExpert coins.
- At the end, mentor will receive ProbExpert coins from the mentee.

Objectives

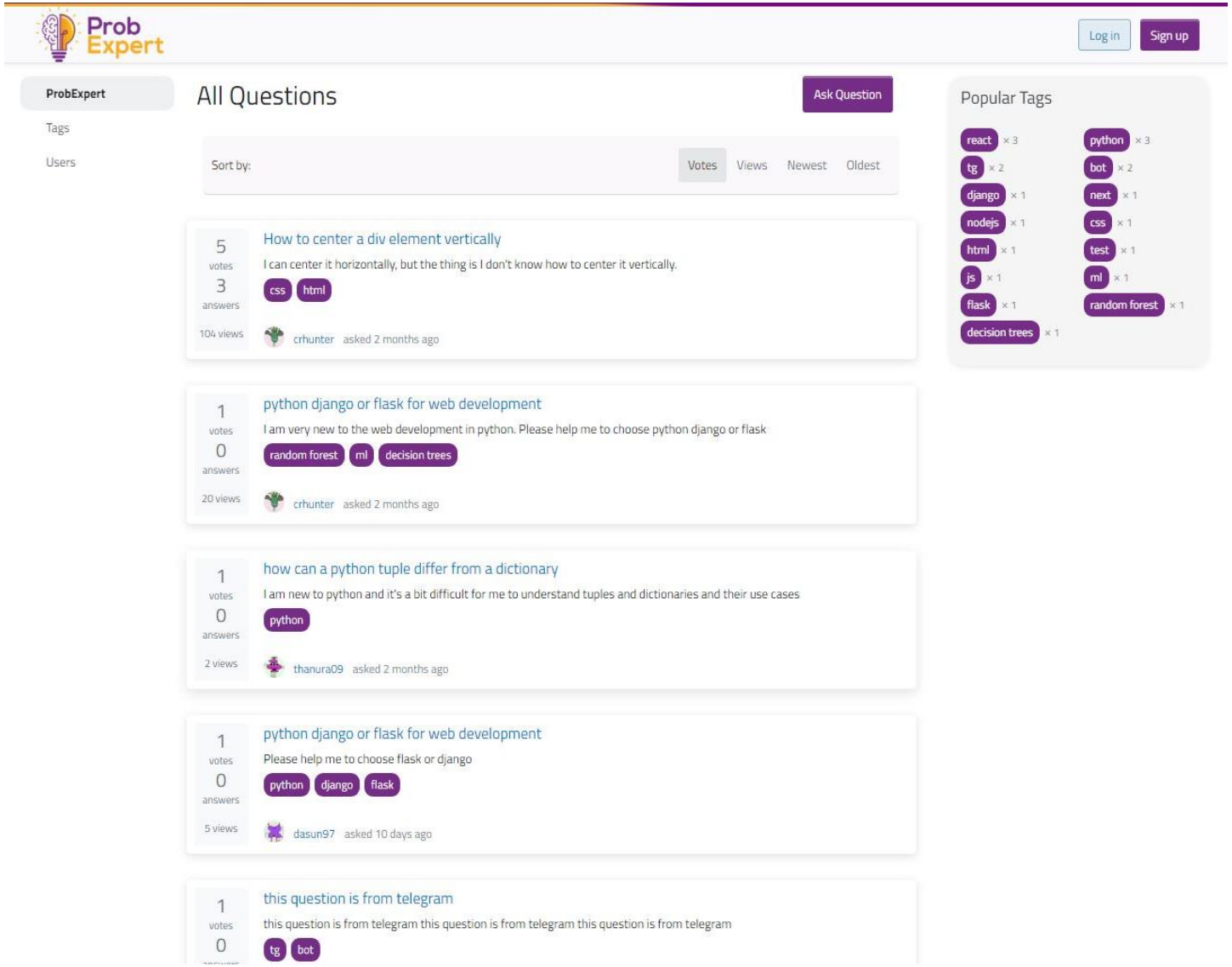
- Detecting users' proficiency level along with an auto-generated portfolio.
- Form an answer automatically using public data from multiple sources for user’s questions.
- Optimal answer generation through up-voted answers.
- Formulating structured type questions for knowledge checking, using existing answered questions of the platform

Overall System Diagram

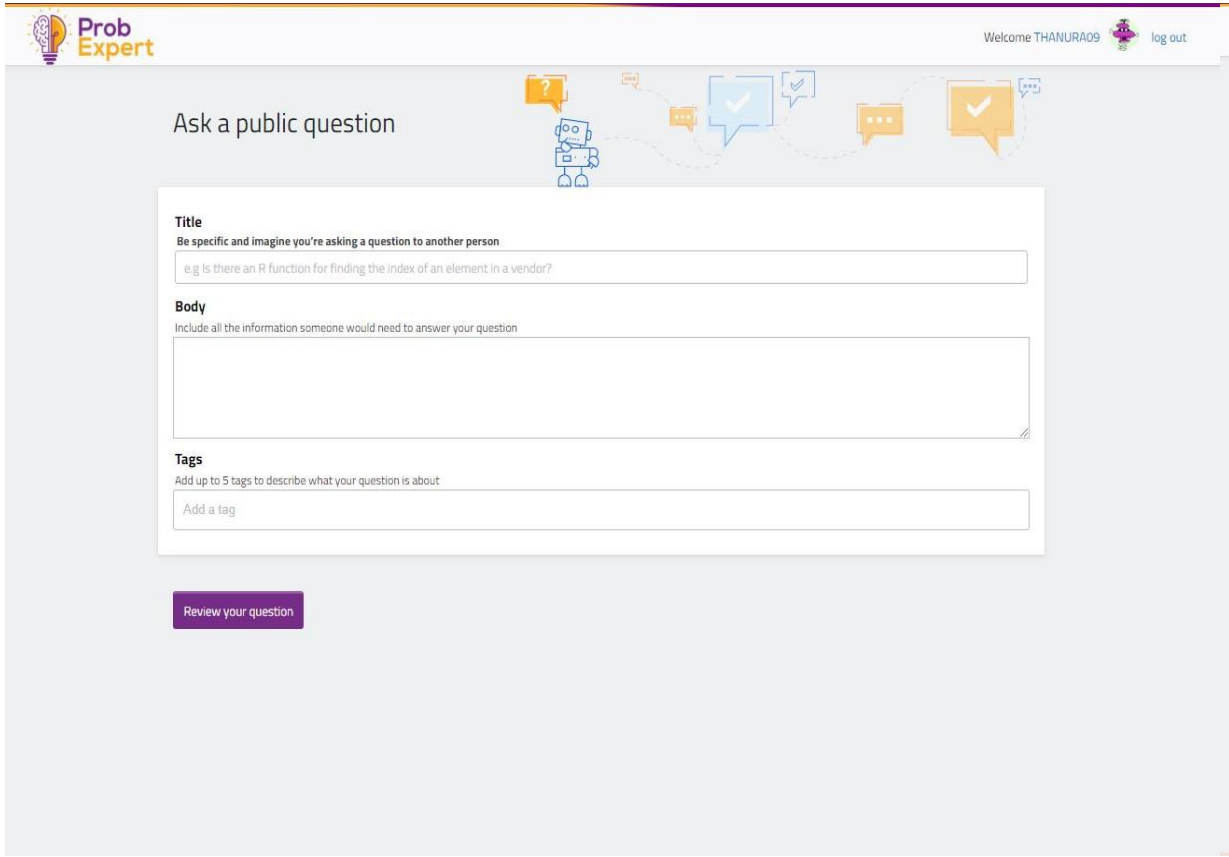


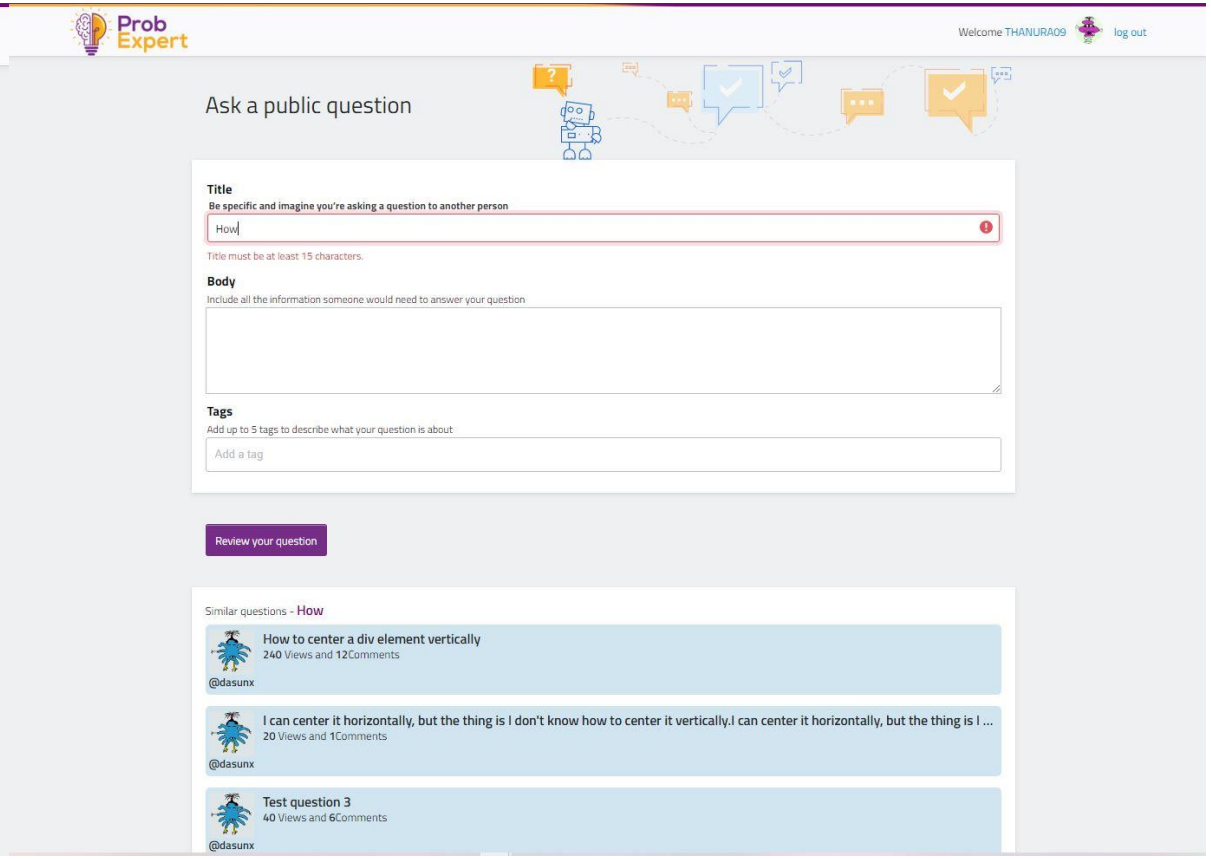
User interface breakdown

Home page

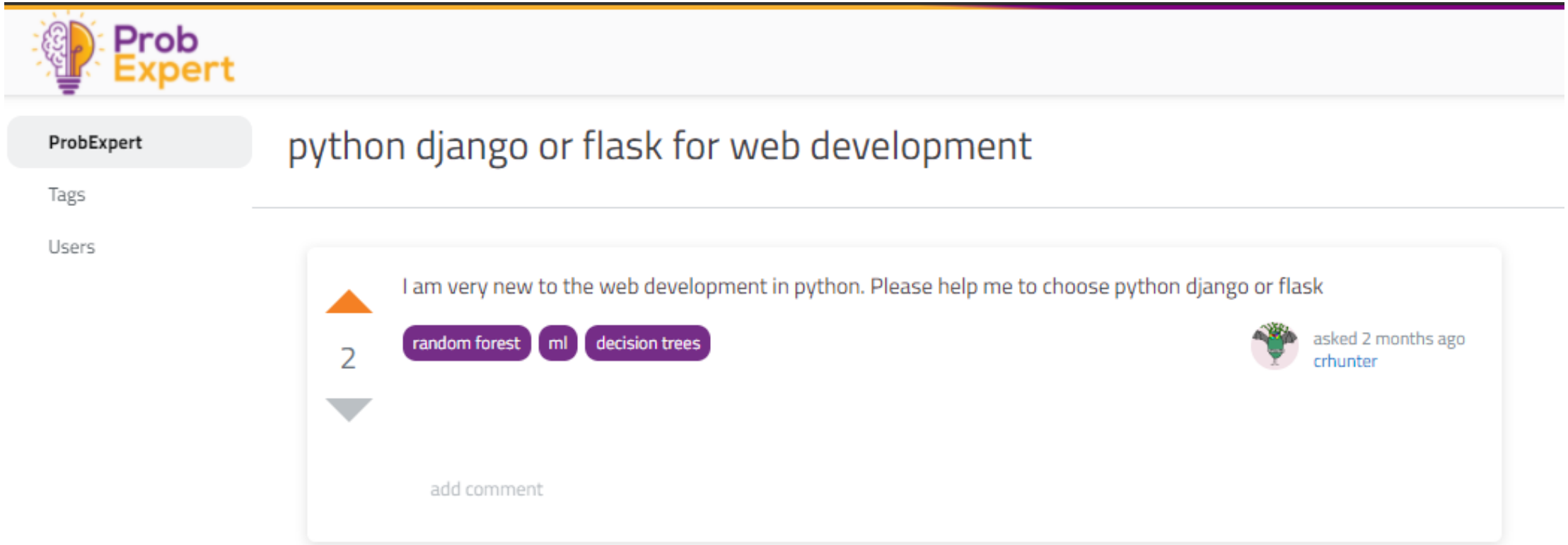

Prob Expert
Log in
Sign up
ProbExpert
Tags
Users
All Questions
Ask Question
Sort by:
Votes
Views
Newest
Oldest
5 votes 3 answers 104 views
How to center a div element vertically
I can center it horizontally, but the thing is I don't know how to center it vertically.
css
html
crhunter asked 2 months ago
1 votes 0 answers 20 views
python django or flask for web development
I am very new to the web development in python. Please help me to choose python django or flask
random forest
ml
decision trees
crhunter asked 2 months ago
1 votes 0 answers 2 views
how can a python tuple differ from a dictionary
I am new to python and it's a bit difficult for me to understand tuples and dictionaries and their use cases
python
thanura09 asked 2 months ago
1 votes 0 answers 5 views
python django or flask for web development
Please help me to choose flask or django
python
django
flask
dasun97 asked 10 days ago
1 votes 0
this question is from telegram
this question is from telegram this question is from telegram this question is from telegram
tg
bot
Popular Tags
react × 3
python × 3
tg × 2
bot × 2
django × 1
next × 1
nodejs × 1
css × 1
html × 1
test × 1
js × 1
ml × 1
flask × 1
random forest × 1
decision trees × 1

Ask a Question and Similar suggestions

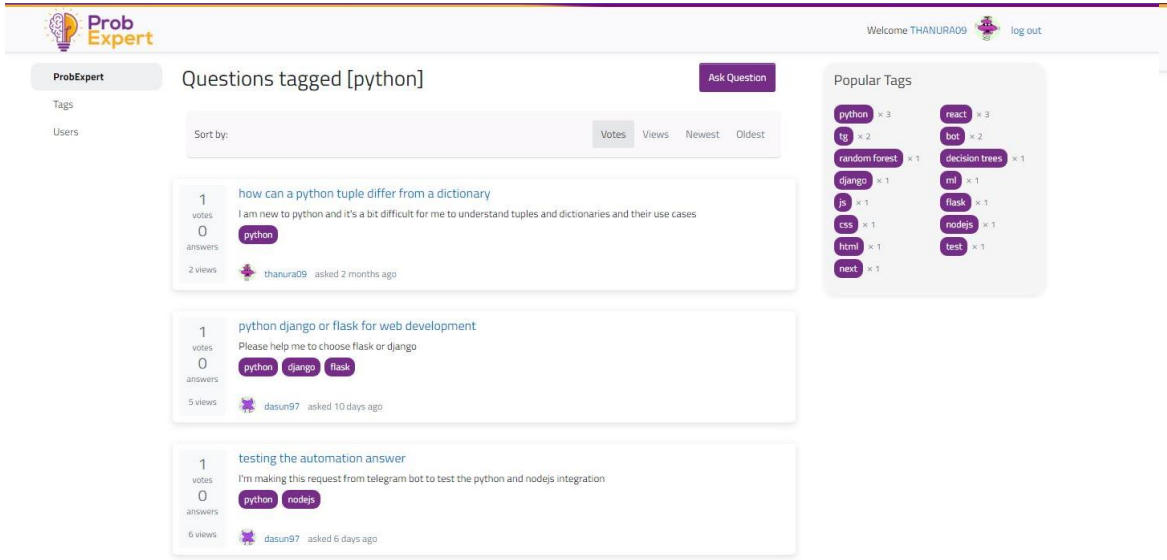


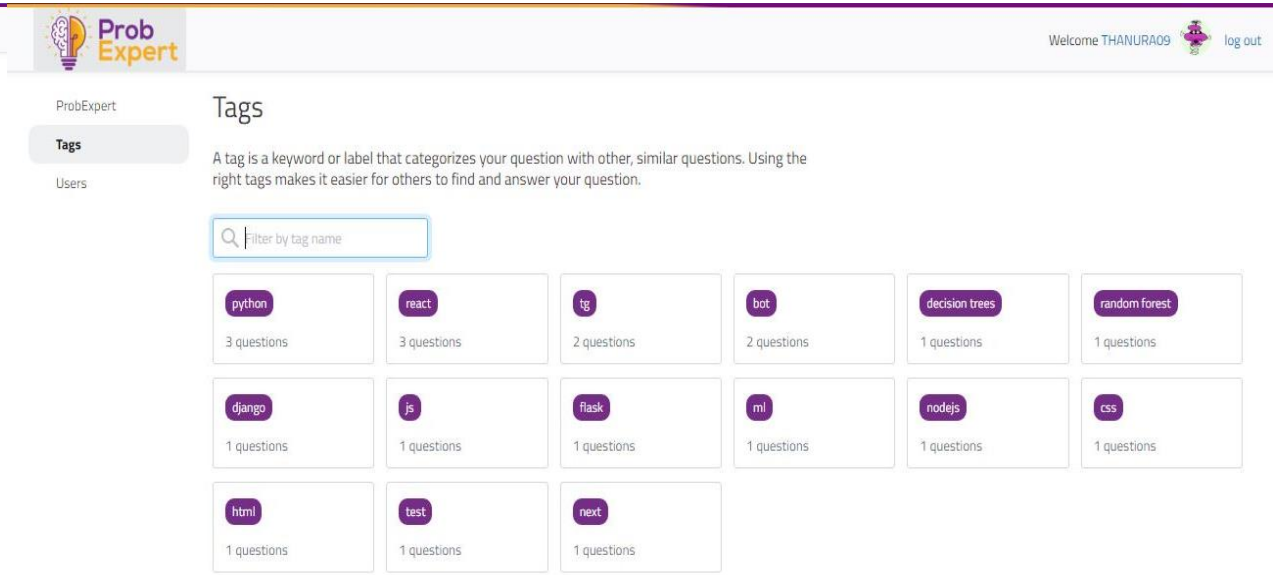


Upvoting

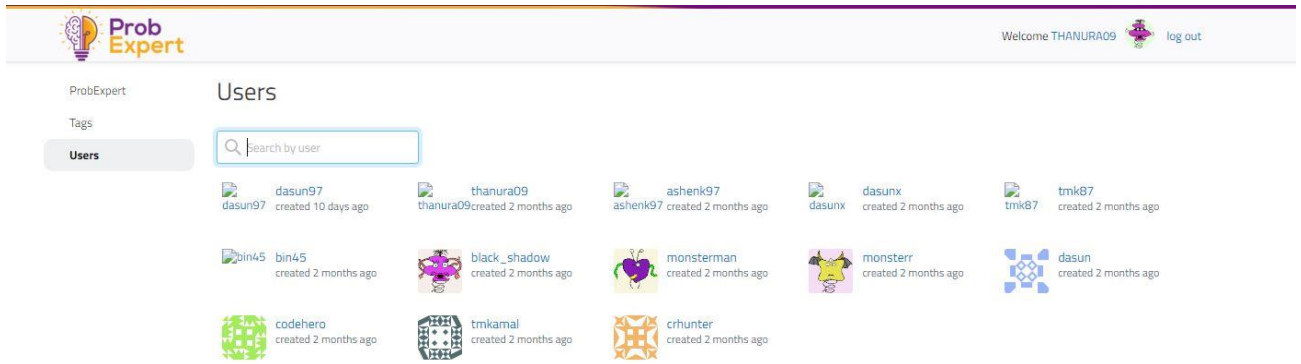

Prob Expert
ProbExpert
Tags
Users
python django or flask for web development
I am very new to the web development in python. Please help me to choose python django or flask
2
random forest
ml
decision trees
asked 2 months ago
crhunter
add comment

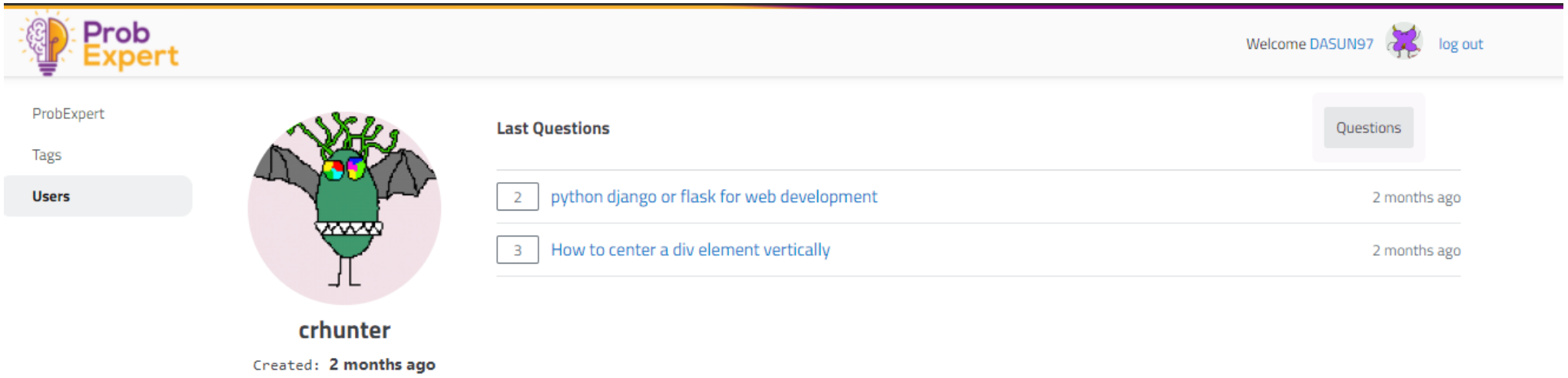
Question tags





Users and User profiles





IT18004564| Thennakoon T.M.K.H.B

B.Sc. (Hons) Degree in Information Technology Specialized in Software Engineering

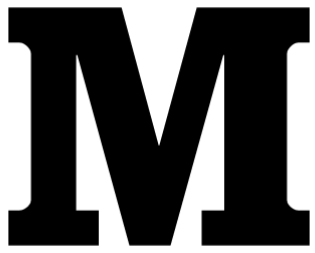
Detecting users' proficiency level along with an auto-generated portfolio

Background/ Research Gap

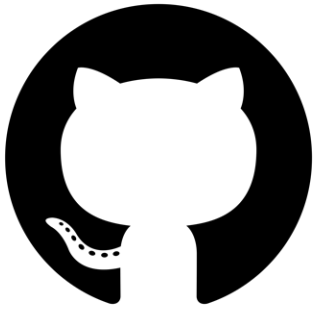
Prevailing methods for detect experts in general social platforms.

- Page Rank
- HITS
- Mutual Reinforcement approach
- Hybrid approach of Wen-Chen
- Topic sensitive probabilistic model

Background/Research Gap



follower count



follower count

No specific evaluation method

Reputation based

- Most of developer's social platforms don't have a method to evaluate their users.
- Platforms like Stack Overflow uses reputation-based system.

Research Gap

Platform	User-level	Publications	Open-Source contributions	Detailed Bio	Blogs
Stack Overflow	Based on the platform Karma	None	None	Medium	None
LinkedIn	Based on endorsements	manually	None	Good	None
Medium	None	None	None	Low	Platform Dependent
GitHub	None	None	Available	Low	None
ProbExpert	Based on all the platforms activities and behaviors	Available	Available	Good	Platform Independent

Research Problem

- No way to find an expert
- Candidates exaggerate their skills in cv
- No platform to showcase proficiency and the contributions to the dev communities.
- Recruiters still don’t have a method to evaluate their candidates in a single place.
- No methods to find a skilled unemployed developers in a short time.
- Newbies don’t have a proper way to follow an expert of their field as a role model, to shape their career paths.

Do you Know?

- 53% of resumes and job applications contain Falsified information.
- 85% of employers caught job candidates lying on their resumes.

Source: CNBC

Specific and Sub Objectives

- **Detecting users' proficiency level along with an auto-generated portfolio**
    - Web Crawlers to scrape user data.
    - Serverless RESTful API to retrieve data from GitHub Graphql service.
    - Scoring model to generate a score for developers' activities and behaviors.
    - A Ranking algorithm to classify developers into suitable classes.
    - Generating a Portfolio based on the generated results and the scraped data.

Research Methodology

Methodology

Data Collection

- **GraphQl** queries used for collect user's GitHub statistics.

- Since other platforms do not provide public APIs to collect data, We proposed an advanced web crawlers to scrape public user data.

  Technologies: Python, Selenium, BeautifulSoup4

Methodology

Data visualization and feature extraction

- Balanced dataset based on Stratified sampling
- Consisted with 6 various strata.

- Separate Jupyter notebook for Feature extration.

  Technologies: Matplotlib, Pandas, Seaborn

Methodology

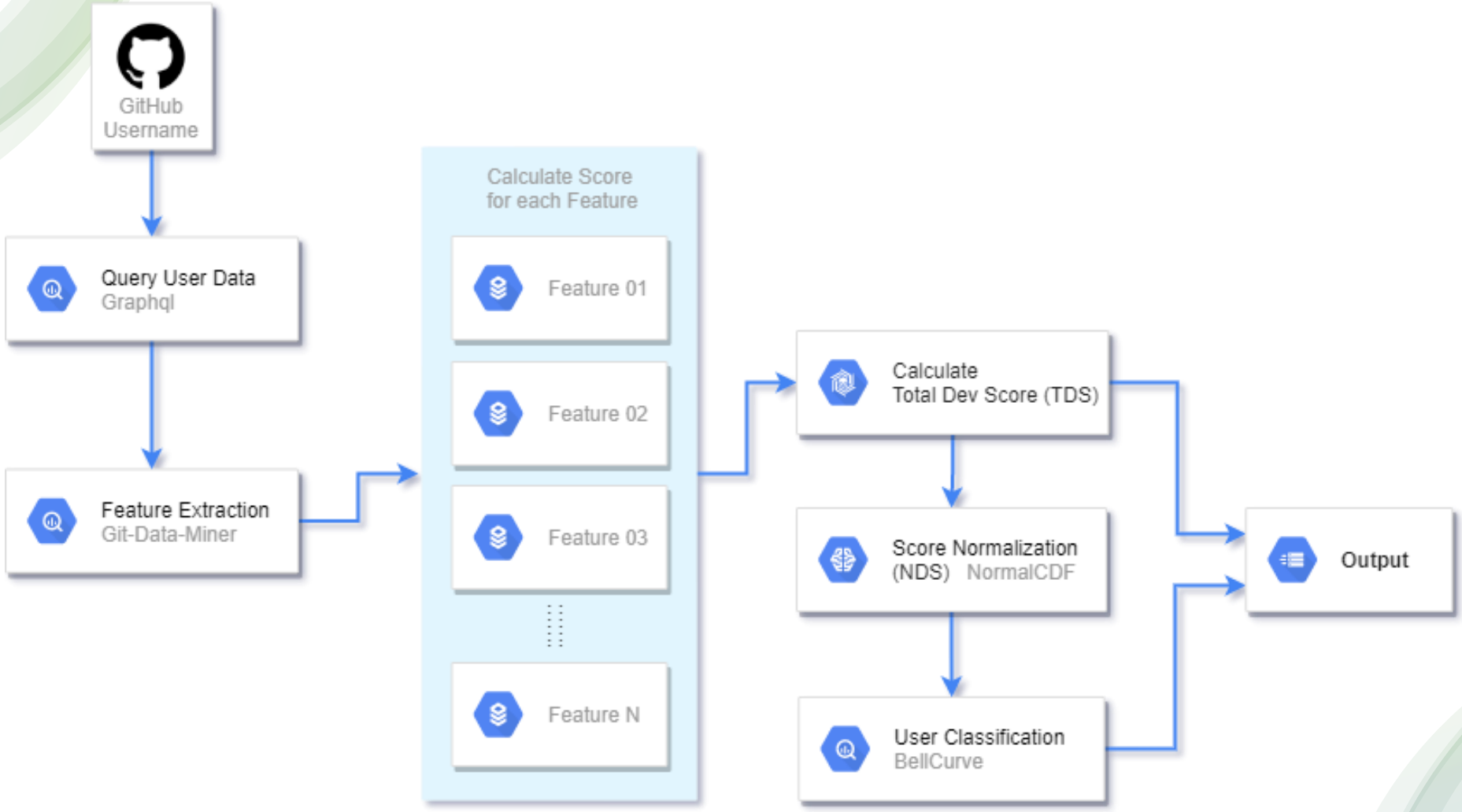
User Ranking Method

- Two separate models calculate users scores.

    1. Social Coding platform data (GitHub)
    2. Q&A platform data (Stack Overflow)

- Ranking algorithm : Bell Curve, NormalCDF

GitHub Scorer Model

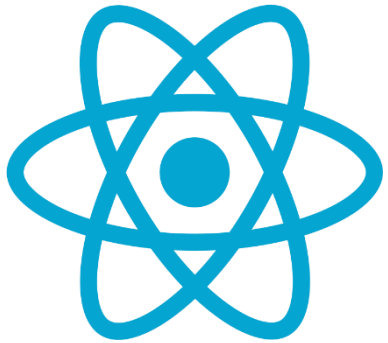


Methodology

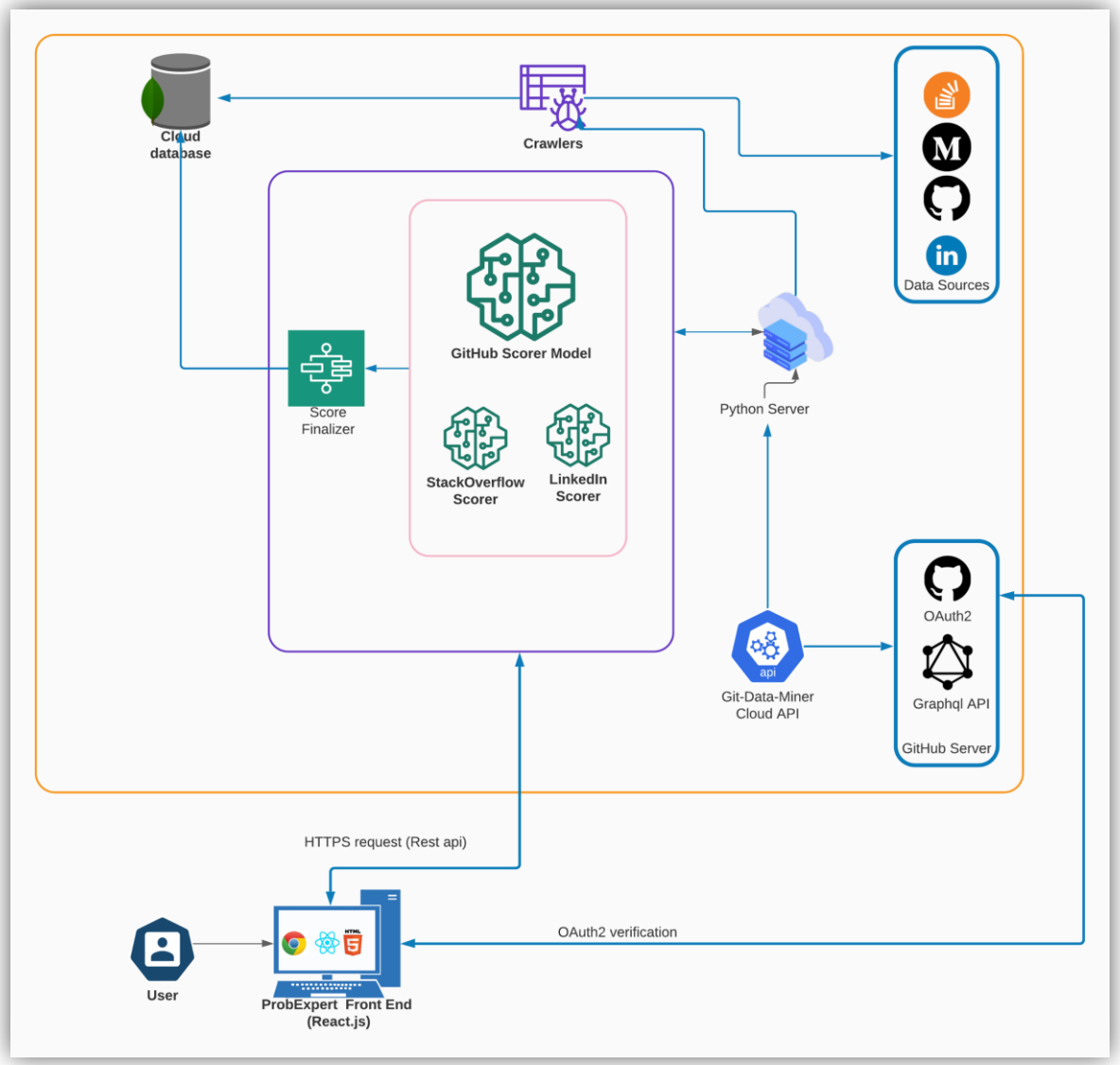
Portfolio

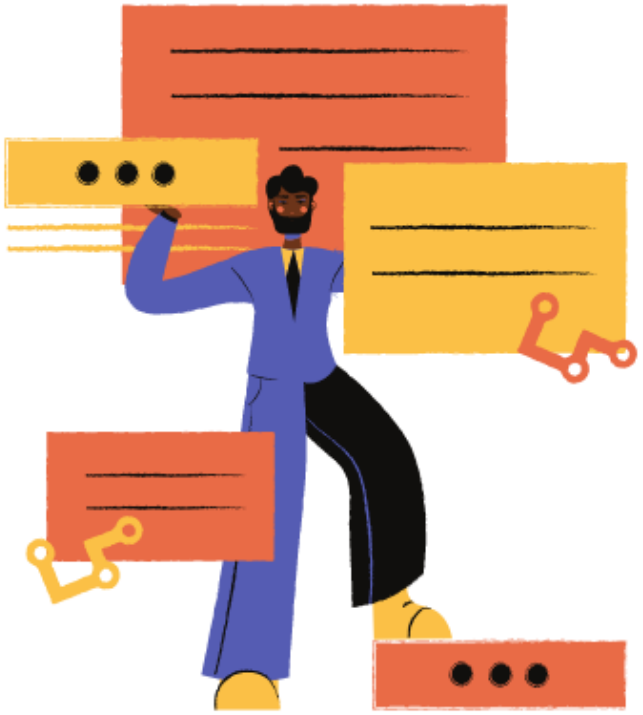
- Cloud Functions used to query and process GitHub data.
- Each Integration as a Micro service (GitHub, Stack Overflow, Medium)
- Next.js Static optimization used to generate server-rendered & static portfolios (Statically optimized)
- No server-side computations, ultra fast loading experiences.

Technologies: React.js, Next.js



System Diagram



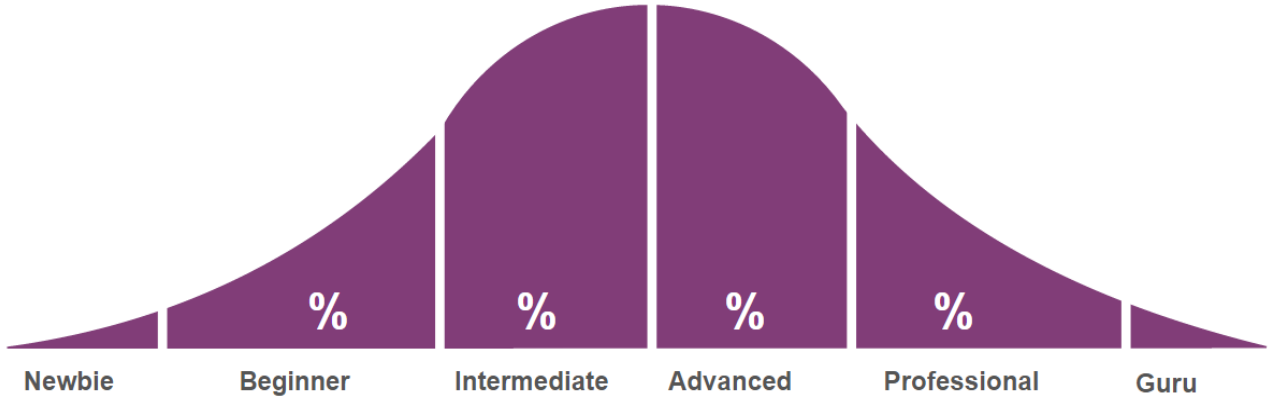


Evidences For Completion

Implementation

Ranking Algorithm (GitHub)

- Bell Curve used to build the ranking algorithm.
- Users divided into 6 classes.



Bell Curve

Implementation

```
const COMMITS_WEIGHT = 1.65;
const CONTRIBS_WEIGHT = 1.65;
const ISSUES_WEIGHT = 1;
const STARS_WEIGHT = 0.75;
const PRS_WEIGHT = 0.5;
const FOLLOWERS_WEIGHT = 0.45;
const REPO_WEIGHT = 1;

const ALL_WEIGHTS =
  CONTRIBS_WEIGHT +
  ISSUES_WEIGHT +
  STARS_WEIGHT +
  PRS_WEIGHT +
  FOLLOWERS_WEIGHT +
  REPO_WEIGHT;

const LVL_GURU = 1;
const LVL_PROFESSIONAL = 25;
const LVL_ADVANCED = 45;
const LVL_INTERMEDIATE = 55;
const LVL_BEGINNER = 80;
const LVL_NEWBIE = 100

const TOTAL_VALUES =
  LVL_GURU + LVL_PROFESSIONAL + LVL_ADVANCED + LVL_INTERMEDIATE +  LVL_BEGINNER + LVL_NEWBIE - 100;

const rewardPoints = (
  totalCommits * COMMITS_WEIGHT +
  contributions * CONTRIBS_WEIGHT +
  issues * ISSUES_WEIGHT +
  stargazers * STARS_WEIGHT +
  totalPRs * PRS_WEIGHT +
  followers * FOLLOWERS_WEIGHT +
  totalRepos * REPO_WEIGHT
) / 100;
```

Ranking Algorithm (GitHub)

Implementation

```
const todayDate=new moment(Date.now());
const joinedDate=new moment(createdAt);
const profileAge=moment.duration(todayDate.diff(joinedDate)).asDays();
let penaltyPoints=0;

// penalty calculation
if(totalCommits<1000){
  penaltyPoints= (totalCommits==0?(-1000):( -1000/totalCommits))+penaltyPoints;
}

if(totalPRs<15){
  penaltyPoints=(totalPRs==0?(-150): (-1500/(totalPRs*100)))+penaltyPoints;
}

if(profileAge<365){
  penaltyPoints= (profileAge==0?(-365):( -365/profileAge))+penaltyPoints;
}


const totalScore=rewardPoints+penaltyPoints;

const normalizedScore = normalcdf(totalScore, TOTAL_VALUES, ALL_WEIGHTS) * 100;

function normalcdf(mean, sigma, to) {
  var z = (to - mean) / Math.sqrt(2 * sigma * sigma);
  var t = 1 / (1 + 0.3275911 * Math.abs(z));
  var a1 = 0.254829592;
  var a2 = -0.284496736;
  var a3 = 1.421413741;
  var a4 = -1.453152027;
  var a5 = 1.061405429;
  var erf =
    1 - ((((a5 * t + a4) * t + a3) * t + a2) * t + a1) * t * Math.exp(-z * z);
  var sign = 1;
  if (z < 0) {
    sign = -1;
  }
  return (1 / 2) * (1 + sign * erf);
}
```

Ranking Algorithm (GitHub)

Implementation

```
if (normalizedScore < LVL_GURU) {
  level = 95;
  rank='Guru'
}
if (
  normalizedScore >= LVL_GURU &&
  normalizedScore < LVL_PROFESSIONAL
) {
  level = 80;
  rank='Professional'
}
if (
  normalizedScore >= LVL_PROFESSIONAL &&
  normalizedScore < LVL_ADVANCED
) {
  level = 70;
  rank='Advanced'
}
if (normalizedScore >= LVL_ADVANCED && normalizedScore < LVL_INTERMEDIATE) {
  level = 50;
  rank='Intermediate'
}
if (normalizedScore >= LVL_INTERMEDIATE && normalizedScore < LVL_BEGINNER) {
  level = 30;
  rank='Beginner'
}
if(normalizedScore>=LVL_BEGINNER && normalizedScore<LVL_NEWBIE){
  level=10;
  rank='Newbie'
}
```

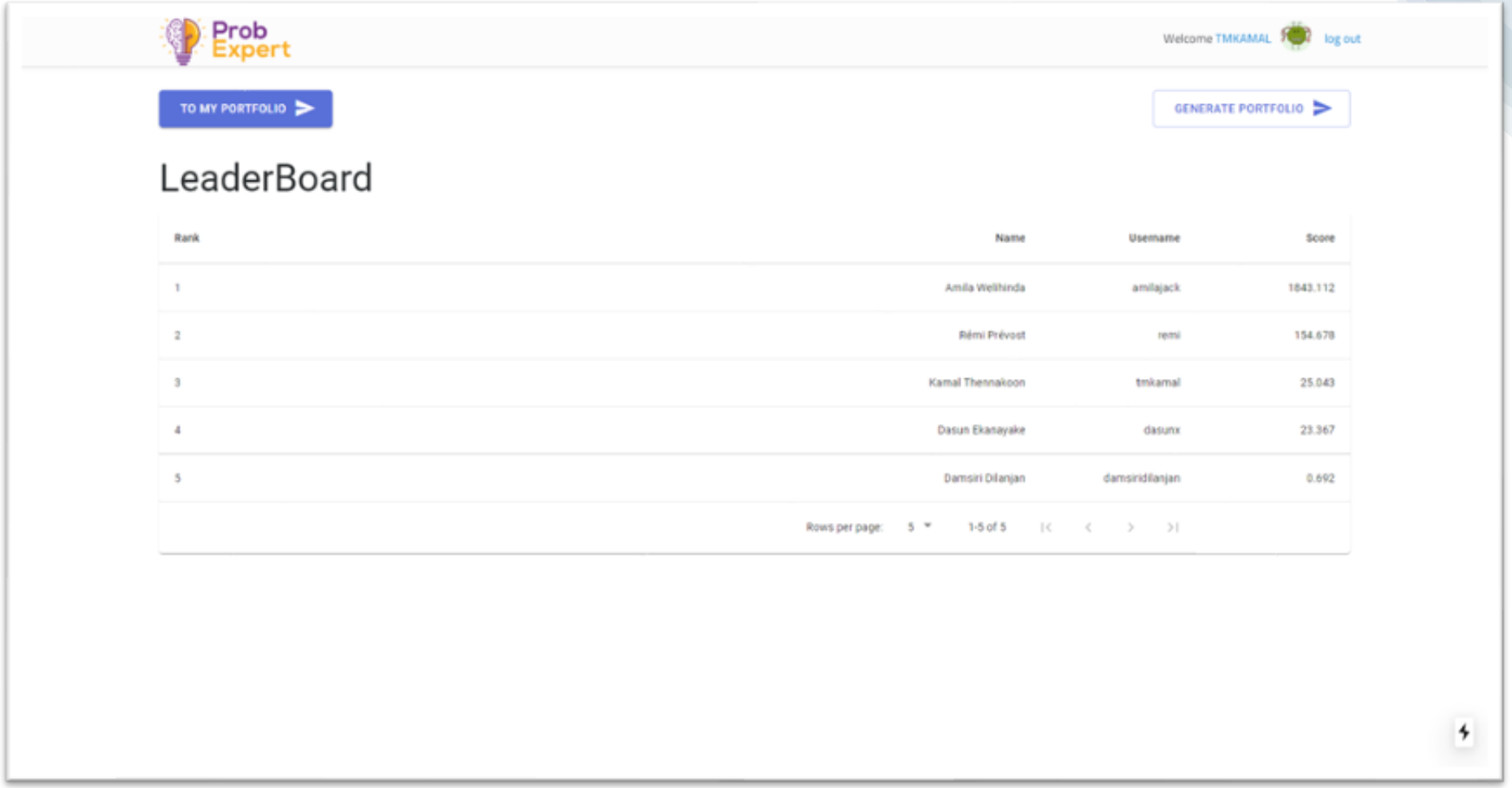
Ranking Algorithm (GitHub)

Implementation

```
return request(
  {
    query: `
    query userInfo($login: String!) {
      user(login: $login) {
        name
        login
        avatarUrl
        bio
        company
        createdAt
        isGitHubStar
        location
        websiteUrl
        organizations {
          totalCount
        }
        contributionsCollection {
          totalCommitContributions
          restrictedContributionsCount
        }
        repositoriesContributedTo(first: 1, contributionTypes: [COMMIT, ISSUE, PULL_REQUEST, REPOSITORY]) {
          totalCount
        }
        pullRequests(first: 1) {
          totalCount
        }
        issues(first: 1) {
          totalCount
        }
        followers {
          totalCount
        }
        repositories(first: 100, ownerAffiliations: OWNER, orderBy: {direction: DESC, field: STARGAZERS}) {
          totalCount
          nodes {
            stargazers {
              totalCount
            }
          }
        }
      }
    }
    `,
    variables,
  },
  {
    Authorization: `bearer ${token}`,
  },
);
```

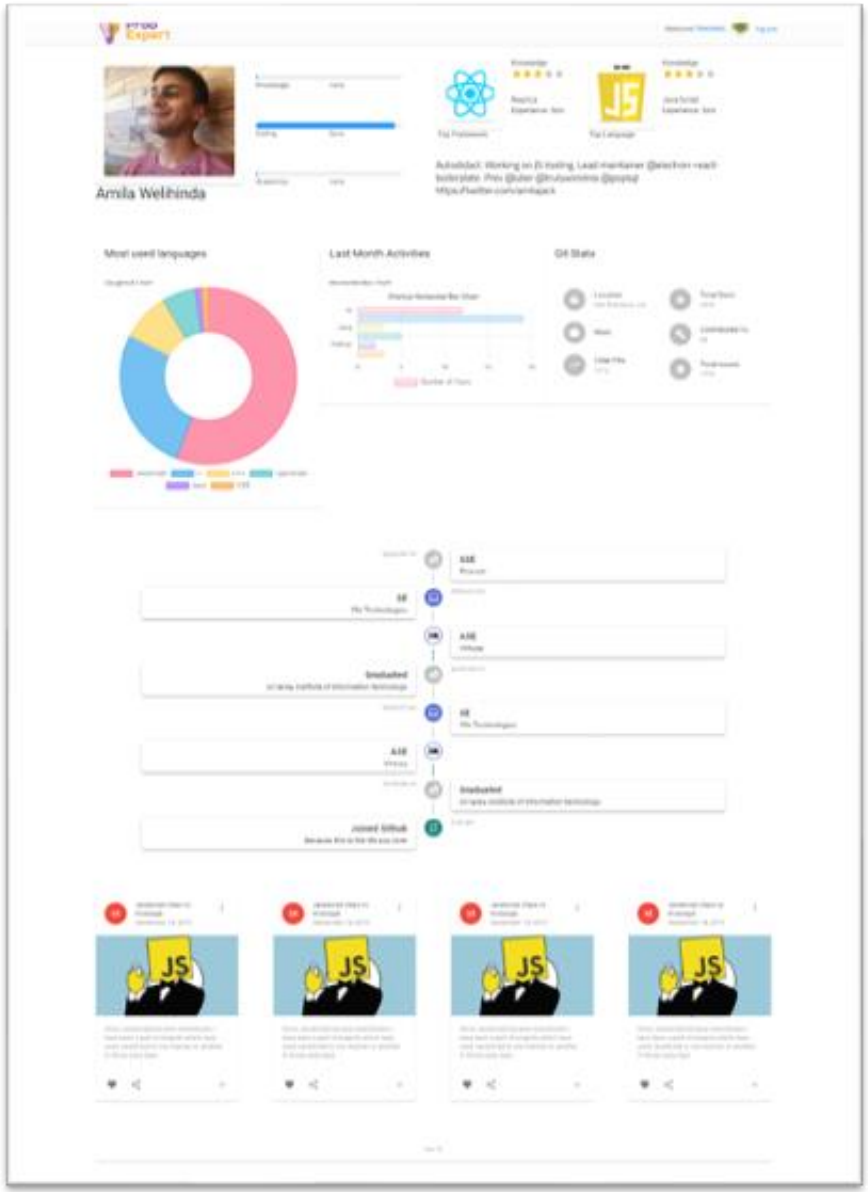
GraphQL Query
(GitHub Info)

Implementation



User Interface (Leaderboard)

Implementation



User Interface (Portfolio)

References

[1] C. Hauff and G. Gousios, "Matching GitHub developer profiles to job advertisements," *IEEE Int. Work. Conf. Min. Softw. Repos.*, vol. 2015-Augus, pp. 362–366, 2015, doi: 10.1109/MSR.2015.41.

[2] J. Zhang, M. S. Ackerman, and L. Adamic, "Expertise Networks in Online Communities: Structure and Algorithms."

[3] A. Dargahi Nobari, M. Neshati, and S. Sotudeh Gharebagh, "Quality-aware skill translation models for expert finding on StackOverflow," *Inf. Syst.*, vol. 87, Jan. 2020, doi: 10.1016/j.is.2019.07.003.

[4] https://stackoverflow.com/questions/5259421/cumulative-distribution-function-in-javascript/5263759#5263759

IT18013610
Ekanayake P.M.D.P

B.Sc. (Hons) Degree in Information Technology Specialized in Software Engineering

Introduction

Generating an answer for users' questions automatically using public information

IT18013610 | Ekanayake P.M.D.P | 2021-155

What is an automated answer?

Fastest way to get an answer from QandA platform

Contains information from multiple resources

Reliable

Up-to-date information

Why an automated answer?

- When using question and answering platforms users must wait for other users' answers.
- Most users have to wait at least 1 hour to get an answer from another user.
- Sometimes the first answer will not contain the information question asker has been looking for.
- If there is a way to find more than one kind of resources at one place (code, blog, video in this research) that would be an extremely valuable and time saving for users.

Benefits of an automated answer?

Find reliable and up-to-date information.

Different variance of resources.

Less waiting time on getting an answer.

Save money and bandwidth

Research Gap

Platform	Features				
	Find similar question	Provide automatic answer	Code Support	Answers from multiple sources	One to one communication
Quora	Yes	No	No	No	No
Stackoverflow	Yes	No	Yes	No	No
Stack exchange	Yes	No	Yes	No	No
ProbExpert	Yes –in Realtime	Yes	Yes	Yes	Yes

Research Problems

How to reduce waiting time spent on getting an answer from QandA platforms.

How to create an answer from multiple resources automatically

How to find YouTube video and GitHub repositories relevant to the question.

Reduce the time spent on multiple visits to different web sites for gather information.

How to provide access to users with low network bandwidth.

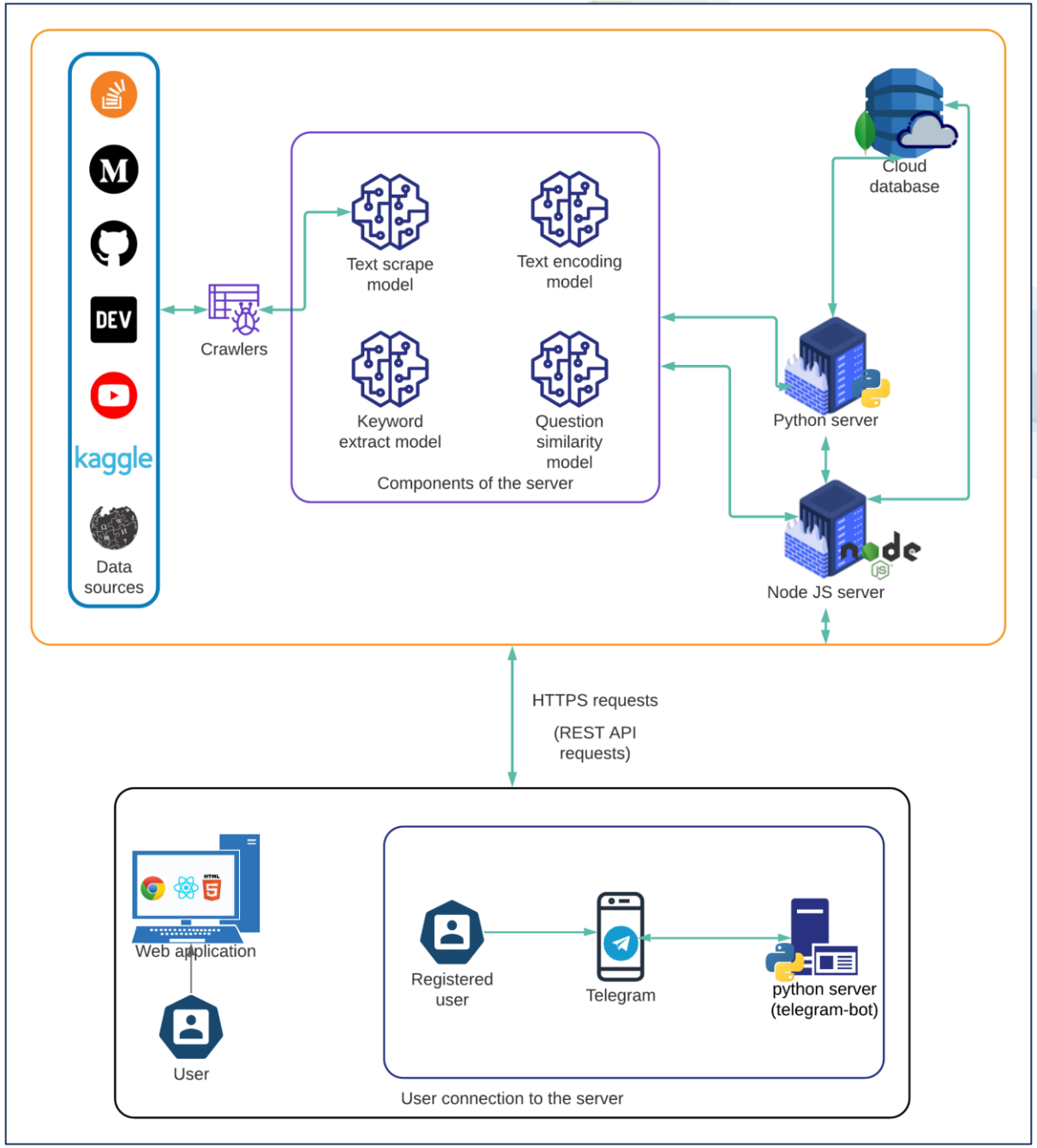
Specific and Sub Objectives

- **Generate an answer automatically for users' questions with multiple resources.**
  - Find reliable and up-to-date resources and scrape information.
  - Keyword extraction and check the relativeness of the information.
  - Find similar questions in database.
  - Find relevant YouTube video and GitHub repository.
  - Combine all the information from each resources and present them to user in a clear user interface

Research Methodology

IT18013610 | Ekanayake P.M.D.P | 2021-155

System diagram



Authenticating and getting user's question.

Authentication

- JWT tokens
- NodeJS with express
- MongoDB database

Asking questions

- NodeJS and Express server
- Server side rendered NextJS web app made with ReactJS.

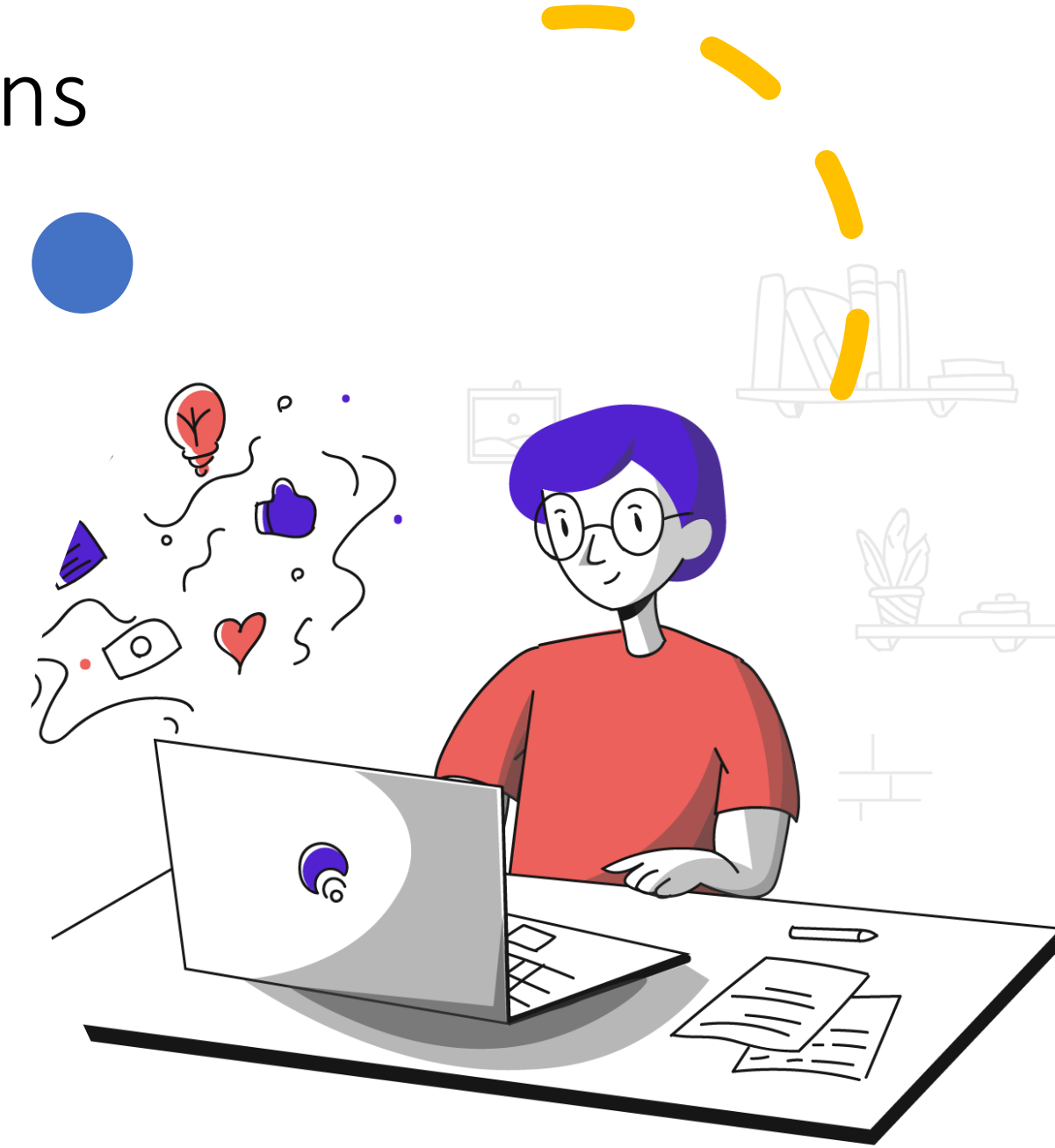
Telegram Bot

- Sign in
- Ask questions

Check for similar questions

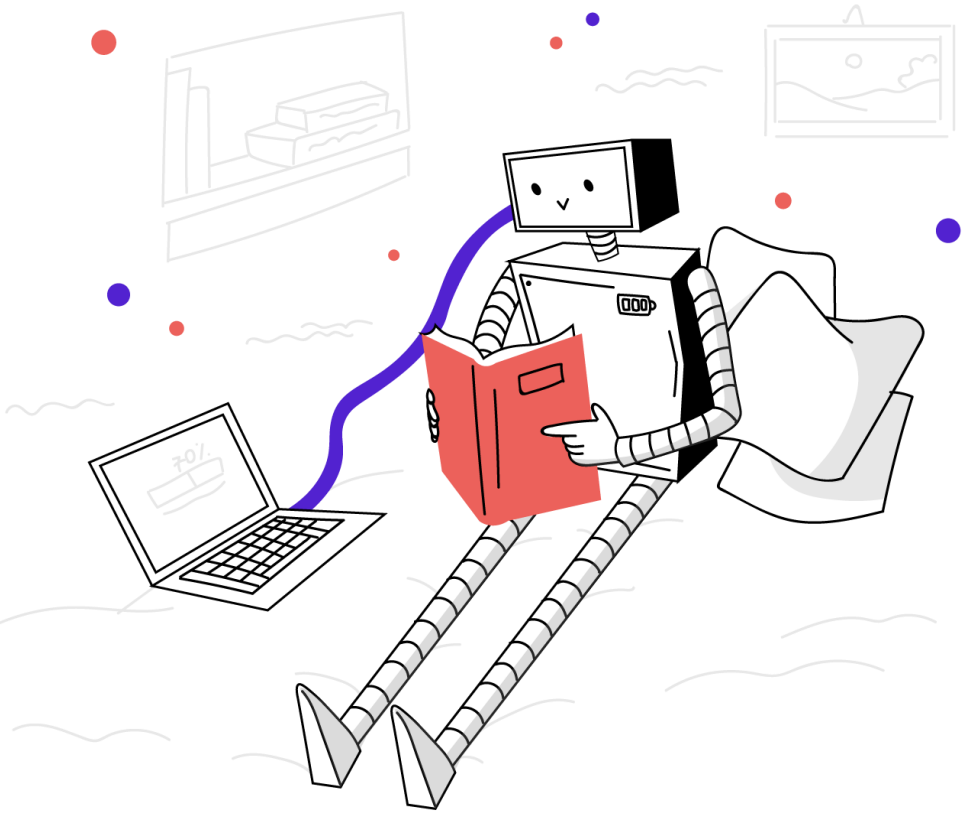
- Find similar questions in database using cosine similarity.

- User typing the question
- Processing the question in the server
- Pass the question to the python machine learning model
- Encode the question using
- Find similar questions in the db.
- Show the similar questions in the web app



Find reliable and up-to-date resources

- Use google search filters to filter out new data
- Specify predefined web sites to search information about



Scrape information

- Custom web scraper
- In order to use the information, we found on previous step we must scrape them and save on a database
- Beautiful soup4, scrapy, pandas for scrape information.

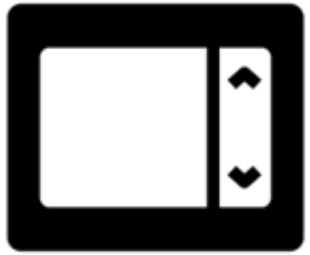
Beautiful Soup

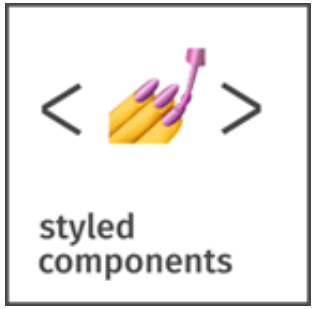
mongoDB.

python™

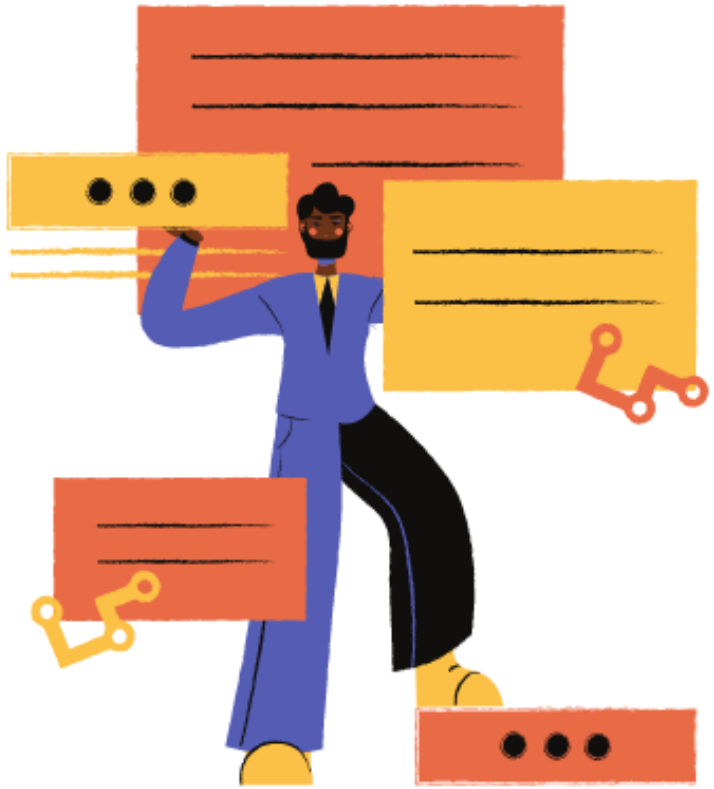
Display answer to the user

- After generating an answer with multiple resources, we must have a way to show it to users
- Otherwise, there is no use to it
- We used styled components, sass to style our web app, and Iframe to mount youtube videos





Evidences For Completion



Scraping Stackoverflow Questions

```
class STOF:
    def __init__(self, qtitle, keywords=[], description=""):
        self.qtitle = qtitle
        self.keywords = keywords
        self.description = description
        self.urls = []

    def searchQuestion(self):
        html_page = requests.get(
            f"https://google.com/search?q=site%3Astackoverflow.com+{self.
        )
        soup = BeautifulSoup(html_page.content, "html.parser")

        for link in soup.findAll("a"):
            if "https://stackoverflow.com" in link["href"]:
                self.urls.append(self.extractSOFUrl(link["href"]))
        ans = self.viewStackUrls()
        return ans

    def extractSOFUrl(self, uriString):
        uriTrimmed = uriString[7:]
        uriTrimmed = re.match(r"^.*?\&sa=", uriTrimmed).group(0)
        return uriTrimmed.replace("&sa=", "")

    def viewStackUrls(self):
        return self.viewStackOverFlowQuestion(self.urls[0])

    def viewStackOverFlowQuestion(self, url):
        html_page = requests.get(url)
        soup = BeautifulSoup(html_page.content, "html.parser")
        dom = etree.HTML(str(soup))
        answers_count = dom.xpath('//*[@id="answers-header"]/div/div[1]/h
            0
        ].text.strip()
        answer = {"url": url}

        if answers_count != "":
            try:
                verified_answer = dom.xpath(
                    '//*[@class="answer accepted-answer"]/div/div[2]/div[
                )[0]
                answer["content"] = etree.tostring(verified_answer).decod
                answer["status"] = "Verified"
                # with open("verified_answer.html", "wb") as htmlF:
                #     htmlF.write(etree.tostring(verified_answer))
                #     htmlF.close()
```

Scraping Blog articles

```python
class Medium:
    def __init__(self, title, tags):
        self.title = title
        self.tags = tags

    def getApiKey(self):
        """
        Returns an API key for retrieve json data
        """
        api_keys = [
            "2rk1eg4sexdnp5umrwtwbtwd2insqvgzvejooqrn",
            "yit6ytfcs3ziawdgasfd3bgkbf4tef1m2nzdxvnz",
            "mpawymyrc6derrwmgodowfsaabtuoes4iiwintd7",
        ]
        return random.choice(api_keys)

    def google(self, query):
        """
        Use a query to search using google search enging
        """
        search_args = (query, 1)
        gsearch = GoogleSearch()
        gresults = gsearch.search(*search_args)
        return gresults["links"]

    def getValidUrls(self, links):
        """
        Validate and filter out the urls.
        Returns the urls that contain medium.com in it as a list
        """
        validUrls = []
        for i in links:
            if "medium.com" in i:
                uriTrimmed = re.match(r"^.*?\&sa=", i[29:]).group(0)
                ur = uriTrimmed.replace("&sa=", "")
                validUrls.append(ur)
        return validUrls
```

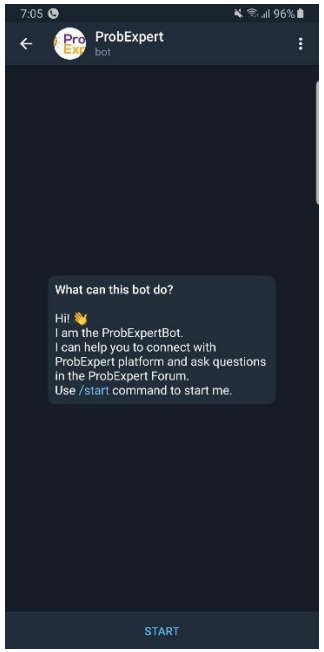
```python
    def getValidSets(self, validUrls):
        """
        Extract usernames and article id's from article url
        pass a list of urls ⇒ returns objects list that contain usernam and article id
        """
        validSets = []
        for url in validUrls:
            try:
                vset = {}
                print(url)
                username = re.search(r"https://medium.com/([^/?]+)", url).group(1)
                tag = re.search(r"https://medium.com/([^/?]+)/([^/?]+)", url).group(2)
                vset["username"] = username
                vset["tag"] = tag
                validSets.append(vset)
            except Exception as e:
                print(e)
                continue
        return validSets

    def getBlogs(self, username, tag):
        """
        Get the content of the article
        """
        blog = {}
        try:
            response = requests.get(
                f"https://api.rss2json.com/v1/api.json?rss_url=https%3A%2F%2Fmedium.com%2
            )
            if response.status_code == 200:
                res = response.json()
                for item in res["items"]:
                    if tag in item["link"]:
                        blog = item
        except Exception as e:
            print(e)
        return blog

    def getMediumArticles(self):
        """
        return a list of articles and/or resources
        """
        links = self.google(f"site:medium.com {self.title} after:2020-01-01")
        validUrls = self.getValidUrls(links)
        validSets = self.getValidSets(validUrls)
        blogs = []
        for validset in validSets:
```

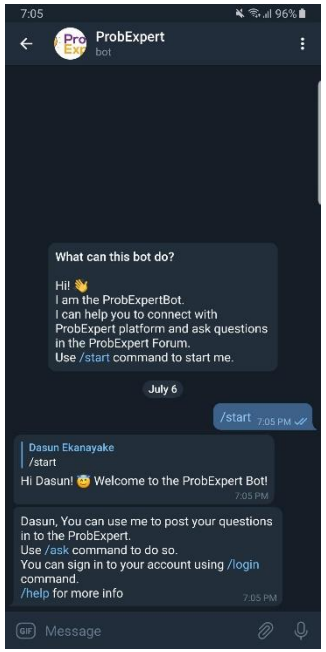
Telegram bot

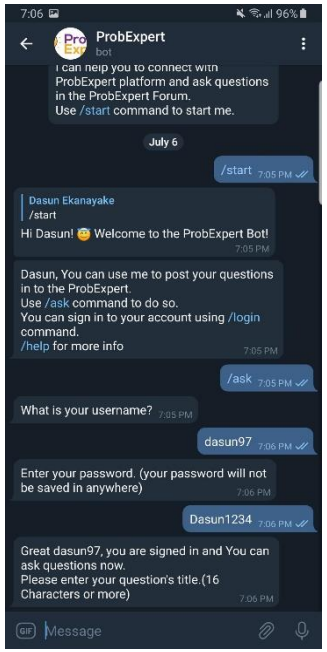
```python
user = {"name": None, "username": None, "token": None}
question = {"title": None, "text": None, "tags": []}
bot = telebot.TeleBot(API_TOKEN)
server = Flask(__name__)


@bot.message_handler(commands=["help"])
def help(message):
    bot.send_message(
        message.chat.id, HELP_MSG.format(name=message.from_user.first_name)
    )


# Handle '/start' and '/help'
@bot.message_handler(commands=["start"])
def send_welcome(message):
    bot.reply_to(message, WELCOME_MSG.format(name=message.from_user.first_name))
    bot.send_message(
        message.chat.id, START_MSG.format(name=message.from_user.first_name)
    )


# Handle '/ask'
@bot.message_handler(commands=["ask"])
def ask(message):
    if user["token"] != None:
        msg = bot.send_message(
            message.chat.id, SIGN_IN_GREETING.format(username=user["username"])
        )
        bot.register_next_step_handler(msg, save_title)
    else:
        checkSignIn(message)


# Handle '/login'
@bot.message_handler(commands=["login"])
def login(message):
    checkSignIn(message)


def checkSignIn(message):
    global user
    if user["token"] != None:
        markup = types.ReplyKeyboardMarkup(one_time_keyboard=True)
        markup.add(ASK_QUESTION, SIGN_OUT)
        msg = bot.send_message(
```

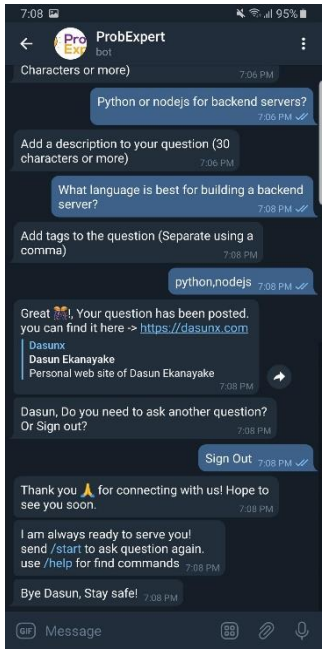
```python
            SIGN_IN_GREETING.format(username=user["username"]),
            reply_markup=markup,
        )
        bot.register_next_step_handler(msg, next_step)
    else:
        signIn(message)


def signIn(message):
    global user
    if user["username"] == None:
        msg = bot.send_message(message.chat.id, USERNAME_REQ)
        bot.register_next_step_handler(msg, add_username)
    else:
        msg = bot.send_message(message.chat.id, PASSWORD_REQ)
        bot.register_next_step_handler(msg, get_token)


def add_username(message):
    global user
    user["username"] = message.text
    signIn(message)


def get_token(message):
    global user
    password = message.text
    try:
        headers = {"Content-Type": "application/json"}
        data = {"username": user["username"], "password": password}
        response = requests.post(
            f"{NODEJS_BASE_URL}authenticate",
            headers=headers,
            json=data,
        )
        if response.status_code == 200:
            # handle success
            user["token"] = response.json()["token"]
            ask(message)
        else:
            bot.send_message(
                message.chat.id,
                SIGN_IN_ERROR.format(
                    name=message.from_user.first_name, error=response.json()["message"]
                ),
            )
            markup = types.ReplyKeyboardMarkup(one_time_keyboard=True)
            markup.add(TRY_SIGN_IN_AGAIN, EXIT)
            msg = bot.send_message(
```

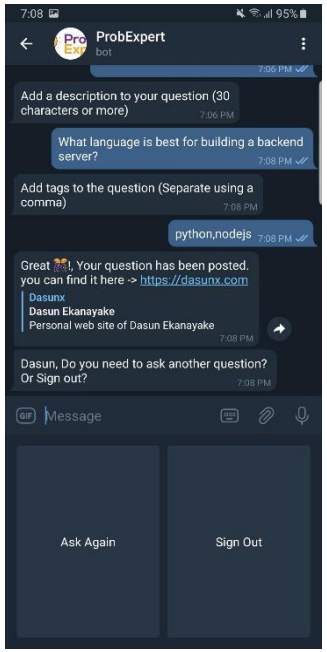
Telegram bot

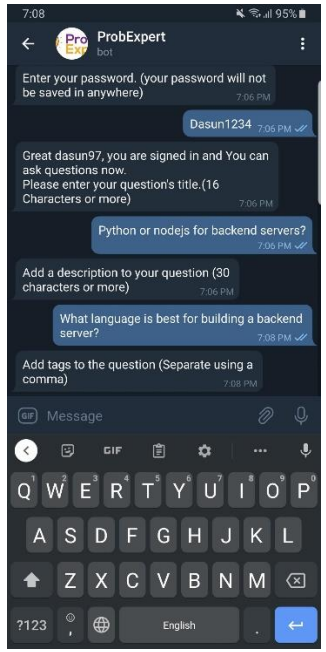






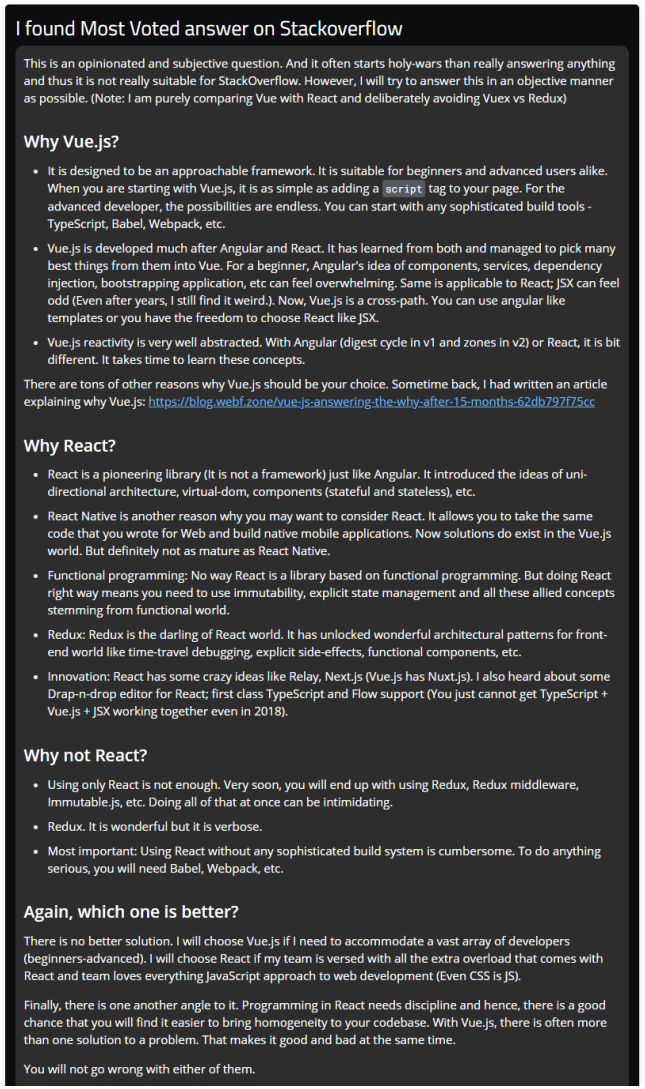






Automated answer (UI)

Automated Answer



I found Most Voted answer on Stackoverflow

This is an opinionated and subjective question. And it often starts holy-wars than really answering anything and thus it is not really suitable for StackOverflow. However, I will try to answer this in an objective manner as possible. (Note: I am purely comparing Vue with React and deliberately avoiding Vuex vs Redux)

Why Vue.js?

- It is designed to be an approachable framework. It is suitable for beginners and advanced users alike. When you are starting with Vue.js, it is as simple as adding a `script` tag to your page. For the advanced developer, the possibilities are endless. You can start with any sophisticated build tools - TypeScript, Babel, Webpack, etc.
- Vue.js is developed much after Angular and React. It has learned from both and managed to pick many best things from them into Vue. For a beginner, Angular's idea of components, services, dependency injection, bootstrapping application, etc can feel overwhelming. Same is applicable to React; JSX can feel odd (Even after years, I still find it weird.). Now, Vue.js is a cross-path. You can use angular like templates or you have the freedom to choose React like JSX.
- Vue.js reactivity is very well abstracted. With Angular (digest cycle in v1 and zones in v2) or React, it is bit different. It takes time to learn these concepts.

There are tons of other reasons why Vue.js should be your choice. Sometime back, I had written an article explaining why Vue.js: https://blog.webf.zone/vue-js-answering-the-why-after-15-months-62db797f75cc

Why React?

- React is a pioneering library (It is not a framework) just like Angular. It introduced the ideas of uni-directional architecture, virtual-dom, components (stateful and stateless), etc.
- React Native is another reason why you may want to consider React. It allows you to take the same code that you wrote for Web and build native mobile applications. Now solutions do exist in the Vue.js world. But definitely not as mature as React Native.
- Functional programming: No way React is a library based on functional programming. But doing React right way means you need to use immutability, explicit state management and all these allied concepts stemming from functional world.
- Redux: Redux is the darling of React world. It has unlocked wonderful architectural patterns for front-end world like time-travel debugging, explicit side-effects, functional components, etc.
- Innovation: React has some crazy ideas like Relay, Next.js (Vue.js has Nuxt.js). I also heard about some Drap-n-drop editor for React; first class TypeScript and Flow support (You just cannot get TypeScript + Vue.js + JSX working together even in 2018).

Why not React?

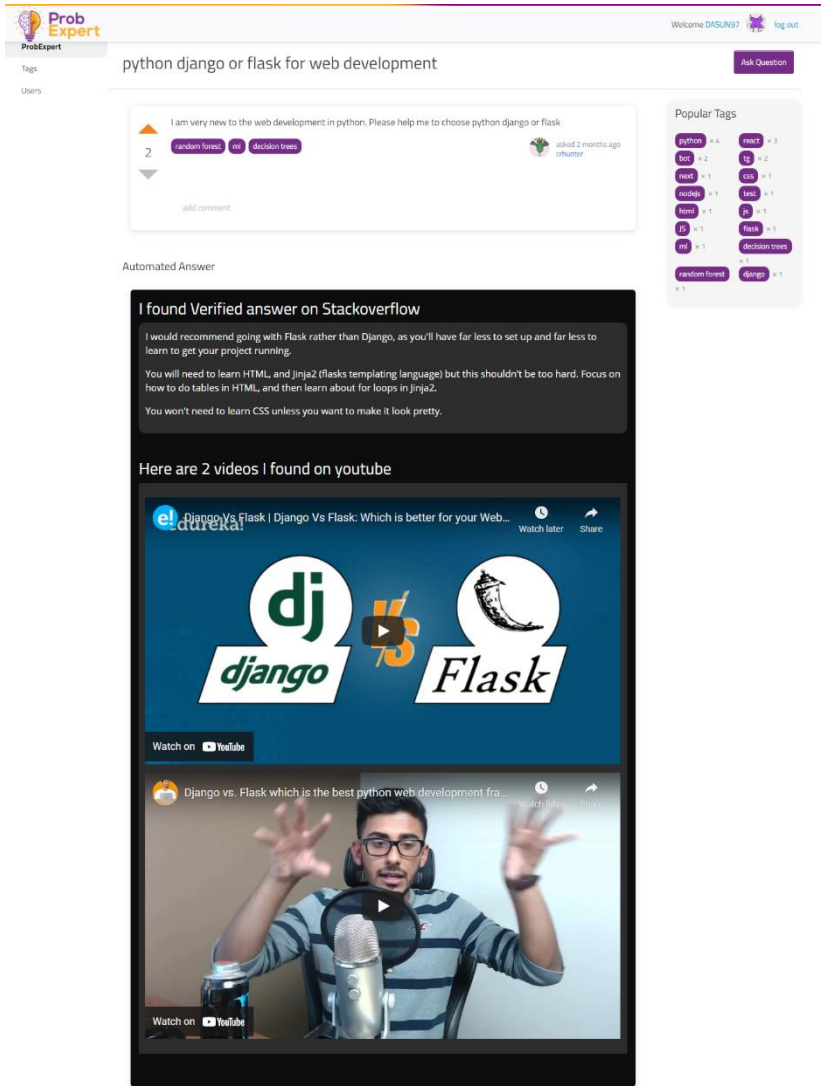
- Using only React is not enough. Very soon, you will end up with using Redux, Redux middleware, Immutable.js, etc. Doing all of that at once can be intimidating.
- Redux. It is wonderful but it is verbose.
- Most important: Using React without any sophisticated build system is cumbersome. To do anything serious, you will need Babel, Webpack, etc.

Again, which one is better?

There is no better solution. I will choose Vue.js if I need to accommodate a vast array of developers (beginners-advanced). I will choose React if my team is versed with all the extra overload that comes with React and team loves everything JavaScript approach to web development (Even CSS is JS).

Finally, there is one another angle to it. Programming in React needs discipline and hence, there is a good chance that you will find it easier to bring homogeneity to your codebase. With Vue.js, there is often more than one solution to a problem. That makes it good and bad at the same time.

You will not go wrong with either of them.



References

[1] W. Song, M. Feng, N. Gu and L. Wenyin, "Question Similarity Calculation for FAQ Answering," Third International Conference on Semantics, Knowledge and Grid (SKG 2007), Xi'an, China, 2007, pp. 298-301, doi: 10.1109/SKG.2007.32.

[2] I.Ali, R. Y. Chang and C. Hsu, "CRED: Credibility-Enabled Social Network Based Q&A System for Assessing Answers Correctness," 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea (South), 2020, pp. 1-6, doi: 10.1109/WCNC45663.2020.9120750.

[3] H. Li, Z. Xing, X. Peng and W. Zhao, "What help do developers seek, when and how?," 2013 20th Working Conference on Reverse Engineering (WCRE), Koblenz, Germany, 2013, pp. 142-151, doi: 10.1109/WCRE.2013.6671289.

IT18011012 | Ranasinghe R.M.A.K

B.Sc. (Hons) Degree in Information Technology Specialized in Software Engineering

Introduction

Optimal answer generation through up-voted answers

Background

People preferred to use Q&A sites to find solutions to challenges they encountered throughout their studies rather than books, documents, or tutorials

The abundance of knowledge has resulted in the problem of information overload.

Average person spend at least 30 minutes of time searching for a solution.

Availability of multiple solutions for a problem.

Research Gap

- When creating an optimized answer from a set of given answers, there are few things to be considered.
  - Keyword extraction
  - Summarization
  - Answer quality
  - Merge answers into a single answer

- Many researches are mainly concerned with summarization, keyword extraction and answer quality.
- There are no research available on how to merge answers into a single optimized answer

Research Question

Reduce wasting time on finding a solution that works for a problem

How to combine all the alternative solutions into a one optimized answer without harming the semantic information

Benefits of an optimized answer

Save time

Improved Efficiency

Accurate Information

Higher Quality Results

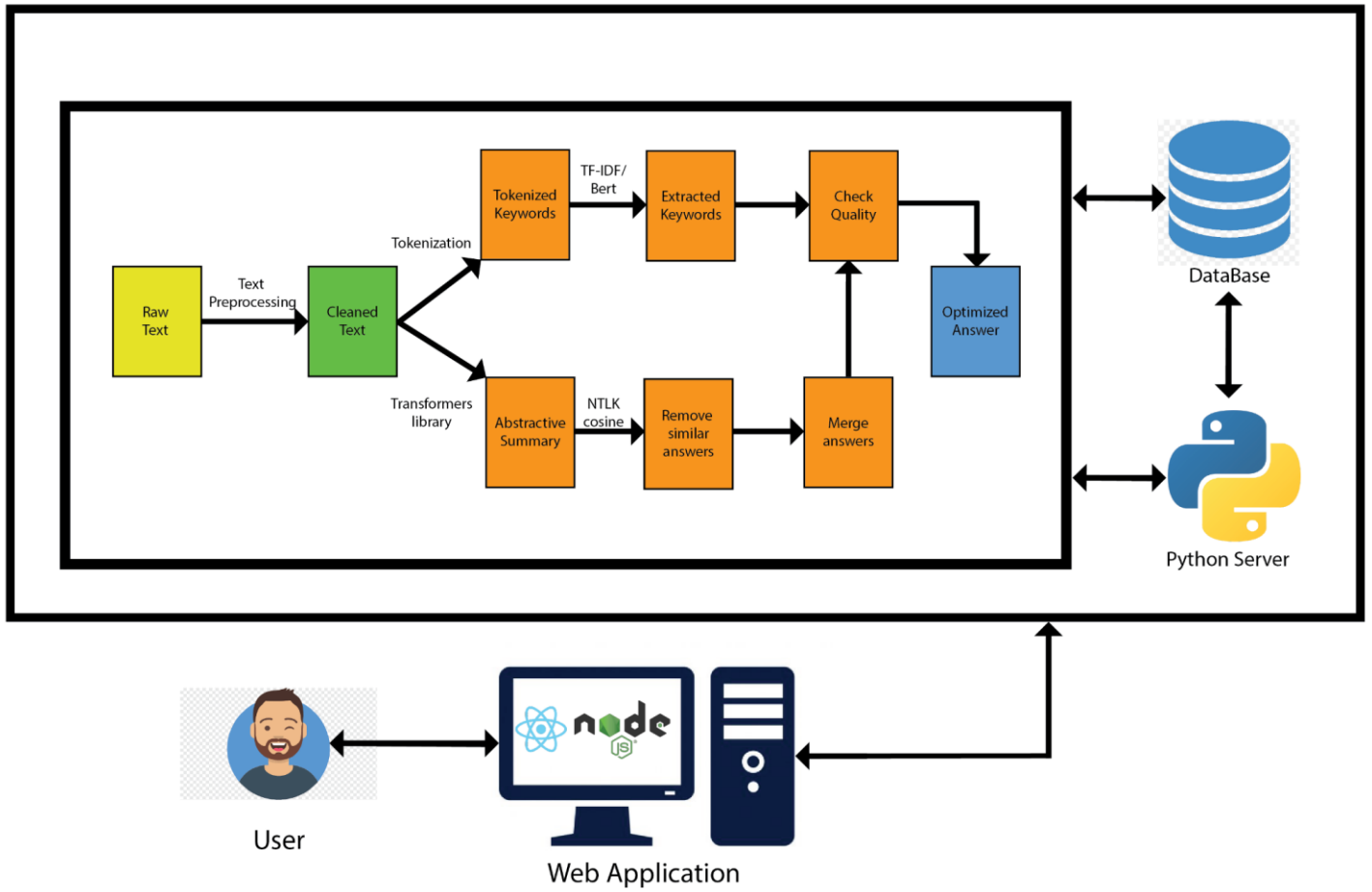
Specific and Sub objectives

- Generating an optimized answer using the top voted answers
  - Data Retrieval
  - Check and remove similar answers
  - Keyword extraction
  - Text preprocessing
  - Summarize answers
  - Merge answers together
  - Check quality of the merged answer

Research Methodology

IT18011012 | Ranasinghe R.M.A.K. | 2021-155

System Diagram



Methodology

➢Question and Answer retrieval

- Using an extraction tool, the questions and answers will be extracted with relevant details. When extracting if there are more than 1 answer, top voted 4 answers will be taken for the optimization.

➢Check and remove similar answers

- Once user post a question to the platform, the top voted answers given to that question will be compared with each other using cosine similarity with natural language toolkit.

Methodology continued

➢ Keyword extraction

- Extracted keywords will be used after summarization to check whether the summarization process was correct, and the answers are retained to a certain value. So, the values before and after summarization must have close values.

➢ Preprocess the Answers

In order to Preprocess the answers, the answers which were left after the similarity checking will be taken. Then through the preprocess technique;

- Letter lowercasing,
- HTML tag/URL removal,
- Stop word removal,
- Remove special characters, processes will take place.

Methodology continued

➢Summarize answers

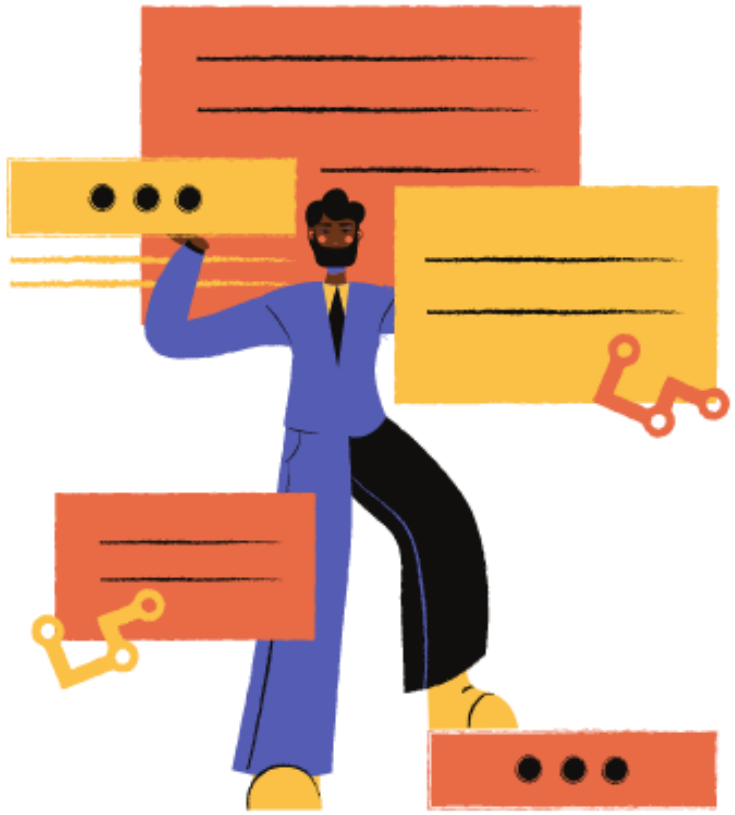
- Selected answers will undergo a summarization process. To be exact an abstractive summarization will take place. After that again key words will be extracted and checked with the previously selected key words for the value similarity.

➢Merge answers together

- Finally, the left answers will be combined to create a technically correct and with the accurate semantic information.

➢Display Optimized answer

- After the end of all the process , final output will be shown to the user via the web application.



Evidence for Completion

IT18011012 | Ranasinghe R.M.A.K. | 2021-155

Data Retrieval

➢ Libraries

- PyMongo
- Bson
- ReactJs

➢ Query used

- textcursor = list(collection.aggregate([
  {"$match": {"$and":[{"_id" : objInstance},{"answers.3": { "$exists": "true" } }]} },
  {"$unwind": "$answers"},
  {"$sort": {"answers.score":-1}},
  {"$project": {"_id" : 0,"answers.text" : 1}}))

➢ Data Store

- MongoDB

Implementation

```
from pymongo import MongoClient
from pprint import pprint
from bson import ObjectId

text = ""

# To connect mongodb
client = MongoClient("mongodb+srv://admin3:ash1234@cluster0.u4vl4.mongodb.net/test?retryWrites=true&w=majority")

# List all databases
client.list_database_names()

# Connect a database
db = client.test

# List all collections in the database
db.list_collection_names()

# Connect collection
collection = db.questions

#Query data by ObjectId
id = "6158a73095189b471c65d1c2"
objInstance = ObjectId(id)
collection.find_one({"_id": objInstance})

#Retrieve data by ObjectId and sort by answer score
cursor = list(collection.aggregate(
   [
    {"$match": {"_id" : objInstance} },
    {"$unwind": "$answers"},
    {"$sort": {"answers.score":-1}},
    {"$project": {"_id" : 1,"title" : 1,"answers._id" : 1,"answers.text" : 1,"answers.score": 1}}

   ]
))

#Final retrieval
textcursor = list(collection.aggregate(
   [
    {"$match": {"$and":[{"_id" : objInstance},{"answers.3": {  "$exists": "true" } }]} },
    {"$unwind": "$answers"},
    {"$sort": {"answers.score":-1}},
    {"$project": {"_id" : 0,"answers.text" : 1}}

   ]
))
pprint(textcursor)
```

Check and Remove similar answers

➢ **Libraries**

- Numpy
- Sklearn
- Cosine Similarity
- CountVectorizer

- Cosine similarity is a widely implemented metric in information retrieval and related studies. This metric model a text document as a vector of terms. By this model, the similarity between two documents can be derived by calculating cosine value between two documents' term vectors [10].

Implementation

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer

def cosine_similarity(x, y):

    # Ensure length of x and y are the same
    if len(x) != len(y) :
        return None

    # Compute the dot product between x and y
    dot_product = np.dot(x, y)

    # Compute the L2 norms (magnitudes) of x and y
    magnitude_x = np.sqrt(np.sum(x**2))
    magnitude_y = np.sqrt(np.sum(y**2))

    # Compute the cosine similarity
    cosine_similarity = dot_product / (magnitude_x * magnitude_y)

    return cosine_similarity

corpus = [  'The short and easy answer is that git pull is simply '
                    'git fetch followed by git merge.\n'
                    '\n'
                    'It is very important to note that git pull will '
                    'automatically merge whether you like it or not. This '
                    "could, of course, result in merge conflicts. Let's say "
                    'your remote is origin and your branch is master. If you '
                    'git diff origin/master before pulling, you should have '
                    'some idea of potential merge conflicts and could '
                    'prepare your local branch accordingly.',
            'git fetch is similar to pull but doesnt merge. it '
                    'fetches remote updates (refs and objects) but your '
                    'local stays the same (i.e. origin/master gets updated '
                    'but master stays the same) .\n'
                    '\n'
                    'git pull pulls down from a remote and instantly '
                    'merges.',
            'One use case of git fetch is that the following will '
                    'tell you any changes in the remote branch since your '
                    'last pull... so you can check before doing an actual '
                    'pull, which could change files in your current branch '
                    'and working copy.' ,
            'You can do a git fetch at any time to update your '
                    'remote-tracking branches under refs/remotes/<remote>/. '
                    'This operation never changes any of your own local '
                    'branches under refs/heads, and is safe to do without '
                    'changing your working copy. I have even heard of people '
                    'running git fetch periodically in a cron job in the '
                    'background (although I wouldnt recommend doing this).\n'
                    '\n'
                    'A git pull is what you would do to bring a local branch '
                    'up-to-date with its remote version, while also updating '
                    'your other remote-tracking branches.']
```

```
corpus = [  'The short and easy answer is that git pull is simply '
                    'git fetch followed by git merge.\n'
                    '\n'
                    'It is very important to note that git pull will '
                    'automatically merge whether you like it or not. This '
                    "could, of course, result in merge conflicts. Let's say "
                    'your remote is origin and your branch is master. If you '
                    'git diff origin/master before pulling, you should have '
                    'some idea of potential merge conflicts and could '
                    'prepare your local branch accordingly.',
            'git fetch is similar to pull but doesnt merge. it '
                    'fetches remote updates (refs and objects) but your '
                    'local stays the same (i.e. origin/master gets updated '
                    'but master stays the same) .\n'
                    '\n'
                    'git pull pulls down from a remote and instantly '
                    'merges.',
            'One use case of git fetch is that the following will '
                    'tell you any changes in the remote branch since your '
                    'last pull... so you can check before doing an actual '
                    'pull, which could change files in your current branch '
                    'and working copy.' ,
            'You can do a git fetch at any time to update your '
                    'remote-tracking branches under refs/remotes/<remote>/. '
                    'This operation never changes any of your own local '
                    'branches under refs/heads, and is safe to do without '
                    'changing your working copy. I have even heard of people '
                    'running git fetch periodically in a cron job in the '
                    'background (although I wouldnt recommend doing this).\n'
                    '\n'
                    'A git pull is what you would do to bring a local branch '
                    'up-to-date with its remote version, while also updating '
                    'your other remote-tracking branches.']

# Create a matrix to represent the corpus
X = CountVectorizer().fit_transform(corpus).toarray()

print(X)

cos_sim_1_2 = cosine_similarity(X[0, :], X[1, :])
cos_sim_1_3 = cosine_similarity(X[0, :], X[2, :])
cos_sim_1_4 = cosine_similarity(X[0, :], X[3, :])
cos_sim_2_3 = cosine_similarity(X[1, :], X[2, :])
cos_sim_2_4 = cosine_similarity(X[1, :], X[3, :])
cos_sim_3_4 = cosine_similarity(X[2, :], X[3, :])

print('Cosine Similarity between: ')
print('\tDocument 1 and Document 2: ', cos_sim_1_2)
print('\tDocument 1 and Document 3: ', cos_sim_1_3)
print('\tDocument 1 and Document 4: ', cos_sim_1_4)
print('\tDocument 2 and Document 3: ', cos_sim_2_3)
print('\tDocument 2 and Document 4: ', cos_sim_2_4)
print('\tDocument 3 and Document 4: ', cos_sim_3_4)
```

Key word extraction

- **Libraries**
  - NLTK
  - TF-IDF
  - Math

  - TFIDF, which stands for term frequency–inverse document frequency, is a numerical statistic that is meant to indicate the importance of a word in a collection or corpus of documents.

Implementation

```
from nltk import tokenize
from operator import itemgetter
import math
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
from pprint import pprint

doc = '''You can do a git fetch at any time to update your
                    remote-tracking branches under refs/remotes.
                    This operation never changes any of your own local
                    branches under refs/heads, and is safe to do without
                    changing your working copy. I have even heard of people
                    running git fetch periodically in a corn job in the
                    background (although I wouldn't recommend doing this).
                    A git pull is what you would do to bring a local branch
                    up-to-date with its remote version, while also updating
                    your other remote-tracking branches.'''

#Find total words in the document
total_words = doc.split()
total_word_length = len(total_words)
print(total_word_length)

#Find the total number of sentences
total_sentences = tokenize.sent_tokenize(doc)
total_sent_len = len(total_sentences)
print(total_sent_len)

#Calculate TF for each word
tf_score = {}
for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in tf_score:
            tf_score[each_word] += 1
        else:
            tf_score[each_word] = 1

# Dividing by total_word_length for each dictionary element
tf_score.update((x, y/int(total_word_length)) for x, y in tf_score.items())
#print(tf_score)

#Function to check if the word is present in a sentence list
def check_sent(word, sentences):
    final = [all([w in x for w in word]) for x in sentences]
    sent_len = [sentences[i] for i in range(0, len(final)) if final[i]]
    return int(len(sent_len))

#Calculate IDF for each word
idf_score = {}
for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in idf_score:
            idf_score[each_word] = check_sent(each_word, total_sentences)
```

```
#Find total words in the document
total_words = doc.split()
total_word_length = len(total_words)
print(total_word_length)

#Find the total number of sentences
total_sentences = tokenize.sent_tokenize(doc)
total_sent_len = len(total_sentences)
print(total_sent_len)

#Calculate TF for each word
tf_score = {}
for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in tf_score:
            tf_score[each_word] += 1
        else:
            tf_score[each_word] = 1

# Dividing by total_word_length for each dictionary element
tf_score.update((x, y/int(total_word_length)) for x, y in tf_score.items())
#print(tf_score)

#Function to check if the word is present in a sentence list
def check_sent(word, sentences):
    final = [all([w in x for w in word]) for x in sentences]
    sent_len = [sentences[i] for i in range(0, len(final)) if final[i]]
    return int(len(sent_len))

#Calculate IDF for each word
idf_score = {}
for each_word in total_words:
    each_word = each_word.replace('.','')
    if each_word not in stop_words:
        if each_word in idf_score:
            idf_score[each_word] = check_sent(each_word, total_sentences)
        else:
            idf_score[each_word] = 1

# Performing a log and divide
idf_score.update((x, math.log(int(total_sent_len)/y)) for x, y in idf_score.items())

#print(idf_score)

#Calculate TF * IDFCalculate TF * IDF
tf_idf_score = {key: tf_score[key] * idf_score.get(key, 0) for key in tf_score.keys()}
#print(tf_idf_score)

#Create a function to get N important words in the document
def get_top_n(dict_elem, n):
    result = dict(sorted(dict_elem.items(), key = itemgetter(1), reverse = True)[:n])
    return result

#Get the top 5 words of significance
pprint(get_top_n(tf_idf_score, 5))
```

Preprocessing / Cleaning Answers

➢ **Libraries**

- NLTK
- Tokenize
- stopwords
- snowballstemmer
- word_tokenize

Implementation

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk import SnowballStemmer
import string

# Method: text_preprocessing
# Input: text
# Output: preprocessed text
def text_preprocessing(text):
    # convert text to lowercase
    text = text.lower()

    # word tokenizing
    tokens = word_tokenize(text)

    # removing noise: numbers, stopwords, and punctuation
    lang_stopwords = stopwords.words("english")
    tokens = [token for token in tokens if not token.isdigit() and \
                            not token in string.punctuation and \
                                token not in lang_stopwords]

    # stemming tokens
    stemmer = SnowballStemmer('english')
    tokens = [stemmer.stem(token) for token in tokens]

    # join tokens and form string
    preprocessed_text = " ".join(tokens)

    return preprocessed_text

# sample text
text = """You can do a git fetch at any time to update your
                    remote-tracking branches under refs/remotes.
                    This operation never changes any of your own local
                    branches under refs/heads, and is safe to do without
                    changing your working copy. I have even heard of people
                    running git fetch periodically in a corn job in the
                    background (although I wouldn't recommend doing this).
                    A git pull is what you would do to bring a local branch
                    up-to-date with its remote version, while also updating
                    your other remote-tracking branches."""

print("The preprocessed text of sample text is:", text_preprocessing(text), sep='\n')
```

Answer Summarization

➢ **Libraries**

- Hugging face Transformers
- T5Tokenizer
- T5ForConditionalGeneration

- Abstractive summarizing is a technique for creating a summary of a text based on its major concepts rather than simply taking the most prominent phrases directly from the text. This is a critical and difficult task in natural language processing.

HUGGING FACE

Implementation

```python
from transformers import T5ForConditionalGeneration, T5Tokenizer

# initialize the model architecture and weights
model = T5ForConditionalGeneration.from_pretrained("t5-base")
# initialize the model tokenizer
tokenizer = T5Tokenizer.from_pretrained("t5-base")

article = """You can do a git fetch at any time to update your
                    remote-tracking branches under refs/remotes.
                    This operation never changes any of your own local
                    branches under refs/heads, and is safe to do without
                    changing your working copy. I have even heard of people
                    running git fetch periodically in a corn job in the
                    background (although I wouldn't recommend doing this).
                    A git pull is what you would do to bring a local branch
                    up-to-date with its remote version, while also updating
                    your other remote-tracking branches."""

# encode the text into tensor of integers using the appropriate tokenizer
inputs = tokenizer.encode("summarize: " + article, return_tensors="pt", max_length=512, truncation=True)

# generate the summarization output
outputs = model.generate(
    inputs,
    max_length=150,
    min_length=40,
    length_penalty=2.0,
    num_beams=4,
    early_stopping=True)
# just for debugging
print(outputs)
print(tokenizer.decode(outputs[0]))
```

References

[1] O. Tas and F. Kiyani, “A survey automatic text summarization,” *Pressacademia*, vol. 5, no. 1, pp. 205–213, Jun. 2017, doi: 10.17261/pressacademia.2017.591.

[2] “The automatic creation of literature abstracts | IBM Journal of Research and Development,” *IBM Journal of Research and Development*, 2021.

[3] E. Yulianti, R.-C. Chen, F. Scholer, W. B. Croft, and M. Sanderson, “Document Summarization for Answering Non-Factoid Queries,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 1, pp. 15–28, Jan. 2018, doi: 10.1109/tkde.2017.2754373.

IT18078992 | Marapana S.K.C.W.K.M.R.T.S.B.

B.Sc. (Hons) Degree in Information Technology Specialized in Software Engineering

Structured-type adaptive questions for knowledge checking, using existing answered questions of the platform and Introduction of a scoring method.

Background/Research Gap

Quizzes are a great way for knowledge-checking on theoretical questions.

Many platforms (HackerRank, LeetCode) just focus on polishing coding skills only and not offer the facility to check knowledge on theoretical questions.

Not getting adaptive or/and reliable answers and not having a proper grading method to measure user answers are few problems that have identified.

Research Question

- How to check theoretical knowledge using structured type questions in more accurate way.
- Adaptive questions on each learner's knowledge level
- Introduce an unbiased scoring method
- How to utilize the questions database of the platform effectively.

Specific and Sub objectives

- Formulating structured type questions for knowledge checking, using existing answered questions of the platform based on user level
- Formulate questions from platform's existing user questions
- Formulate answers from optimal answers and top 4 voted answers and transform user's answer for similarity checking
- Similarity checking using cosine similarity and score assignment

Research Methodology

Methodology

- The use of word embedding frameworks like ***SentenseTransformers*** helps to handle semantic textual similar, semantic search or paraphrase mining. The framework is based on PyTorch and Transformers and offers a large collection of pre-trained models tuned for various tasks.

- WebCrawler for questions scraping
  - If user's desired questions are not available. Relevant set of questions will be scraped from StackOverflow and displayed.

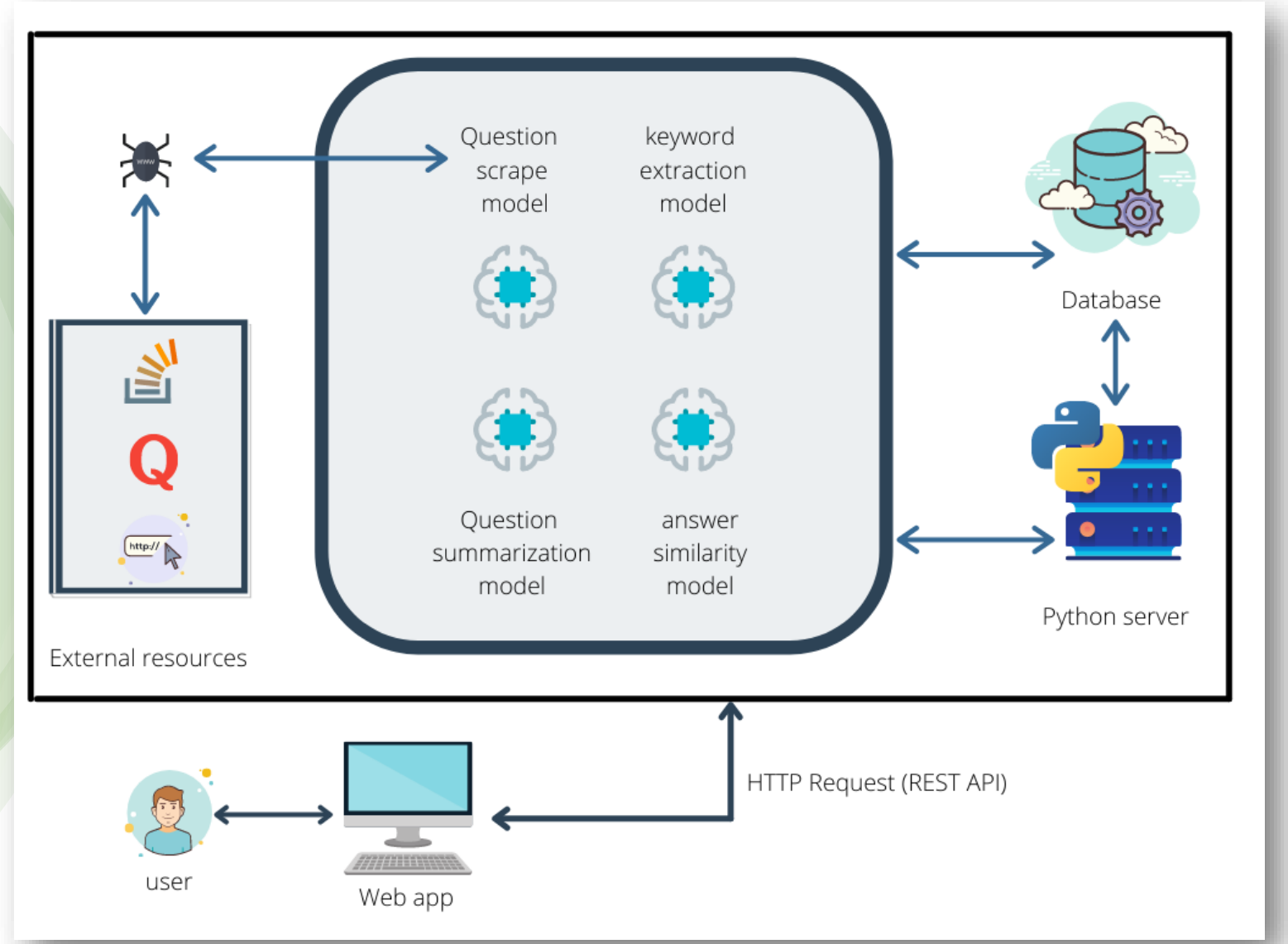
Methodology continued..

- Keyword extraction
  - We need to extract important keywords which helps to formulate quality answers, questions and also for marking method. This task can be achieved with BERT by making a keyword extraction model

- Formulate an unbiased answer
  - We need summarize the user answer so that the answer is having a core meaning despite of the word count

- Similarity checking
  - Cosine similarity can be used in this scenario to determine the similarity level between the user answer and the platform's answer. Two vectors will be checked and based on the cosine similarity score; a score can be given for the answer.
- Marking
  - User answer will be similarity checked with the platform answer using an accurate marking method

System Diagram



Evidence for Completion

Similarity checking between answers

- PyTorch
- Hugging face transformers - BERT pretrained model
  - sentence-transformers/bert-base-nli-mean-tokens

HUGGING FACE

Implementation

```
from transformers import AutoTokenizer, AutoModel
import torch
import pandas as pd

def model_train():
    sentences = [platform_answer, user_answer]

    model_name = "sentence-transformers/bert-base-nli-mean-tokens"

    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModel.from_pretrained(model_name)

    tokens = {'input_ids': [], 'attention_mask': []}

    for sentence in sentences:
        new_tokens = tokenizer.encode_plus(sentence, max_length=128, truncation=True, padding='max_length', return_tensors='pt')
        tokens['input_ids'].append(new_tokens['input_ids'][0])
        tokens['attention_mask'].append(new_tokens['attention_mask'][0])

    tokens['input_ids'] = torch.stack(tokens['input_ids'])
    tokens['attention_mask'] = torch.stack(tokens['attention_mask'])

    tokens['input_ids'].shape

    outputs = model(**tokens)
    outputs.keys()

    embeddings = outputs.last_hidden_state
    embeddings.shape

    attention = tokens['attention_mask']
    attention.shape

    mask = attention.unsqueeze(-1).expand(embeddings.shape).float()

    mask_embeddings = embeddings * mask

    mask_embeddings.shape

    summed = torch.sum(mask_embeddings, 1)
    summed.shape

    counts = torch.clamp(mask.sum(1), min=1e-9)
    counts.shape
```

Implementation continued..

```
    embeddings = outputs.last_hidden_state
    embeddings.shape

    attention = tokens['attention_mask']
    attention.shape

    mask = attention.unsqueeze(-1).expand(embeddings.shape).float()

    mask_embeddings = embeddings * mask

    mask_embeddings.shape

    summed = torch.sum(mask_embeddings, 1)
    summed.shape

    counts = torch.clamp(mask.sum(1), min=1e-9)
    counts.shape

    mean_pooled = summed / counts
    mean_pooled.shape
    mean_pooled

    from sklearn.metrics.pairwise import cosine_similarity

    mean_pooled = mean_pooled.detach().numpy()

    score = 0

    score = cosine_similarity([mean_pooled[0]],mean_pooled[1:])
    final_score = pd.DataFrame(score).to_json('data.json', orient='split')
    #return score
    print(final_score)
    return final_score

model_train()
```

Keyword Extraction

- Hugging face Transformers
  - distilroberta-base
- spaCy
- N-gram

spaCy

HUGGING FACE

Implementation

```python
from sklearn.feature_extraction.text import CountVectorizer
import spacy
from transformers import AutoModel, AutoTokenizer
from sklearn.metrics.pairwise import cosine_similarity

def keywords():
    n_gram_range = (1, 2)
    stop_words = "english"

    # Extract candidate words/phrases
    count = CountVectorizer(ngram_range=n_gram_range, stop_words=stop_words).fit([text])
    all_candidates = count.get_feature_names()

    print(all_candidates[:10])


    #pos tagg
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(text)
    noun_phrases = set(chunk.text.strip().lower() for chunk in doc.noun_chunks)

    nouns = set()
    for token in doc:
        if token.pos_ == "NOUN":
            nouns.add(token.text)

    all_nouns = nouns.union(noun_phrases)

    candidates = list(filter(lambda candidate: candidate in all_nouns, all_candidates))

    print(candidates[:10])


    #model initialize and embedding
    model_name = "distilroberta-base"
    model = AutoModel.from_pretrained(model_name)
    tokenizer = AutoTokenizer.from_pretrained(model_name)

    candidate_tokens = tokenizer(candidates, padding=True, return_tensors="pt")
    candidate_embeddings = model(**candidate_tokens)["pooler_output"]

    candidate_embeddings.shape
```

Implementation continued..

```
    candidate_embeddings.shape

    text_tokens = tokenizer([text], padding=True, return_tensors="pt")
    text_embedding = model(**text_tokens)["pooler_output"]

    text_embedding.shape

    candidate_embeddings = candidate_embeddings.detach().numpy()
    text_embedding = text_embedding.detach().numpy()

    #cosine similarity

    top_k = 10
    distances = cosine_similarity(text_embedding, candidate_embeddings)
    keywords = [candidates[index] for index in distances.argsort()[0][-top_k:]]

    return keywords
    print(keywords)

keywords()
```

User answer summarization

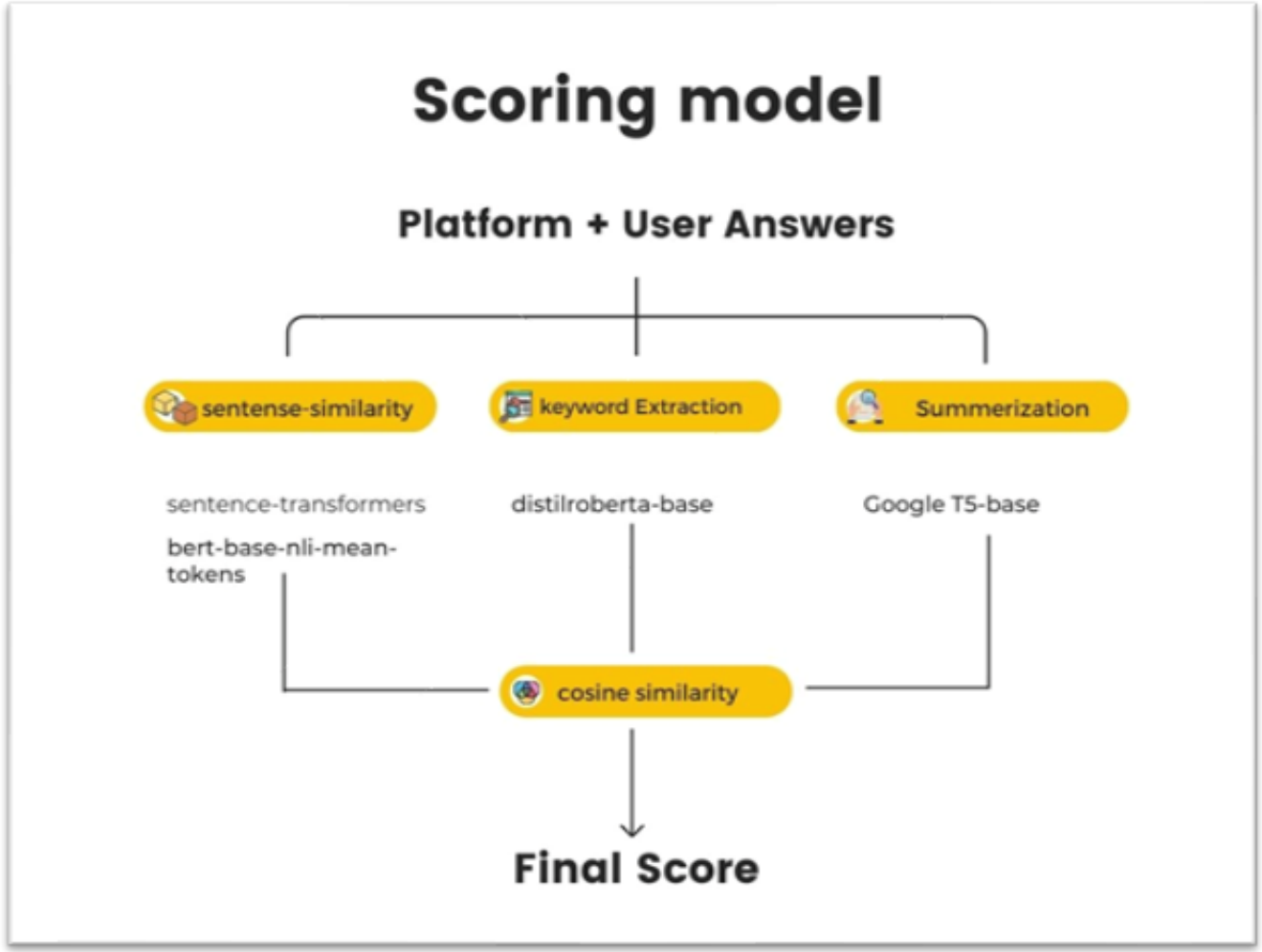
HUGGING FACE

- PyTorch
- Hugging face Transformers
  - T5-base

Implementation continued..

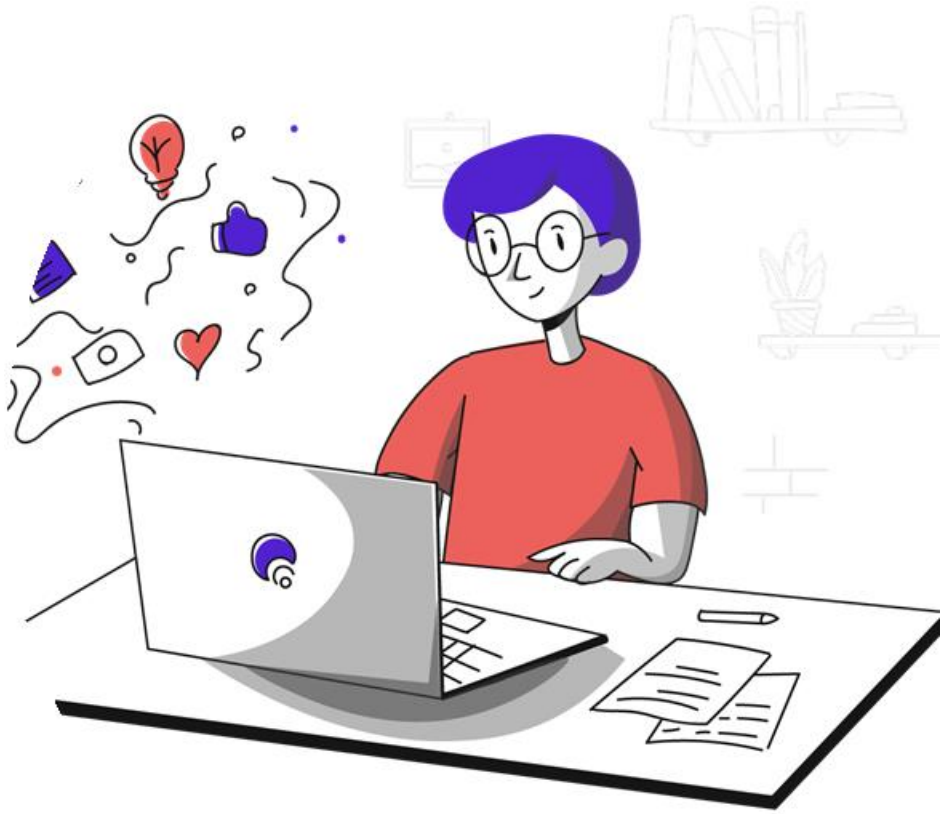
```
import torch
from transformers import AutoTokenizer, AutoModelWithLMHead
from sklearn.metrics.pairwise import cosine_similarity
from transformers import AutoTokenizer, AutoModel
# from sentence_transformers import SentenceTransformer

tokenizer = AutoTokenizer.from_pretrained('t5-base')
model = AutoModelWithLMHead.from_pretrained('t5-base', return_dict=True)

def summerize_answers():
    inputs = tokenizer.encode("summarize platform answer: " + sequence, return_tensors='pt', max_length=512, truncation=True)
    inputs2 = tokenizer.encode("summarize user answer: " + sequence2, return_tensors='pt', max_length=512, truncation=True)
    print(inputs)
    print(inputs2)


    summary_ids = model.generate(inputs, max_length=50, min_length=5, length_penalty=5., num_beams=2)
    summary_ids2 = model.generate(inputs2, max_length=50, min_length=5, length_penalty=5., num_beams=2)

    summary = tokenizer.decode(summary_ids[0])
    summary2 = tokenizer.decode(summary_ids2[0])

    print(summary)
    print(summary2)

    return summary, summary2

summerize_answers()
```

Implementation continued..



Commercialization

Premium Subscriptions for users for Individuals and Organizations for unlimited questions.



References

[1]Devlin, J.; Chang, M.-W.; Lee, K. & Toutanova, K. (2018), 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding' , cite arxiv:1810.04805Comment: 13 pages .

[2] X. Jin, S. Zhang and J. Liu, "Word Semantic Similarity Calculation Based on Word2vec," 2018 International Conference on Control, Automation and Information Sciences (ICCAIS), Hangzhou, China, 2018, pp. 12-16, doi: 10.1109/ICCAIS.2018.8570612.

[3] K. Jayakodi, M. Bandara and D. Meedeniya, "An automatic classifier for exam questions with WordNet and Cosine similarity," 2016 Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 2016, pp. 12-17, doi: 10.1109/MERCon.2016.7480108.

ProbExpert Poster

ProbExpert

An Enhanced Q&A Platform for Reducing Time Spent on Learning and Finding Answers.

Authors
Kamal Thennakoon
Dasun Ekanayaka
Ashen Ranasinghe
Thanura Marapana

Affiliations
Sri Lanka Institute of Information Technology (SLIIT).
Department of Computer Science & Software Engineering.
Malabe,
Sri Lanka.

Prob Expert

Introduction

Information users find on the internet may not be up-to-date due to the rapid pace of change and having to spend less time on the internet for researching and debugging tasks is an added luxury. Having an expertise level while providing answers through a platform is convenient for users, yet when a user signs into a platform, the user must start from the beginning, regardless of the level of competence in the field. Moreover, not having a proper way to evaluate the existing programming knowledge is another obstacle. To address mentioned complications, researchers of this paper have introduced a new e-learning platform- 'ProbExpert' which provides systems in automated answering, optimized answer generation, structured question-based quiz evaluation together with a fully automated portfolio generation.

Objectives

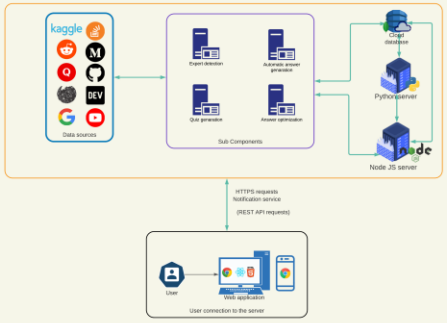
- Develop automatic answer generation system.
- Detecting users' proficiency level along with an auto-generated portfolio.
- Structured type quizzes for knowledge checking and Scoring method.
- Optimal answer generation through up-voted answers.

Results

- A system for automated answer generation has been developed to generate an answer with multiple resources in 10 - 20 seconds on average.
- Structured type theoretical Quizzes from platform questions and introduction of a unbiased scoring method.
- Hiring managers can find the unemployed or free-lancing developers in no time by filtering the leaderboard by using the available to hire tag.Since all the developers are classified into the classes according to the generated Bell curve and listed on the leaderboard according to the total score,
- Optimized answer

System Overview

Expert detection and portfolio generation, automatic answer generation, quiz generation, and answer optimization are the four key components of ProbExpert. Machine learning, deep learning methodologies, natural language processing approaches, and web scraping technologies were used to develop the aforementioned components. ProbExpert has two backend servers, one that uses nodeJS to control user requests and the other that uses Python to control the aforementioned components. Users can gain access to the platform via the web application or the Telegram bot.



Overall System Diagram

Conclusion

The implemented system is able to provide solutions for the problems this research has been addressed, by providing systems for automatic answer generation , detecting user proficiency along with auto generated portfolio, structured type quiz generating and optimal answer generation through upvoted answers. as the future works, models can be further optimized to gain more accuracy. Furthermore, a blockchain system can be applied to introduce a digital currency for transactions within the platform, which can be used to do various tasks, such as hiring an expert for a dedicated session.

Related literature

B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge," in 2013 International Conference on Social Computing, 2013, pp. 188–195, doi: 10.1109/SocialCom.2013.35.

K. Mao, Y. Yang, Q. Wang, Y. Jia, and M. Harman, "Developer recommendation for crowdsourced software development tasks," in Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015, Jun. 2015, vol. 30, pp. 347–356, doi: 10.1109/SOSE.2015.46.

L. Salas-Morera, A. Arauzo-Azofra, and L. García-Hernández, "Analysis of Online Quizzes As a Teaching and Assessment Tool," J. Technol. Sci. Educ., vol. 2, no. 1, 2012, doi: 10.3926/jotse.30.

B. Ross, A. M. Chase, D. Robbie, G. Oates, and Y. Absalom, "Adaptive quizzes to increase motivation, engagement and learning outcomes in a first year accounting unit," Int. J. Educ. Technol. High. Educ., vol. 15, no. 1, pp. 1–14, 2018, doi: 10.1186/s41239-018-0113-2.

S. Onoue, H. Hata, and K. I. Matsumoto, "A study of the characteristics of developers' activities in GitHub," Proc. - Asia-Pacific Softw. Eng. Conf. APSEC, vol. 2, pp. 7–12, 2013, doi: 10.1109/APSEC.2013.104.

J. E. Montandon, L. L. Silva, and M. T. Valente, "Identifying Experts in Software Libraries and Frameworks among GitHub Users Software Developers Expertise View project Software Defects View project Identifying Experts in Software Libraries and Frameworks among GitHub Users."

R. M. Alguliyev, R. M. Aliguliyev, and N. R. Isazade, "Multiple documents summarization based on evolutionary optimization algorithm," Expert Syst. Appl., vol. 40, no. 5, 2013, doi: 10.1016/j.eswa.2012.09.014.

T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," pp. 38–45, 2020, doi: 10.18653/v1/2020.emnlp-demos.6.

2021-155 | E-Learning | Sri Lanka Institute of Information Technology | Department of Computer Science & Software Engineering

Thank you!

Questions?