# AN INTELLIGENT MOBILE APPLICATION TO HELP STUDENTS WITH PROBLEMS BETWEEN THEIR PEERS USING MACHINE LEARNING

Audrey Chen[1], Victor Phan[2]

[1]West Windsor-Plainsboro High School North, 90 Grovers Mill Rd,
Plainsboro Township, NJ 08536
[2]Computer Science Department, California State Polytechnic University,
Pomona, CA 91768

***ABSTRACT***

*Tutoring is not a commonly available service for middle and high school students who often have to learn the toughest subjects in the K-12 education system [1][2]. This can negatively impact their learning and cause losses in opportunities that could have been avoided. In response, we suggest a mobile application that is able to assist students in their learning as well as help them educate others so that all students can learn together [3]. This application uses the Flutter framework as well as Google's Firebase service to store user-generated data [4]. An AI recommender system as well as a subject assignment system is hosted on a Python Flask backend server [5]. It will determine what questions someone should look at, as well as what kind of subject a question seems to be, respectively. These systems rely on an AI that uses natural language processing. To test the effectiveness of our application, we devise two experiments. In the first experiment, we determine the accuracy of our subject assigner by giving it several mock questions. In our second experiment, we compare and contrast different recommender systems. Ultimately, our subject assigner is at 50% accuracy and the recommenders we used were roughly the same in accuracy. Overall, this application is polished and with some more improvements to the backend and AI integration, it will be a useful tool for students who need an extra boost in school.*

***KEYWORDS***

*Education, Database, Natural Language Processing, Machine Learning*

## 1. INTRODUCTION

The purpose of this mobile application is to create an environment where students and their peers can help each other solve problems. The problem this application is trying to solve is the lack of availability for tutoring for many middle school and high school students. According to ChalkBeat, only 1 in 10 students are able to receive tutoring. Many people are not able to receive tutoring either because of the lack of availability in tutors or the affordability for these tutors. Due to this surprising statistic, this app was created in hopes to solve this problem. This mobile application is free to download, meaning students won't have to worry about the cost of getting a tutor. Additionally, it allows more and more students to connect through the internet and collaborate on problems they are having trouble with. This not only helps with problem-solving skills, but it also helps students learn how to collaborate with others on a problem. In the long-run, this mobile application hopes to have an impact on the number of students who can receive a

David C. Wyld et al. (Eds): ICDIPV, CBIoT, ICAIT, WIMO, NC, CRYPIS, ITCSE, NLCA, CAIML -2023
pp. 411-421, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131332

tutor. If more students start to join this application, it creates a larger group of students who can help each other solve problems.

In the first experiment, we try to test the accuracy of our subject assignment system for the application. We sent our server several queries using mock questions to test if it could accurately assign the correct subject. It assigned the correct subject in 50% of questions tested with it. This is a good start, though some improvements are necessary. It got a little confused with STEM related topics but was able to accurately identify language related topics [6]. In the second experiment, we try to test the most effective recommendation system for the application. We tested gensim as well as a combination of scikit learn and NLTK [7][8]. Once again, we come up with 10 or so mock queries to send to the server to see how each library would respond to our queries and what questions it would recommend. Both libraries ended up recommending the same questions. In general our final library, gensim, was preferred slightly over scikit and NLTK.

The solution to the aforesaid problem is creating a community of students and tutors to help each other with problems in subjects they are interested in. This solution solves the problem of the lack of tutors among students because this mobile application is a free to download app and is available to everyone with any type of phone. This means, more students will have access to the application, and therefore, more students will either be able receive or give help. This is an effective solution to allow more students to receive tutoring because it creates a safe environment where students can help each other and collaborate on a problem. Creating a mobile application for students to help each other with problems is more effective than hiring a tutor at times because it not only is more cost-effective, but it also allows for students to work on a problem on their own before asking someone directly for the answer. It allows them to receive help while also improving their ability to solve problems at the same time.

In the first experiment, we try to test the accuracy of our subject assignment system for the application. We sent our server several queries using mock questions to test if it could accurately assign the correct subject. It assigned the correct subject in 50% of questions tested with it. This is a good start, though some improvements are necessary. It got a little confused with STEM related topics but was able to accurately identify language related topics. In the second experiment, we try to test the most effective recommendation system for the application. We tested gensim as well as a combination of scikit learn and NLTK. Once again, we come up with 10 or so mock queries to send to the server to see how each library would respond to our queries and what questions it would recommend. Both libraries ended up recommending the same questions. In general our final library, gensim, was preferred slightly over scikit and NLTK.

## 2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

### 2.1. Get users

One of the challenges that we need to address is getting students and tutors to sign up to our mobile application and regularly use the application. The solution to this challenge could be adding a reward system for those who answer questions correctly. To do this, we could add a feature that allows the creator of the question to mark if an answer is right. Once an answer is marked as correct, the author of the answer would receive volunteer hours. Once someone has reached a certain number of correct answers, they would receive one volunteer hour, and they could continue to increase their volunteer hours by answering more questions. If there is an

incentive for others to answer questions, it is likely that more and more people would want to answer these questions correctly.

### 2.2. Asking questions

Another one of the challenges that we needed to address was regarding asking questions. The first solution to this was creating the Firebase Database to store all the user questions [9]. Whenever a user created a question, the database would store the question under the "Questions" tab, storing the author, time it was asked, the title, content of the question, etc. This would make it easier when accessing these questions to recommend to others. Another solution for the question asking feature would be allowing the user to add images to their question. To do this, the storage section in the firebase database would need to be created. This way, whenever the user adds an image to their question, the image link would be saved in both the database and the storage database. The final solution for the question asking feature is formatting the way the question page looked on the app. To fix this, I would use Containers and the SingleChildScrollView Widget to account for questions that are longer as well.

### 2.3. Getting the recommendation system to work properly

Another challenge that we faced was getting the recommendation system to work properly. To do this, we would have to first gather data about the user's interests and the questions they have recently looked at. This data would then be given to the python code. The recommendation program would obtain key words from this data, and give out the most similar questions based on the user's interests. Although the recommendation system was difficult to get working at first, more testing and a few changes helped progressively make the recommendation system more accurate. When the user goes on the question page, it will give out questions that the user would most likely be interested in based on their recent activity.

## 3. SOLUTION

The program starts with the user either logging into their account or creating an account if they don't have one. After that, the program will lead them into their profile page for their account. In their profile page, the user will have the option to go to their history page and look at questions they have asked or answered, or they can go to their Questions page, where they can see suggested questions for them to answer. If the user chooses to navigate to their Questions page, the user will be suggested certain questions through a recommendation system based on the user's interests and recent activity. The user can click on a question tile, which will lead them to a page with the details about that question. In that question page, the user can look at replies to that question, and can reply to it as well. Other than choosing to click on a question tile in the Questions page, the user can also click a button at the top of the page to create a new question. The program will then be able to figure out what subject that question is based on the content, and then recommend it to those who are interested in those types of questions. Finally, on the Questions page, the user can search through questions if they are looking for something specific.

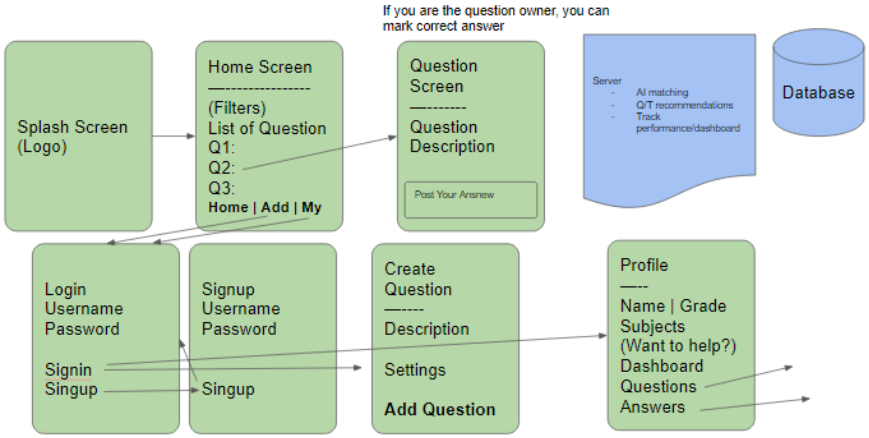



Figure 1. Overview of the solution

The first component is question querying. This component displays and recommends certain questions for the user based on recent activity and interests, or based on what they search up. Querying questions does rely on a server that allows the program to make accurate recommendations for the user.

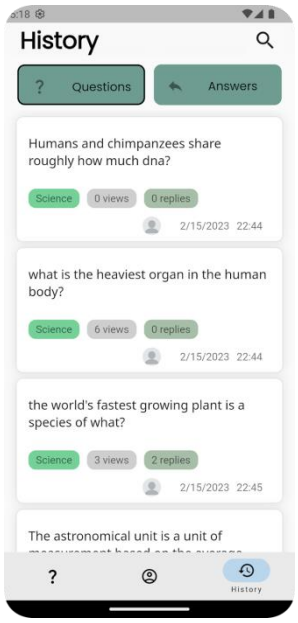



Figure 2. Screenshot of history

```
Future<void> queryAllQuestions() async {
  await FirebaseDatabase.instance.ref().child("Questions").once().
  then((result) {
    var info = result.snapshot.value as Map;
    setState(() {
      var keys = info.keys.toList();
    });
    print(info);
    addAllQuestions(info);
  }).catchError((error) {
    print("could not get question info $error");
  });
}

///Gets all questions with a similarity score to the filter over zero. Occurs only when a filter is nonempty.
Future<void> queryWithFilter() async {
  String url = "https://Tutor-AI-Server.bigphan.repl.co/recommend/$filter";
  final uri = Uri.parse(url);
  final response = await http.get(uri);
  var responseData = json.decode(response.body);
  print(responseData);
  for (int i = 0; i < responseData.length; i++) {
    Question questionToAdd = await getQuestionInfo(responseData[i].toString());
    setState(() {
      questions.add(questionToAdd);
    });
  }
}

///Updates the questions list by referencing the Firebase Realtime Database.
Future<void> getQuestions() async {
  questions = [];

  if (filter.isNotEmpty) {
    print("searching!");
    await queryWithFilter();
    return;
  }

  var uuidList = await getUUIDsForRecentlyCommentedOnQuestions();
  if (uuidList.isEmpty) {
    print("no comments made! querying everything");
    await queryAllQuestions();
  }
  else {
    print("comments found! recommending");
    await queryWithRecommendation(uuidList);
  }
```

Figure 3. Screenshot of code 1

When the user navigates to the Questions page, the program will call the getQuestions() method, which recommends certain questions to the user. First, this method checks if the user just searched something up, because when they press the enter button, this method is also called. If the user is searching something up, then the queryWithFilter() method is called. This method takes in the user's input and makes a call to the server. The server will then return a list of questions that best match the user's search, which is then displayed for the user on this page. If the user has not searched something up, the program checks the user's history page to see if they have answered any questions yet. The user's answered questions history will allow the program to determine the user's recent interests. If the user has not commented on any questions thus far, then the program will display all questions in the system for the user. If the user has answered a few questions, then the program will make a call to the server, and display questions that are similar to the questions that the user has answered recently.

The next component is the comments component. For each question that is created, the user is able to comment on the question. For each comment, users are also allowed to reply to comments, thus creating an additional comment system within each comment. In each comment, the user is allowed to reply, like, dislike, or favorite each comment.

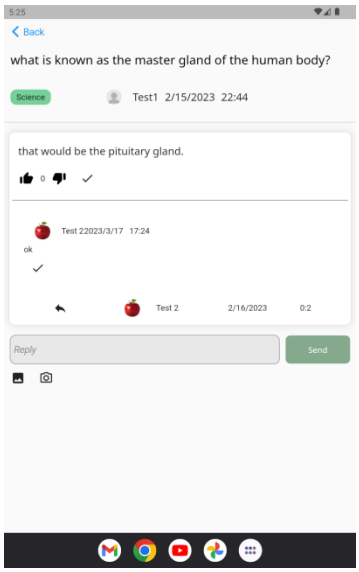



Figure 4. Screenshot of qestion and answer

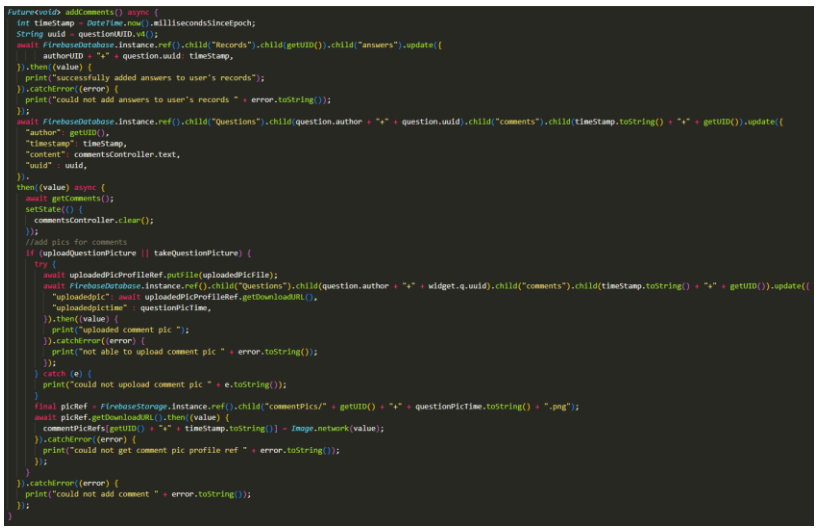
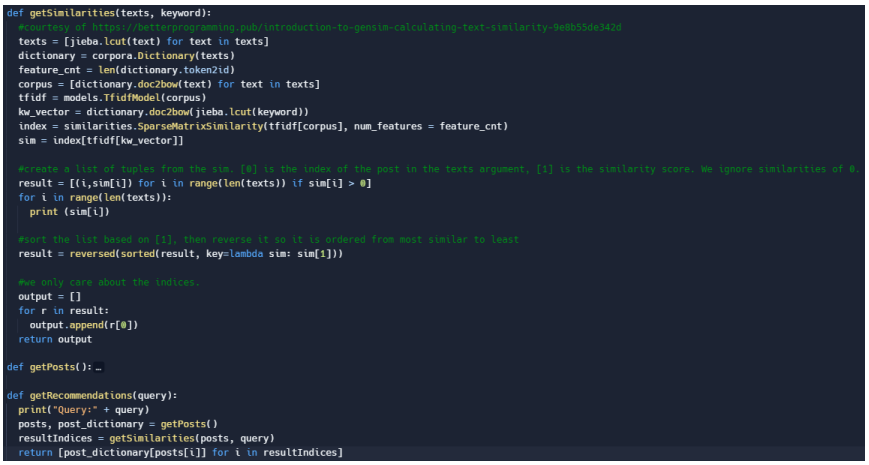


Figure 5. Screenshot of code 2

When adding a comment, the program first adds it to firebase. This then creates a new comment under the “comments” section in firebase. When each comment is created, an author, time stamp, content, and uuid property is created, and this data is also stored in the firebase database. After the comment is added to firebase, the program checks if the user uploaded or took a picture when creating the comment. The program is able to do this by checking if they have pressed the add image button, and if they have chosen an image as well. If so, then these properties are added to the comment in the firebase database and the storage database as well. This comment is also added to a list in the program that keeps track of all the comments for each question, which is then displayed when the user clicks on the question tile for details.

The third component is the post recommendation component. The recommendation system in the program suggests the user questions to answer when they navigate to the Questions page. This system is based on the current questions in the database already, and then recommends questions that are similar to what is inputted into the server.



```
def getSimilarities(texts, keyword):
  #courtesy of https://betterprogramming.pub/introduction-to-gensim-calculating-text-similarity-9e8b55de342d
  texts = [jieba.lcut(text) for text in texts]
  dictionary = corpora.Dictionary(texts)
  feature_cnt = len(dictionary.token2id)
  corpus = [dictionary.doc2bow(text) for text in texts]
  tfidf = models.TfidfModel(corpus)
  kw_vector = dictionary.doc2bow(jieba.lcut(keyword))
  index = similarities.SparseMatrixSimilarity(tfidf[corpus], num_features = feature_cnt)
  sim = index[tfidf[kw_vector]]

  #create a list of tuples from the sim. [0] is the index of the post in the texts argument, [1] is the similarity score. We ignore similarities of 0.
  result = [(i,sim[i]) for i in range(len(texts)) if sim[i] > 0]
  for i in range(len(texts)):
    print (sim[i])

  #sort the list based on [1], then reverse it so it is ordered from most similar to least
  result = reversed(sorted(result, key=lambda sim: sim[1]))

  #we only care about the indices.
  output = []
  for r in result:
    output.append(r[0])
  return output

def getPosts(): …

def getRecommendations(query):
  print("Query:" + query)
  posts, post_dictionary = getPosts()
  resultIndices = getSimilarities(posts, query)
  return [post_dictionary[posts[i]] for i in resultIndices]
```

Figure 6. Screenshot of code 3

When the server is called for a recommendation, the program gives an HTTP request to the server to ask for recommendations for questions [10]. To do this, the program first looks at the user's recent activity to determine what kind of questions they would be interested in. The question and content are given to the server as one string, and it is then broken down into its key words by removing all unnecessary words like "the", "to", etc. This allows the AI to recognize a general idea of the question, and therefore allow it to match it with other questions currently in the database [10]. Once the call to the server is completed, the AI is able to tell how closely matched questions in the database are with the user's most recent activity. In the end, the server will output a list of questions to recommend to the user, which is given back to the program to display for the user.

## 4. EXPERIMENT

### 4.1. Experiment 1

For this experiment, we will be testing if the suggestion recommendation algorithm for our backend server is able to consistently and accurately assign the correct subject automatically given a question. We want to test this so that questions in the app are categorized correctly.

For this experiment, we will be using a set of 10 experiment mock questions to query the server with. The server will send back a response which details which subject it thinks the question is associated with.

- What is the limiting reactant in the process of making copper chloride?
- What muscle helps with movement in your heel and upper ankle joint?
- How many feet are in a mile?
- How many liters of a 55 percent salt solution must be added to a 10 percent salt solution tomake this solution a 30 percent salt solution?
- What alliance was the US part for WWI and WWII?
- What were the main factors for the colonists arriving in North America?
- What are the similarities between English and Spanish?
- Why is it called "Día de los Reyes Magos"?
- What is unique about Shakespeare's writing?
- What is a theme in Little Women?

There are 5 subjects the server can identify a question as. In order of the mock questions, they are: Science, Math, History, Language, and English. Since there are ten questions, every subject will have two questions respectively. We will after the experiment see which subjects had the highest accuracy and what the overall accuracy was.

| Question | Response | Correct? |
|---|---|---|
| What is the limiting reactant in the process of making copper chloride? | History | Wrong |
| What muscle helps with movement in your heel and upper ankle joint? | History | Wrong |
| How many feet are in a mile? | History | Wrong |
| How many liters of a 55 percent salt solution must be added to a 10 percent salt solution to make this solution a 30 percent salt solution? | Math | Correct |
| What alliance was the US part for WWI and WWII? | History | Correct |
| What were the main factors for the colonists arriving in North America? | English | Wrong |
| What are the similarities between English and Spanish? | Language | Correct |
| Why is it called "Día de los Reyes Magos"? | Language | Correct |
| What is unique about Shakespeare's writing? | English | Correct |
| What is a theme in Little Women? | History | Wrong |

Figure 7. Figure of experiment 1

In the results of this experiment, there were many cases in which the system was not accurately able to tell the subject of a question, while there were also many cases where the program could tell; in total the accuracy was 50%. The subject with the lowest accuracy was science, with a 0%, while the subject with the highest accuracy was Language. The most common subject the server responded with was History. The reason for this could be that the recommendation system is not advanced enough to tell the difference between these questions sometimes. Also, many of these questions are worded similarly and have similar structures, which would make it harder to tell the difference between subjects. So, the main reason for this low accuracy could be that the program can't tell distinct traits about a subject because of how similar these questions are in terms of structure. However, there were still many cases in which the system was able to accurately tell what subject a question was.

## 4.2. Experiment 2

This experiment tested whether the Gensim recommendation was the most accurate and best recommendation system compared to others. We checked this to make sure tutors get recommended questions closest to their interests.

To carry out this experiment, an alternate recommendation system was set up to compare to the Gensim recommendation system. The other recommendation system used in this experiment was Scikit, which uses NLTK as a package to analyze the data. First, the recommendation program using Scikit was created to determine the similarities between the existing questions in the database and the user's interest. After both the recommendation programs using Gensim and

Scikit were completed, the experiment was carried out using 5 different queries. The queries used for this experiment were: “planet”, “human body”, “dna”, “artery”, and “war”. These queries were used because there were multiple questions in the database that contained these keywords, so the experiment would be able to show which question it thought was the most similar to another.

| | Gensim | NLTK + SciKit |
|---|---|---|
| Planet | - What is the largest planet?<br>- What planet is furthest from the sun?<br>- What planet is closest to the sun?<br>- What is the only planet that spins clockwise? | - What is the largest planet?<br>- What planet is closest to the sun?<br>- What planet is furthest from the sun?<br>- How many planets in our solar system have moons?<br>- that would be uranus. |
| Human Body | - What is the heaviest organ in the human body?<br>- What is known as the master gland of the human body?<br>- In what year did the Apollo 7 human spaceflight take place? | - What is the heaviest organ in the human body?<br>- that would be the pituitary gland.<br>- Humans and chimpanzees share roughly how much DNA?<br>- It is between the Earth and the Sun. |
| DNA | - When was DNA discovered?<br>- Humans and chimpanzees share roughly how much dna? | - When was DNA discovered?<br>- Humans and chimpanzees share roughly how much dna? |
| Artery | - biology (description had the word “artery”)<br>- biology question (description had the word “artery locations”) | - biology (description had the word “artery”)<br>- biology question (description had the word “artery locations”) |
| War | - What caused the civil war?<br>- when did the war of 1812 start | - What caused the civil war?<br>- when did the war of 1812 start |

Figure 8. Figure of experiment 2

In the end, both programs gave its results for which question it thought was most similar to the user’s interest. In general, both recommendation programs gave the same results for each query, like “artery”, and “war”. However, for the other three queries, the programs gave different results. When the “planet” query was tested, the two recommendation programs gave the same questions, but in different orders of which it thought was most similar. However, for the other two queries (“dna” and “human body”), the recommendation programs gave different results. Additionally, for both recommendation systems, when the “dna” query was tested, it was important that “dna” was all lowercase and not uppercase. If it was uppercase, then both systems would not recognize it as similar to any of the questions in the database. Ultimately, both systems were able to recognize the similarity of all the questions compared to the user’s interest, and both also required that the “dna” query be all lowercase. So, both systems equally work the same and generally give the same results.

## 5. RELATED WORK

The SmartTutor, created by School of Professional and Continuing Education, The University of Hong Kong, is an online tutor app, developed to help those who need it [12]. According to this article, the application mainly focuses on the “personalization and intelligent tutoring” to make their app unique and different from others. This is similar to BRYT Minds, because we also focus on personalization to make the experience better for the user. A recommendation system is created, which gathers data about the user’s recent activity on the app, and determines the user’s interests based on that. Then, the program recommends questions to the user based on their

interests. This personalized experience allows users to see questions they are more interested in. SmartTutor is a little different from BRYT Minds, because SmartTutor was designed to distance learning from far away. While BRYT Minds can also be used for that purpose, this application was initially created so that students within schools or tutors could help each other after school.

This Intelligent Tutoring System was created by Professors at the Irsaa University [13]. This tutoring system was created to teach more students about technology because of the rapid technological developments that have been occurring recently. This tutoring system is similar to my application because it focuses on teaching other students about a certain subject that they might be interested in. This is a little different from BRYT Minds because they mainly focus on computer science and artificial intelligence. However, the BRYT Minds application focuses more on a variety of subjects for all the students who need help in those subjects.

The Intelligent Tutoring System described in this paper was created because the authors realized that traditional learning in classrooms was no longer as effective as it used to be [14]. So, this tutoring system allows students to learn material online, which the authors believe to be better and more effective when learning new material. This is similar to the BRYT Minds application because BRYT Minds also focuses on pushing the learning experience online. We believe that this can be more effective when learning because your teachers may not be available when you are home. However, this is a little different from BRYT Minds, because this tutoring system focuses on having the teachers act as tutors. However, BRYT Minds focus more on having students and peers tutor each other.

## 6. CONCLUSIONS

One of the limitations to this project is that some people might not want to use it because they would rather information come from a more trustworthy source rather than their peers, who could potentially be wrong. They also may believe that tutoring one-on-one is more effective. However, we have tried to address this by creating incentives, such as volunteer hours and points, to entice more people to join and use this application. Also, the peer voting on each question reply could make others feel more at ease because they are more guaranteed that the reply is correct. If I had more time with this project, I would probably try to polish the application a little better and make sure that there are no small errors that potentially need to be fixed. Also, I would try to make the notification feature work a little better, so that students not only get a notification when a question that they would be interested in gets posted, but they would also get notified if someone has replied to their question.

Overall, I think this app was a good experience to go through. At the beginning of this, I was not very familiar with the Dart programming language and the Flutter widgets [15]. However, with some help, I was able to quickly learn how to become fluent in these languages. Now, I have more experience with creating an application, and I hope to use it more often in the future whenever I need to.

## REFERENCES

[1] VanLehn, Kurt. "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems." Educational psychologist 46.4 (2011): 197-221.

[2] Thornburg, David D. "Technology in K-12 Education: Envisioning a New Future." For full text: http://www. air. org/forum/forum. htm. For full text: http://www. air. org/forum/abthornburg. htm., 1999.

[3] Islam, Rashedul, Rofiqul Islam, and Tohidul Mazumder. "Mobile application and its global impact." International Journal of Engineering & Technology 10.6 (2010): 72-78.

[4] Chatterjee, Nilanjan, et al. "Real-time communication application based on android using Google firebase." Int. J. Adv. Res. Comput. Sci. Manag. Stud 6.4 (2018).
[5] Singh, Mandeep, et al. "Implementation of database using python flask framework." International Journal of Engineering and Computer Science 8.12 (2019): 24890-24893.
[6] McIntyre, Miranda M., Jessica L. Gundlach, and William G. Graziano. "Liking guides learning: The role of interest in memory for STEM topics." Learning and Individual Differences 85 (2021): 101960.
[7] Bisong, Ekaba, and EkabaBisong. "Introduction to Scikit-learn." Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners (2019): 215-229.
[8] Bird, Steven. "NLTK: the natural language toolkit." Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions. 2006.
[9] Moroney, Laurence, and Laurence Moroney. "The firebase realtime database." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform (2017): 51-71.
[10] Schechter, Stuart, Murali Krishnan, and Michael D. Smith. "Using path profiles to predict HTTP requests." Computer Networks and ISDN Systems 30.1-7 (1998): 457-467.
[11] Brynjolfsson, Erik, and A. N. D. R. E. W. Mcafee. "Artificial intelligence, for real." Harvard business review 1 (2017): 1-31.
[12] Cheung, B., et al. "SmartTutor: An intelligent tutoring system in web-based adult education." Journal of Systems and Software 68.1 (2003): 11-25.
[13] Hasanein, Hasan A. Abu, and Samy S. Abu-Naser. "Developing education in israa university using intelligent tutoring system." (2018).
[14] Qwaider, SabreenRefat, and Samy S. Abu-Naser. "Excel intelligent tutoring system." (2018).
[15] Boukhary, Shady, and Eduardo Colmenares. "A clean approach to flutter development through the flutter clean architecture package." 2019 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 2019.