AN INTELLIGENT MOBILE APPLICATION TO FACILITATESTUDENT'S NETWORKING USING NATURAL LANGUAGE PROCESSING ALGORITHMS

Ruijin Deng[1] and Victor Phan[2]

[1]Shawnigan Lake School, 1975 Renfrew Rd, Shawnigan Lake, BC V0R 2W1, Canada
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

This paper tackles the prevalent problem of students facing resource limitations in pursuing their passions, often dueto the lack of like-minded peers within their school environment. To address this issue, we propose the development of a team-management app that serves as a transformative platform, enabling students to connect with others who share their interests and aspirations [1]. The application contains three components: the team management system, the search system, and the chat system. Throughout the development process, we encountered various challenges, such as ensuring user privacy. Comparative analysis with three alternative solutions demonstrated several notable advantages, including an intuitive and user-friendly interface, granting students full autonomy over their clubs, and being open to all students from diverse club interests. This innovative application has the potential to empower students, foster collaboration, and provide a valuable resource for educational institutions and students alike [2].

KEYWORDS

Students Networking, Natural Language Processing Algorithms, Mobile Application, Social Networking

1. INTRODUCTION

The issue that is addressed in this research is students lacking in-school connection with like-minded peers [3]. The background and history of this problem can be traced back to the traditional education system, where students are often placed in classrooms based on their age and location, rather than their interests and passions [4]. This issue is crucial to address because it acts as an obstacle in a student's journey of pursuing their passion. It is also important to address because it helps students shape their identity and increase their confidence by sharing their interests with like-minded peers. In the long run, addressing this problem can lead to more engaged and motivated students, which ultimately contribute to a more inclusive and nurturing educational environment [5].

Of the 3 methodologies analyzed in section 5, “Club Activity Management System" created by Kyouhei Higashi mainly focused on improving the student's autonomy in sports management and reducing unnecessary co-curriculum stress. However, because of its sport-focused, it has a slightly

David C. Wyld et al. (Eds): ICCSEA, NLAI, SCAI, CSIA, IBCOM, SEMIT, NECO, SPPR, MLDS -2023
pp. 77-91, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.132407

narrower diversity. The “Ikka Veima Football Club Content Management System" created by Prof. Kari Systä focused on managing the football club with a precise system that involves 3 approaches, which may lead to excessive complexity and lack of analysis. In addition to the project involvinga Sports Club Intranet, created by Oriol González Navarro that utilizes a database system for centralizing data t. This solution has some limitations in managing the database. StarSeek improves on these works by providing a platform for all types of clubs, an easy-to-use user interface, and a strong database stored in Firebase.

Clubs nowadays can be managed in various methods, which are not foolproof [6]. They can be managed through private group chats such as Snapchat or Discord [7]. They can also be maintained through Facebook Groups or public-facing Instagram Accounts. The downside of using these systems, however, is that they are not professional (especially when done via a group chat), they aren’t organized, and often students either wouldn’t otherwise regularly use these apps, or these apps don’t have a student market in mind. We propose an application named StarSeek, that will serve as a club-focused app for a student-only market. StarSeek provides a students-only platform that is academic-focused, which could enhance the connection among students while keeping it trustworthy. This solution is effective because its target market is students in North America; and because the data is verified by the schools, it is very secure in terms of student’s personal information.

The first experiment that we conducted was a survey to test user satisfaction with using StarSeek, and also if users would consider using StarSeek as a legitimate application in their school life. We asked a variety of questions about StarSeek’s user interface as well as its perceived utility. Respondents gave generally fair to positive feedback regarding StarSeek as an application. We believe that improving StarSeek as an app and getting a larger user base would improve these survey scores.

The second experiment that we conducted was an accuracy test between two natural language recognition models to determine which model we would use as our recommender system in the actual StarSeek application. We tested 10 cases against these models (cosine and gensim, respectively), with a dataset of 16 realistic-sounding team descriptions. After testing, we determined that the cosine model had by a large margin the most accurate recommendations, and thus we utilize cosine similarity within the production application.

2. CHALLENGES

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

2.1. To Ensure Data’s Accuracy

One challenge that comes with implementing the accounts to be used within teams is to ensure that the data associated with each account is accurate. A student may be able to falsify data on their account and act like they are part of a club or school that they aren’t affiliated with. Schools that affiliate themselves with StarSeek would be able to use their administrative powers to enforce policies for students to ensure truthfulness. Otherwise, schools may request us (the app’s developers) to delete fraudulent accounts in the database for them. Another concern is that a student applies to join a team that does operations that requires its members to be closeby. To address this, managers for a team are able to deny requests to join and can look through the profiles of applicants before approving or denying them.

2.2. Present Features

One design consideration when it came to the chat system was if features present in other chatting applications such as sending stickers, images, videos, or gifs should be allowed. We decided that for this application, having members only be able to send over text to one another would be the safest option. This is because it allows for conversations had through the application to be more sanitized and more focused on direct communication between members. Giving members the ability to send images and videos to one another may encourage more casual conversations and less productive coordination between management and student members

2.3. How To Ensure Good Results

One of the biggest concerns that come with our AI machine-learning team search algorithm is how to ensure the best and most accurate results when performing queries [8]. There is no guarantee that the recommender system in this application will be the most robust or the most accurate, but we strive to maintain the best quality that we can achieve. If we find through testing that our recommender system is not as accurate as we believe it should be, our backend infrastructure is flexible enough for us to be able to easily swap it out with a different recommender system if we happen to find a better one.

3. SOLUTION

Users begin their journey on the sign-in/sign-up page where they can either log in or create a new account. After successfully logging in, they are directed to the questionnaire page to provide essential information, such as their preferences and interests. Once completed, users land on their dashboard, the central hub of the platform. From the dashboard, they can create and manage teams, and previously created teams are conveniently listed for reference. Clicking the "Make Team" button on the dashboard leads users to the team creation page, where they can define the team's title and description. Additionally, users can explore existing teams by navigating to the team search page, where they can apply to join a team of interest. Team owners are promptly notified of applications, and they can accept or reject new members. The chat feature allows for one-on-one or group communication by email or within existing teams. Lastly, users can view and personalize their profiles on the user profile page, and they can update their personal information in profile settings. This comprehensive design caters to a social and collaborative platform, ensuring a seamless experience for users to create, connect, and communicate effectively.

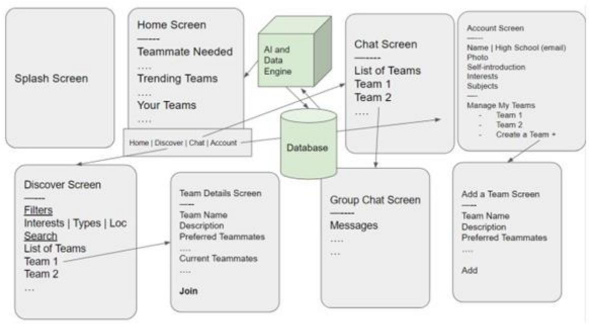


Figure 1. Overview of the solution

One of the systems in my app is an AI system [9]. Specifically, this is a natural language processing AI, which adapts the use of algorithms to be able to interpret natural languages such as English and extract connections and meanings from human speech. In this app, the AI is used as a recommendation system which was incorporated in the search feature of the app to give users relevant teams to join.

Figure 2. Screenshot of the project 1

```python
def cosine_similarity_score(sentence1, sentence2):
    # Tokenize and convert sentences to lowercase
    sentence1 = sentence1.lower()
    sentence2 = sentence2.lower()

    # Initialize CountVectorizer to convert a collection of text documents to a matrix of token counts
    vectorizer = CountVectorizer().fit_transform([sentence1, sentence2])

    # Compute cosine similarity
    vectors = vectorizer.toarray()
    similarity = cosine_similarity(vectors)

    # Cosine similarity is the element at (0, 1) or (1, 0)
    similarity_score = similarity[0][1]
    return similarity_score

def get_recommendation(word, sentences):
    output = {}
    for s in sentences:
        output[s] = cosine_similarity_score(word, s)
    output = dict(sorted(output.items(), key=lambda x:x[1], reverse=True))

    return output.keys()

def recommend_team(dict_of_teams, query):
    result_strings = get_recommendation(query, dict_of_teams.keys())
    output = []
    for result in result_strings:
        output.append(dict_of_teams[result])
    return output
```

Figure 3. Screenshot of code 1

For a cosine similarity model, the accuracy score is determined by the cosine between the vectors of two given strings. In the screenshot, under the method cosine_similarity_score, we take in two strings that are to be compared to one another, named sentence and sentence2. We utilize a count vectorizer to convert these strings into vector representations. We then compare the cosine of the angle between these two vectors in order to determine the similarity score. In the method get_recommendation, we take in a query, and a set of string to compare our query to. We will calculate the similarity score for each string in our dataset, sort our dataset by the strings with the highest similarity, and then return an ordered list of our recommendations. In the method recommend_team, we take in a dictionary of teams, mapped from description to title, and a string that is our query. We will get a recommendation list based on these two parameters, and then return the names of the teams in order of that team's similarity to our query.

The chat system in my app functions as a communication tool between users to effectively interact and exchange messages with one another [10]. Its primary purpose is to facilitate real-time communication and enable users to send text-based messages. This function relies on Firebase's Firestore, where the chat logs and chat metadata are stored. Real-time databases can allow for instant updates and synchronization for the messages that users sent.

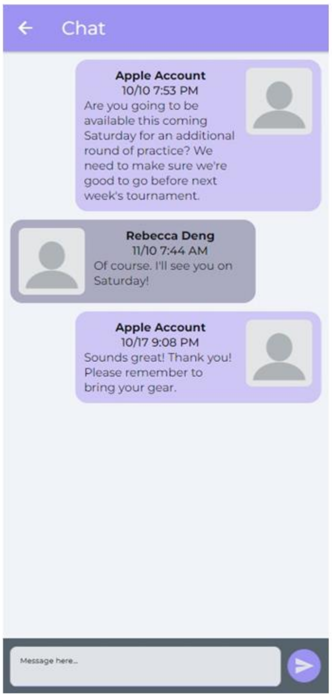


Figure 4. Screenshot of project 2

```
onPressed: () async {
  var chatsRecordReference = ChatsRecord.collection.doc();
  await chatsRecordReference.set({
    ...mapToFirestore(
      {
        'members': _model.groupMembers,
      },
    ),
  });
  _model.createdDocument = ChatsRecord.getDocumentFromData({
    ...mapToFirestore(
      {
        'members': _model.groupMembers,
      },
    ),
  }, chatsRecordReference);
  await _model.createdDocument!.reference.update({
    ...mapToFirestore(
      {
        'members': FieldValue.arrayUnion([currentUserReference]),
      },
    ),
  });
  var invitesRecordReference = InvitesRecord.collection.doc();
  await invitesRecordReference.set({
    ...createInvitesRecordData(
      sender: currentUserReference,
      chatRef: _model.createdDocument?.reference,
    ),
    ...mapToFirestore(
      {
        'recipients': _model.groupMembers,
        'members': _model.groupMembers,
      },
    ),
  });
  _model.createdInvite = InvitesRecord.getDocumentFromData({
    ...createInvitesRecordData(
      sender: currentUserReference,
      chatRef: _model.createdDocument?.reference,
    ),
    ...mapToFirestore(
```

```
      {
        'recipients': _model.groupMembers,
        'members': _model.groupMembers,
      },
    ),
  }, invitesRecordReference);
  await _model.createdInvite!.reference.update({
    ...mapToFirestore(
      {
        'members': FieldValue.arrayUnion([currentUserReference]),
      },
    ),
  });
  await UserChatsRecord.collection.doc().set(createUserChatsRecordData(
        user: currentUserReference,
        chatref: _model.createdDocument?.reference,
      ));
  context.safePop();
  setState(() {});
},
```

Figure 5. Screenshot of code 2

The codes use several Firestore collections. The first one is called chats, where all of the group chats that people make are stored. The second one is invites, which are documents that contain invites to join the group chat, containing the group chat's creator, the members, and the recipients. When someone accepts an invitation, they would be removed from the recipients' list. The third one is chat_refs, where a document is made in this collection when someone accepts an invitation and serves as their key for accessing the group chat. When someone decides to make a group chat, the collection creates a chat document and sets that chat's members. They then add the creator to the chat's members and create an invite document, setting the invitation's members list, sender, and recipients. The creator is then added to the invite document's members list. Then the chat_ref document is created so that the chat creator can access their own chat page. Then we go back to the previous page.

The team system is the biggest part of this app, where users can make teams, join teams, and manage teams. This component serves as the foundation of the app's function, where all the connections are based. Users can initiate the creation of new teams, manage team membership by deciding who can join or leave the team, and accept or decline invitations. This component is based on Firebase's Firestore, where all the structured data such as user-generated content for managing teams and their associated data [15].

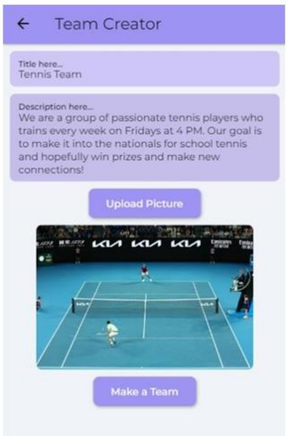


Figure 6. Screenshot of project 3

```
// Generated code for this Button Widget...
if ((_model.textController1.text != null &&
        _model.textController1.text != '') &&
    (_model.textController2.text != null &&
        _model.textController2.text != '') &&
    (_model.uploadedFileUrl != null && _model.uploadedFileUrl != ''))
  FFButtonWidget(
    onPressed: () async {
      var chatsRecordReference = ChatsRecord.collection.doc();
      await chatsRecordReference.set({
        ...mapToFirestore(
          {
            'members': [currentUserReference],
          },
        ),
      });
      _model.groupChat = ChatsRecord.getDocumentFromData({
        ...mapToFirestore(
          {
            'members': [currentUserReference],
          },
        ),
      }, chatsRecordReference);
      await UserChatsRecord.collection.doc().set(createUserChatsRecordData(
            user: currentUserReference,
            chatref: _model.groupChat?.reference,
          ));
      await TeamsRecord.collection.doc().set({
        ...createTeamsRecordData(
          description: _model.textController2.text,
          owner: currentUserReference,
          title: _model.textController1.text,
          image: _model.uploadedFileUrl,
          groupChat: _model.groupChat?.reference,
        ),
        ...mapToFirestore(
          {
            'members': [currentUserReference],
            'admins': [currentUserReference],
          },
        ),
      });
```

Figure 7. Screenshot of code 3

The application contains a page for a user to create a team, by filling out a form containing the team's name, description, and banner picture. When the user presses the "create" button, this code is run. First, the group chat for the team needs to be generated, so we begin by creating a document in the chats collection, with the team creator as its only member. After that, a user_chats document is generated so the team creator can access their own group chat.For the final step, a team document is created that sets the description, title, and image to be what the owner filled out in the original form. We set the owner, members, and admins field of the document to be just the owner's user ID. Lastly, we set the group_chat field to be the generated group chat document from step 1.

4. EXPERIMENT

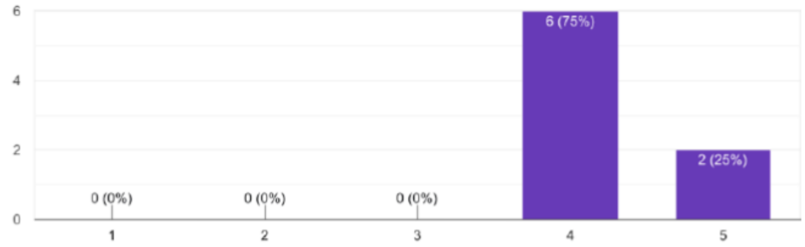
4.1. Experiment 1

This experiment is conducted through a survey. In this experiment, there are three unknown objectives that are to be determined. First, through this experiment, user satisfaction are being measured from the survey. Second, the utility of this app will be perceived. Lastly, the utility of certain features is also being determined from the survey.

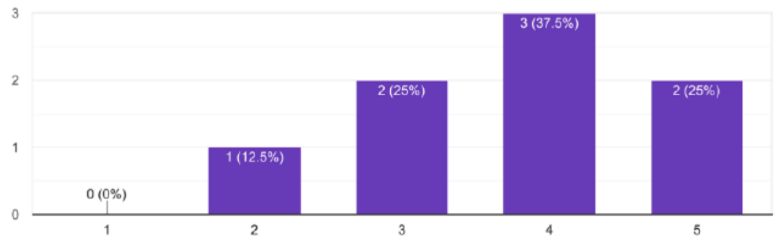
This experiment is set up through a Google survey that is sent out to a group of students in a Canadian boarding school with a demo instruction beforehand. Some of the participants will be regular students, some will be club members, and others will be club presidents. The form has the following questions.

1. On a scale of 1-5, how easy was it to navigate through StarSeek?
2. On a scale of 1-5, how clear was StarSeek's design?
3. On a scale of 1-5, how likely are you going to use StarSeek to find someone outside of school?
4. On a scale of 1-5, how effective do you think StarSeek would be at bridging connectionsbetween schools?
5. If you were a member of a club at your school, on a scale of 1-5, would you suggest to yourclub leader to create a team on StarSeek?
6. StarSeek contains an in-app group chat system. On a scale of 1-5, how likely do you think youwould use it as your primary contact method among group members?
7. StarSeek is built to be a simple team management app. On a scale of 1-5, how likely do you think you would use StarSeek over another platform such as a Facebook group for club management?
8. On a scale of 1-5, how useful do you think StarSeek would be when trying to do networking among peers?
9. On a scale of 1-5, how well do you think StarSeek would be when trying to recruit members fora club?
10. On a scale of 1-5, what would you overall rate StarSeek?

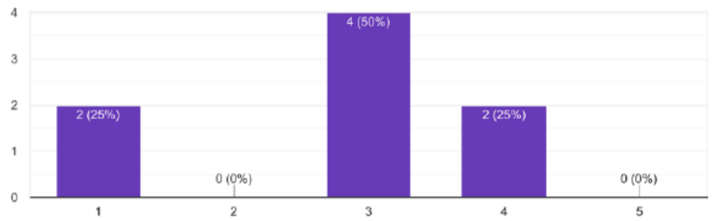
On a scale of 1-5, how easy was it to navigate through StarSeek?
8 responses



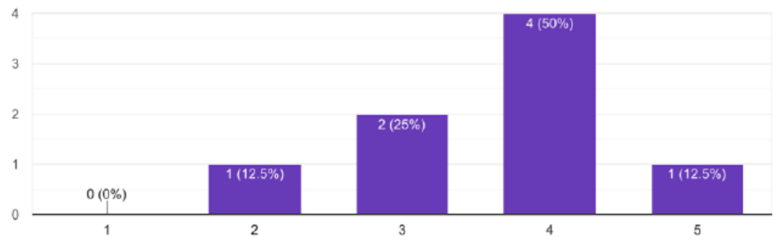
On a scale of 1-5, how clear was StarSeek's design?
8 responses

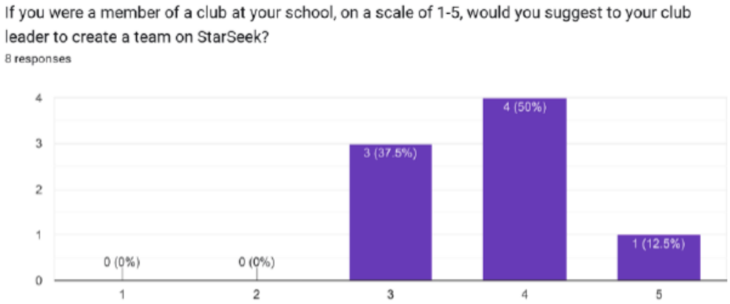


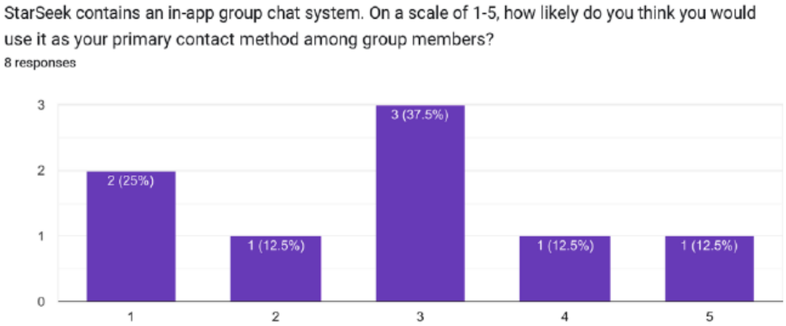
On a scale of 1-5, how likely are you going to use StarSeek to find someone outside of school?
8 responses

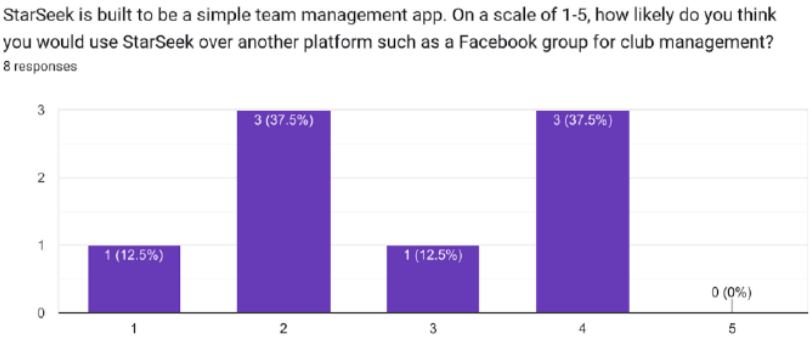


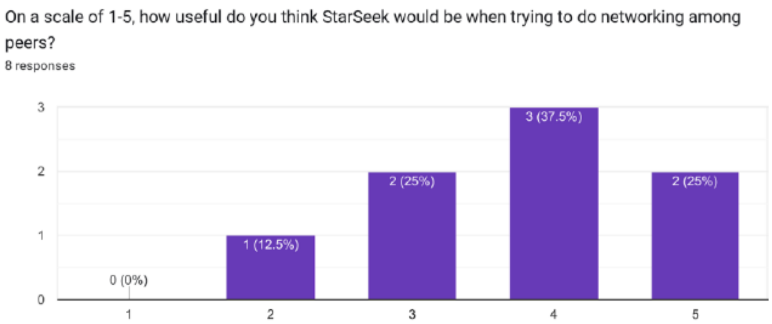
On a scale of 1-5, how effective do you think StarSeek would be at bridging connections between schools?
8 responses




If you were a member of a club at your school, on a scale of 1-5, would you suggest to your club leader to create a team on StarSeek?
8 responses
0 (0%)
0 (0%)
3 (37.5%)
4 (50%)
1 (12.5%)
1
2
3
4
5


StarSeek contains an in-app group chat system. On a scale of 1-5, how likely do you think you would use it as your primary contact method among group members?
8 responses
2 (25%)
1 (12.5%)
3 (37.5%)
1 (12.5%)
1 (12.5%)
1
2
3
4
5


StarSeek is built to be a simple team management app. On a scale of 1-5, how likely do you think you would use StarSeek over another platform such as a Facebook group for club management?
8 responses
1 (12.5%)
3 (37.5%)
1 (12.5%)
3 (37.5%)
0 (0%)
1
2
3
4
5


On a scale of 1-5, how useful do you think StarSeek would be when trying to do networking among peers?
8 responses
0 (0%)
1 (12.5%)
2 (25%)
3 (37.5%)
2 (25%)
1
2
3
4
5

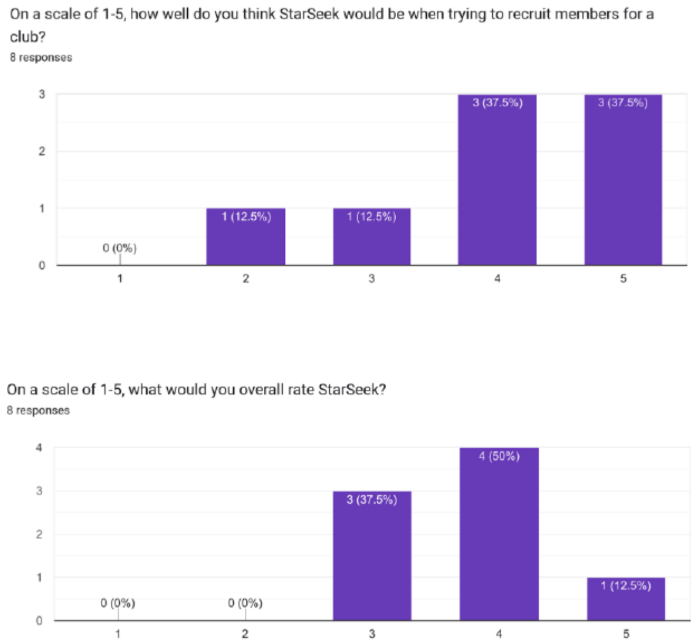


Figure 8. Figure of experiment 1

The mean score for the question “On a scale of 1-5, how easy was it to navigate through StarSeek?” was 4.25, and the mode was 4. The lowest value was 4, and the highest was 5. This data indicates that StarSeek is an intuitive application to navigate.

The mean score for the question “On a scale of 1-5, how clear was StarSeek's design?” was 3.75, and the mode was 4. The lowest value was 2, and the highest was 5. This is likely based on the participants’personal preferences.

The mean score for the question “On a scale of 1-5, how likely are you going to use StarSeek to find someone outside of school?” was 2.75 and the mode was 3. The lowest value is 1, and the highest value is 4. Participants likely gave it a lower score because there are currently as of writing not a large amount of active users. If StarSeek had a larger install base, then we predict that this score would have been higher.

The mean score for the question “On a scale of 1-5, how effective do you think StarSeek wouldbe at bridging connections between schools?” was 3.63, and the mode was 4. The lowest value is 2, and the highest value is 4. Participants who rated this question lower might be concerned about the feasibility of StarSeek.

The mean score for the question “If you were a member of a club at your school, on a scale of 1-5, would you suggest to your club leader to create a team on StarSeek?”was 3.75, and the modewas 4. The lowest value is 3, and the highest value is 5. Participants would likely score higher if there were more users on StarSeek.

The mean score for the question "StarSeek contains an in-app group chat system. On a scale of 1-5, how likely do you think you would use it as your primary contact method among group members?" was 2.75, and the mode was 3. The lowest value is 1, and the highest value is 5. Participants gave a lower score because the chat system isn't as robust as other communication apps.

The mean score for the question "StarSeek is built to be a simple team management app. On a scale of 1-5, how likely do you think you would use StarSeek over another platform such as a Facebook group for club management?" was 2.75, and the mode was 2. The lowest value is 1, and the highest value is 4. Participants gave a lower score because people are more familiar with Facebook than StarSeek. However, more people will be familiar with StarSeek as it goes.

The mean score for the question "On a scale of 1-5, how useful do you think StarSeek would be when trying to do networking among peers?"was 3.75, and the mode was 4. The lowest value is2, and the highest value is 5. The data indicate that participants do consider utilizing StarSeek to network among peers.

The mean score for the question "On a scale of 1-5, how well do you think StarSeek would be when trying to recruit members for a club?" was 4, and the mode was 4. The lowest value is 2, and the highest value is 5. This indicates that users would find StarSeek great for club recruitment.

The mean score for the question "On a scale of 1-5, what would you overall rate StarSeek?" was 3.75, and the mode was 4. The lowest value is 3, and the highest value is 5. This indicates that overall users found that StarSeek was a good app. With further iteration we expect this score to be higher once we address user concerns.

4.2. Experiment 2

StarSeek utilizes a recommender system, which allows students to be recommended teams to join based on a search query that they enter. It is vital for this search system to be as accurate as possible because it affects the usability and capability of students to network through the app.

For this experiment, we will be conducting an accuracy assessment for two types of natural language recognition machine learning models. The first utilizes cosine similarity, and the second utilizes a Python library named Gensim. We will train the models with a set of descriptions and names for 16 teams. These teams are not real teams but nonetheless are written to be plausible and strong in their theming. The teams are:

- 1 general STEM team
- 1 STEM green energy team
- 1 STEM Sociology team
- 1 general engineering team
- 1 engineering aeronautics team
- 1 green energy team
- 1 programming team
- 1 biotechnology team

- 1 mathematics team
- 1 mathematics team with a focus on cryptography and quantum mechanics
- 1 data science team
- 1 robotics team
- 1 art team
- 1 health team
- 1 marinology team

We will test these models with the following 10 test cases:

1. Ocean
2. Engineering
3. Robotics
4. science technology engineering math
5. renewable green energy
6. aeronautics and space
7. medicine and health and fitness
8. art technology
9. team projects
10. creativity and problem-solving

For each of these cases, we will get the top 3 recommendations back from both models. For each of these recommendations, we award a point to the model if the recommendation is considered relevant or accurate given the query. For each case, a winning model is determined based on the model with the higher point count.

Query	Cosine	Gensim	Winner
"ocean"	2	1	Cosine
"engineering"	3	1.5	Cosine
"robotics"	2	2	N/A
"science technology engineering math"	2	0	Cosine
"renewable green energy"	2	2	N/A
"aeronautics and space"	2	1	Cosine
"medicine and health and fitness"	1	1	N/A
"art technology"	1	1	N/A
"team projects"	3	1	Cosine
"creativity and problem solving"	3	2	Cosine

Figure 9. Figure of experiment 2

The average accuracy score awarded to the cosine model was 2.1. For the gensim model, the average accuracy score awarded was 1.25. On one occasion, the gensim model was awarded half a point due to the recommendation given being slightly relevant, but not enough to earn a point, though this half point did not have any bearing on whether gensim won that case. The cosine model is thus on average significantly more accurate than the gensim model. On 3 occasions, both models had the same accuracy score and thus the winner was a tie, or non-applicable to the final results.

When tallying up the winning model, cosine is unanimously the winner. There were no cases in which the gensim model won. Therefore, we can say that the cosine model is generally a more accurate machine learning natural language recognition model than that of gensim. This experiment has demonstrated to us empirically that the cosine model is more suitable for good results in our application, and thus the backend of our application utilizes the same cosine model as used in this experiment.

5. RELATED WORK

Another system, the “Ikka Veima Football Club Content Management System” created by Prof [11]. Kari Systä offers a highly efficient management system tailored for football clubs, ensuring precise management within the context of sports teams. However, its narrow focus on football clubs limits its applicability to a broader spectrum of organizations and activities, potentially missing out on valuable insights. Furthermore, the solution's precision may inadvertently lead to excessive complexity, which can be time-consuming. In contrast, StarSeek provides an accessible and functional platform that connects users with like-minded individuals without overwhelming them with a complex user interface. Its openness to various clubs encourages a more comprehensive exploration of interests and collaborations across different domains, making it a more versatile option.

A project involving a Sports Club Intranet, created by Oriol González Navarro, utilizes a database system for centralizing data, making it easier to access and manage compared to scattered paper

records [12]. However, this solution has some potential limitations in establishing a structured database. As this solution still heavily relies on manual processes, its dependence on Microsoft Access as technology can possibly become a bottleneck. StarSeek has a strong database stored in Firebase, where maintenance is rarely needed. Starseek does not rely on local expertise to find and retain personnel for data management.

6. CONCLUSIONS

Improvements to StarSeek include upgrading the chat system and refining the search capabilities. To improve the chat system, features like image and document sharing can be integrated, and user identification can transition from email-based to usernames for enhanced privacy. Simultaneously, the search system can be refined for accuracy by prioritizing clubs in the same geographical region, fostering in-person interactions. To realize these improvements, the chat system may require an updated design and functionality on FlutterFlow, while the search system could benefit from additional tools like Python libraries or ChatGPT for enhanced search algorithms [13]. These modifications aim to create a more dynamic and user-friendly experience within StarSeek, fostering deeper connections and facilitating more meaningful club interactions, whether virtually through chat or through localized, real-world engagements.

StarSeek continues to update and is on the path to becoming a more dynamic and user-centric platform [14]. It continues to explore and implement new strategies and ensure that students have the opportunity to thrive in what they are passionate about. With these potential improvements, StarSeek wants to facilitate deeper club interactions and create a more vibrant community for its users.

REFERENCES

[1] Necşoi, Daniela Veronica. "Team and team management-a practical approach." Educația Plus 12.1 (2015): 317-328.
[2] MacAllister, James. "What should educational institutions be for?." British Journal of Educational Studies 64.3 (2016): 375-391.
[3] McCarthey, Sarah J. "Home–school connections: A review of the literature." The Journal of Educational Research 93.3 (2000): 145-153.
[4] Jayasinghe, Udeni, Anuja Dharmaratne, and Ajantha Atukorale. "Students' performance evaluation in online education system Vs traditional education system." Proceedings of 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV). IEEE, 2015.
[5] Finn, Jeremy D. "Expectations and the educational environment." Review of educational research 42.3 (1972): 387-410.
[6] Svensson, Lars EO. "The zero bound in an open economy: A foolproof way of escaping from a liquidity trap." (2000).
[7] Moschogianni, Georgia. "Investigating factors Impacting Discord and Snapchat Use Behavior: Perspective from Swedish users." Journal of Digitovation and information system 2.1 (2022): 41-54.
[8] Mellarkod, Veena S., Michael Gelfond, and Yuanlin Zhang. "Integrating answer set programming and constraint logic programming." Annals of Mathematics and Artificial Intelligence 53.1-4 (2008): 251-287.
[9] McKinney, Scott Mayer, et al. "International evaluation of an AI system for breast cancer screening." Nature 577.7788 (2020): 89-94.
[10] Dewes, Christian, Arne Wichmann, and Anja Feldmann. "An analysis of Internet chat systems." Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement. 2003.
[11] Kulikova, Lidiya, and Anna Goshunova. "Evaluation of management system quality: case of professional football clubs." Academy of Strategic Management Journal 15 (2016): 122.
[12] González Navarro, Oriol. Sports Club Management. BS thesis. Universitat Politècnica de Catalunya, 2017.
[13] Ray, Partha Pratim. "ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope." Internet of Things and Cyber-Physical Systems (2023).
[14] Recordon, David, and Drummond Reed. "OpenID 2.0: a platform for user-centric identity management." Proceedings of the second ACM workshop on Digital identity management. 2006.
[15] Chougale, Pankaj, et al. "Firebase-Overview and Usage." International Research Journal of Modernization in Engineering Technology and Science 3.12 (2021): 1178-1183.