Tutors+ App

An application to help in tuition search and promotion

Ryan Camilleri
Faculty of ICT (Department of Artificial Intelligence)
University of Malta
ryan.camilleri.18@um.edu.mt

Francesco Borg Bonello
Faculty of ICT (Department of Artificial Intelligence)
University of Malta
francesco.borg.18@um.edu.mt

ABSTRACT

The Maltese culture has always encouraged students to perfect their studies and attend private tuition in order to achieve their academic goals. With an increasing population, this academic sector is seeing a huge rise in numbers and countless students are now in need of tuition. Currently, there lacks a dedicated platform aimed at helping these individuals to find tuition, along with tutors that have no professional means of promoting themselves. This paper focuses on the development of the Tutors+ app using Flutter as the development environment of choice along with Firestore, for cloud storage. This paper highlights the use of AI concepts like those of user profiling, recommender systems, geolocation and more, used within the app; All of which help to facilitate the requirement of providing this essential one-stop-shop for all stakeholders making use of the application, in an efficient and smart manner.

KEYWORDS

User Profiling, Recommender Systems, Cloud Storage, Firebase, Education, Private Tuition, Geolocation, Categorical Filtering, Google Maps, Range Search, Review System, Flutter

1. INTRODUCTION

1.1 Motivation

It is common within the Maltese educational ecosystem, that students attend private lessons apart from lessons at school. This would most often be done by individuals to improve their grades and/or for personal revision before sitting for exams, like those of MATSEC. With the growth in population, the number of students that currently populate the Maltese islands is at a daily increase. Due to such growing numbers, the private tuition industry is also seeing a beneficial growth and is greatly searched for by the students themselves or otherwise by the student's guardians. As it stands, the culture mainly relies on the 'word-of-mouth' principle whilst also using social media sites, to both advertise and search for tuition. When trying to find any possible alternatives, it was observed that there are little to no professionally-dedicated platforms which would help make such a process for searching and promoting tuition faster and more accessible to all. This lack of a professional platform in today's academic industry can be profoundly observed when going through several social media sites, mostly that of Facebook, and observing posts made by countless prospective students on social pages dedicated to tuition posting. These students would all require some form of tuition or another and would have no clue which classes exist, and which do not. Taking tutors into consideration, these currently have no other medium from which they could promote their tuition and attract prospective students, and hence are forced to adopt brute posting procedures.

1.2 Problem & Challenge

The motive leads to various problems and challenges that must be evaluated. The crux of the problem is that currently there is no application in the market which serves as a platform for the aforementioned stakeholders: tutors, students & guardians. This leads to tutors resorting to posts on social media feeds across different groups with no intent of a central hub. There is also no existing catalogue of tuition, so any prospective student would not have access to further information about the tuition's tutor, such as, their qualifications, teaching warrant, and police conduct.

Moreover, Tutors' advertisements get buried under countless other tutors' posts which would seemingly make these posts redundant and out-dated after a short amount of time. This problem of '*stack posting*' favors tutors who frequently post, since their adverts would always be on top of other tutor's posts, which have been posted earlier. This poses a threat to other tutors who may find it a struggle to keep up with the overload of frequent posts, only for their listing to get lost deep in one's newsfeed which is highly demotivating for the tutor. In this way, the student would have potentially missed out on finding the right tuition for them.

On the other hand, from the point of view of a student/guardian there is an evident categorical problem, as there is no way of filtering the clutter of posts that they find on social media. This leads to countless hours of manually searching various social media communities in order to find a tuition that satisfies their requirements. Such categorical filters may include their desired category, level, location, along with other factors such as price or even if tuition is offered through online means in the case of social distancing or other problematic situations.

Every student has their own unique interests when it comes to education. There is currently no recommendation system that automatically links the student with a tuition based on their common interests. User profiling can help the student feel constantly updated with new tuition, while the tutor has a stress-free experience when attracting prospective students. Otherwise, the traditional system forces one to search for tuition manually rather than having a system that recommends tuition to them automatically.

1.3 Aim & Objectives

The aim of this paper is to derive a solution in the form of an app-based platform which fills the gap in the education environment by designing a system to meet the needs of both tutors and students. In this way, it would be beneficial to all three stakeholders. This paper will tackle the highlighted problems such as user-profiling, tuition search, tutor advertising, tuition filtering and also recommendation systems. The primary objective is to build a platform via an app, called Tutors+. This app will relieve the stress and hassle involved in finding the best tuition for oneself and one's child, ensuring peace of mind due to tutor credibility. This objective will be accomplished through many other features revolving around the fact that the app is user friendly and eye-catching through the use of simple graphics and an effective infrastructural backend to support the system. The core features that aid this objective are described in the coming paragraphs.

The users of the app who are students/guardians should have the facility of customizing their profile with their basic details and also interests. This feature goes hand-in-hand with a recommendation system that will facilitate the searching process in a reverse manner, which means that instead of the student/guardian searching for a tuition, the tuition finds its way to the user automatically. The interests will not only be manually inputted by the user but also automatically added to their user model as they search for that category, hence further profiling their interests. The user would also have the ability to enroll in a tuition or add it to favourites to refer to it in the future.

The tutors should have a clear and customisable source which is consistent throughout advertisements, meaning that they only need to edit it at least once and it will always be visible and can be seen by any prospective student in a constant fashion, through results from filtering or recommendations. Their profile should indicate details of the tutor such as their qualifications and warrant along with their available tuition. This profile should also include a public review system for students that have attended their tuition to enhance tutor credibility for prospective students.

Another important feature is having a map populated with a plethora of tuition in accordance with a list view of the different tuition visible on the map. This information is visualized through location and range filtering that alters the user interface, turning it into a dynamic, user-friendly map that is updated in real-time in accordance with the backend system. The tuition on this map is to be filtered by category, level, and locality whilst also algorithmically promoting tuition that holds premium status. The premium status means that priority is given to tuition belonging to tutors that subscribe to the monthly fee.

2. LITERATURE REVIEW

In order to satisfy the derived objectives, various techniques must be researched to implement our features. Such research will be of significant aid when planning the platform design. Moroney L. defines the Firebase Realtime Database as "a NoSQL cloud-based database that syncs data across all clients in real time, and provides offline functionality" [1]. This satisfies the aforementioned objective to host a platform that revolves around a robust and extensive backend that is capable of holding a medium to large amount of users that asynchronously transfer data between them. The storage of all the data in a JSON file gives rise to real time functionality since all connected clients share one instance, automatically receiving updates with the newest data [1]. Alsalemi et al. state that Firebase proved to be a proper communication solution that satisfies the urgent nature of medical training in its application of the ECMO procedure [2]. This reinstates the vital aspect of fast and immediate data that can be accessed by all users in a real time manner.

Since Firebase does not consist of tables, an important aspect for its usage is NoSQL. As its name implies, NoSQL does not require creation/manipulation of data via SQL commands. Ohyver et al. highlight various advantages of NoSQL over traditional SQL databases, such as its cost of adding a server is much cheaper than upgrading the capacity of the server. With NoSQL, data will automatically be balanced from various servers to the server pool, meaning that if there is one server down, the data can be directly migrated to another server. Thus, the possibility for server down is smaller when compared to SQL [3]. Stonebreaker M. argues that "blinding performance depends on removing overhead" [4]. He then states that overhead has nothing to do with SQL but instead revolves around traditional implementations of ACID transactions, multithreading, and disk management. Stonebreaker implicitly states that the performance arguments in favour of NoSQL databases are insufficient, hence supporting SQL. Tests conducted by Ohyver et al in [5] conclude that Firebase Realtime Database is more suitable for their test ground mobile app. Apart from significantly better performance rates, Firebase's key features proved to be more suitable for their app since it provides real time data and simplifies the backend process due to the developer not having to worry about the server-side code. The results from this paper along with the highlighted points mentioned throughout this paragraph inspire the core of this application to achieve maximum efficiency and performance.

The hierarchical structure of the JSON Tree used by Khawas and Shah in [6] proves to be the optimal means as to structure data in Firebase, with a root directory comprising of distinct collections that hold different documents which in turn reference each other and hold the data in fields of different types. Structuring the database as such, should be important in satisfying our goal.

The mobile app e-commerce revolution is causing the mainstream app design to evolve. A great example of this is TripAdvisor as highlighted by P. O'Connor in [7]. Take the example of finding a nearby restaurant to view further details about it such as its location, contact details, photos and reviews. This is a source of inspiration to develop the platform in the way users can similarly access profiles of tutors. This will allow users, in this case, tutors, to customize their profile with details about themselves that will be displayed to other users in a clear and consistent format which is another objective, that our app must satisfy. From a business perspective, this allows the development of a subscription feature where the tutor pays a monthly fee to gain perks and in turn receive priority for their advertisements to appear more frequently, which would gain more attention from other users. From another user's perspective, they should add tutors to their favourites list, have the option to enroll, and place a review if they have received tuition from them. This perspective is like that of a user on TripAdvisor [7].

Another objective this app seeks to accomplish is providing an interface in the form of a map indicating locations of different tuition. Hu and Dai create a similar system in [8] that combines Google Maps API along with a database that together allow a user to visualize data scattered over the map. Dong et al. use Google Maps along with Geolocation services in order to calculate range between the user and a point of interest [9]. Categorical filtering of data is essential for a user to specifically display the data that they want to see. Martinez and Lopez implement filtering done client-side in their application using Firebase as a backend in [10]. Cubranic carries out categorical filtering based on multidimensional data in [11] with the returned data then displayed to the user. Taking inspiration from all papers leads to a fully interactive map that gives the user the ability to dynamically filter and visualize changes on the map in real time leading to a user friendly and attractive experience. This leads to the intuitive idea of implementing such a feature when searching for tuitions based on categorical filtering and also range, with all changes visualized on the same screen.

In order to facilitate the process that a user undertakes to find a tuition, it is vital that we profile the user so as to further understand the subjects they are interested in, both directly and indirectly, so then our system can automatically recommend tuition which might interest them. Schiaffino and Amandi describe a profile as "a description of someone containing the most important or interesting facts about him or her" [12]. It is also mentioned within this paper that discovering these differences is vital to provide users with personalized services, which is what the platform to be developed intends to achieve. This information can be gathered either explicitly or learned via some intelligent technique by inferring unobservable information about users from observable information about them such as their actions or utterances (Zukerman and Albrecht, 2001). Given that our user has the ability to keep a profile with basic information, a feature can be included where they may optionally input their interests located in the respective screen. However, this method is explicit. Middleton et al. in [13] explore an ontological approach to user profiling in the context of a recommender system. Middleton et al. describe the benefit of using recommender systems as "remove(ing) the burden of explicit search queries by learning profiles of the sort of things relevant to users, and then recommending new items that similar people have liked or are similar to previously relevant items." Schafer et al. in [13] define its usage as "Recommender systems are used by E-commerce sites to suggest products to their customers." Schafer et al. also state that information can also be inferred from the demographics of a user or based on the analysis of past behaviour. Middleton et al.'s system called Foxtrot (in their paper [13]) infers user interests from profile visualisation. This gives rise to the idea that our app can take a similar approach when building the user's interests. This can be done by having the system intelligently learn about user's interests from their search history when the user applies filters, building a conceptual model for each specific user containing such deduced information that our app intelligently learns over time. This information can then be used to recommend tuition that might interest them. Hence, having a synergy of information obtained explicitly, and having intelligent techniques, will in turn aid the system to create a dynamic user profile and advertise tuition that will likely interest them. This is the main inspiration for another core feature that the app aims to implement, that of being an adaptive recommender system that gains information through a growing user profile.

3. ANALYSIS & METHODOLOGY

3.1 Development Kit & Database Structure

Extensive research was carried out in order to determine the best types of developmental environments to build the app. This however was heavily dependent on which mobile operating system was to be targeted by Tutors+. Initially, it was decided that the Android OS would be the targeted system of choice and hence, Android Studio [14] was what seemed to be the ideal choice at the time. Upon further research, the team came across Flutter which is an open-source UI software development kit, published by Google [15]. Being a relatively new environment, which allowed for developing a single codebase for both iOS and Android, the team was instantly in agreement for using such software. Further benefits to using this kit, included those of faster development through Flutter's hot reload function thanks to the Dart language it uses [16]. This language features a 'just-in-time' execution engine which allows for such hot reload procedures, to be possible; In addition, Dart compiles directly to native machine code leading to better performant applications.

Each and every Flutter app can be seen as a composition of widgets where each widget is built from a nested collection of other widgets. Essentially, everything in Flutter development relates to widgets which contribute to both the graphical interface of the app, along with the back-end procedures necessary for distribution of data. Some widgets allow for a change in state whilst others maintain it. The so called Stateful widgets, are special widgets that generate State objects to build more complex experiences and allow for a dynamic interface. This is achieved by changing the state of the widget thereby rebuilding the whole widget tree under it, hence updating all other widgets inheriting from it, when responding to some input. An application of this widget within our app, can be seen in the user's list of favourite tuition. Whenever the user taps on the heart icon, the state changes and the icon displayed is reversed to represent the user action. On the other hand, Stateless widgets are those which receive arguments from parent widgets and cannot change state. These widgets are static representations which have no memory of their previous states, used when change in data does not occur or is not required. Hence, when a dynamic page was required throughout the app, the most common approach was to have a Stateful widget at the source, which feeds data to its child Stateless widgets, controlling which widgets are shown and how data is distributed in each state.

As previously described, some widgets are targeted towards back-end processes as opposed to UI. One such widget which was often used, was the *StreamBuilder* widget [17]. This widget allowed for real time updates of Stateful widgets by supplying a stream of data from the database in the form of snapshots. This stream is specifically selected and is used in ways depending on the widget being built. Such streams can take the form of a stream of user-data which supplies the widgets with a constant flow of up-to-date information, such that when this user data changes, the stream informs the widget of this alteration and forces it to rebuild to account for it. This type of stream is used in almost every view within the app. One example of where this stream is used can be seen within the sign-up instance. Once the user signs up, the user is immediately redirected to the homepage screen. This occurs due to a change in user stream data which informs the app to rebuild its widget tree and allow the user to view previously hidden pages. By using this widget alongside others, it was ensured that the application was real-time, allowing for data to be instantly updated when changed in the database.

This data was classified into three models for the making of the prototype. These include the user, tutor and tuition model. Each model stored data related to the object they are referred to, of which is brought about from the database. As previously stated in section 2, it was of best interest to employ the Firebase database, more specifically, the cloud Firestore [18]. Firestore is a cloud-hosted NoSQL database, used to build hierarchical data structures that scale in real-time as data is read from or written to it. This database offers a unique experience compared to those of SQL databases since data is not stored in tables, but instead, it is stored in collections, each of which have a list of documents related to them. Such documents have a variety of data-fields which store data linked directly to that specific document. One noticeable difference when shifting from SQL databases to NoSQL had to do with the fact that no foreign key-like data structures existed to link certain documents to others, especially in different collections. A similar structure, however, was the reference field type which stored the reference link of other documents to create some form of relationship. This however was still heavily oriented towards the SQL approach many are used to, which is not always recommended when using such hierarchical databases.

In Firestore, repeated data is greatly suggested when dealing with relationships which greatly opposes the goal of SQL databases. Given the example that a tutor has a tuition list full of tuition that he himself had created, instead of storing just a reference to the tuition document within the collection of all tuition, data related to it should be stored alongside the tuition reference. By doing this, instead of accessing another document separately and reading its contents, the tutor document would have all the necessary data linked to his tuition in order to be displayed on his account. By allowing for such repeated data, instead of performing two reads considered as being an inefficient practice, the database is tasked to read just the tutor document and nothing else. By keeping strict with this idea and accessing references only when needed the overall infrastructure was ensured to be as efficient as possible, sticking to several guidelines throughout the application back-end.

3.2 Map Interface & Filtering

The integration of Google Maps API was essential in providing a dynamic interface for the user. This together with the ability to categorically filter tuition gives the user the facility to customize their search and aid the tuition searching process. The visualization of the map was made possible using the Google Maps Flutter library [19]. The Geolocator library [20] was used in conjunction with the Google Map widget to indicate the user's real time location while using the feature.

All available tuition is fetched from Firebase, with each tuition document made into both a tuition object and also a marker object that forms part of the Google Maps API library [21]. Each marker's distance from the user is also calculated. These markers are displayed in the map. This, together with a slider that allows the user to control the radius, gives the user the ability to adjust the maximum range of tuition that they require to be displayed on the map. For an enhanced user experience, the range is dynamically depicted as a circle on the map which scales according to the requested range. Therefore, if a tuition is within this circle, then it is displayed, otherwise it is hidden.

Besides radial filtering, a sliding up panel from the *sliding_up_panel* library [22] presents the user with a variety of different categorical filters amongst others. The categorical filters are: level; category; and locality. The user also has the option to disable the radius filter to display all available tuition that matches their requirements, in turn hiding the button that enables the visibility of the slider which controls this functionality. The user can also enable the locality option which then allows them to categorically select which locality they would like to search for tuition in. These filters can then be applied or reset to default. The map is then updated with the tuition that match these filters. It should be noted that the process of filtering according to the user's needs also plays an important role in profiling the user, namely by their selection of category and level. This will be explained later on.

Apart from the core map interface, the user is shown a list view of all tuition that are currently present on the map. These are ordered by tuition of tutors holding premium status appearing before those tutors who are otherwise freemium. This gives tutors an incentive to subscribe to the monthly fee. Each 'card' in the list displays basic details about the tuition, such as the name, level, and category. An info button is also shown, that when selected, displays an alert dialog box with a further description also containing a button to 'favourite' that particular tuition as well as a button that takes the user directly to the tutor's profile. Here, the user can then enroll in the tuition of their choice.

The map interface is synergized along with the list view, such as when a marker is tapped on the map, the list view automatically scrolls and highlights the 'card' of the particular tuition, whilst the map zooms in on the selected marker. The user is given the functionality to toggle between normal view of the map and satellite view. This is done so they can further understand the location of the tuition. The user can also focus on their current position by selecting the location button. The user is also given the intuitive capability of using core app gestures, such as pinch, drag and zoom, even though zooming in/out of the map is done automatically.

3.3 Recommender System with User Profiling

The need for a recommender system gives rise to the importance of user profiling, which can be done both explicitly or gathered by our system using an intelligent means. Apart from letting the user manually input their interests, Tutors+ caters for the intelligent acquisition of this information through the categorical filters the user applies when searching for tuition. For example, a user has not explicitly stated that they are interested in French, however results based on their search history when filtering a category for a tuition show that they indeed have searched for a French tuition. Hence, our system caters for this inexplicit data by holding a search history and extracting information that it can use to recommend tuition later on.

As previously stated, Tutors+ builds a unique model based on each user's explicit interests along with those intelligently derived based on their search history. The backend system holds this data in the form of 3 lists for each user: explicit category interests; searched categories; and searched academic level. These lists are updated whenever the user edits their profile or applies a new respective filter when searching for tuition. In the case of intelligently acquired data, the database updates these lists automatically by first checking if the category being searched is new, or already exists. If the former occurs, then it is added, otherwise it is ignored. If adding this category to the list causes the amount of categories to be greater than 5, then the first (i.e. the oldest) category in this list is removed. This ensures the maximum list size is always 5 to ensure that only the latest categories are considered when understanding the user's interests. Tracy and Robbins in [23] highlight a significant change in student interests over time. Our design complies with the change in users' interests over time therefore adapting to the latest interests they may have.

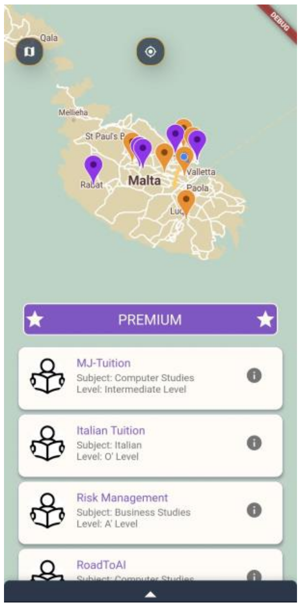
Now that the user's interests have been captured, the system can now recommend tuition to the user. This recommender system is displayed on the home screen in the form of a list view with the basic details of the tuition offered. This list view is formed by first checking which tuition in the database conforms with the user's respective interests by the category of tuition. These are ordered by the level that the user has recently searched for. For instance, if a user has recently searched for Maltese O'Level tuition, then the system automatically recognizes Maltese as an interest if it was not already explicitly stated, along with the fact that the user is searching for O'Level tuition. Therefore, the recommender system orders all the tuition in the list view in the order of the level that the user recently searched for, in this case, O'Level. Now every time the user logs into the app, they are presented with this recommender list view on the homepage. In turn, this keeps the user informed with the latest tuition that they might be interested in, consequently facilitating the overall task of finding tuition that suits their needs. This entire recommendation procedure can be seen to work in a reverse manner, which means that instead of the student/guardian searching for a tuition, the tuition finds its way to the user automatically. It should also be noted that if no tuition is found to conform to the user's interests, then the 5 most recent tuition are displayed in the list view.

4. TESTING, RESULTS & EVALUATION

A series of tests were conducted to ensure that the objectives of the app have been accomplished in an efficient and clear manner. Since the app produced is a prototype, testing was done within a closed community, i.e. a small group of people who were tasked to create various accounts of different interests and also add tuition of their own. Both premium and freemium accounts were created to further visualize how the different account statuses varied from each other. Apart from general usage of the app such as creating/logging in accounts, editing profiles, becoming a tutor as well as adding tuition, the main components that needed attention are the following: searching for tuition on the map and applying filters, and using the user's search history in conjunction with their explicit user interests. This would be done, to intelligently recommend tuition via the app's recommender system on the homepage. The test results will be displayed via screenshots of the app from the perspective of a new account. For showcasing purposes, consider an example of a user searching for Computer Studies A' Level tuition.

4.1 Tuition Search & Filtering

The user is initially presented with a map and list view interface as depicted in Figure 1. Upon applying a filter on category (Computer Studies) and level (A' Level) as well as adjusting the radius to 8km, the markers on the map are dynamically filtered. This can be seen in Figure 2. The markers are colored according to the membership status of the tutor of the tuition: purple for premium and orange for freemium. This filtering also contributes to the recommender system that will be shown in the next section.



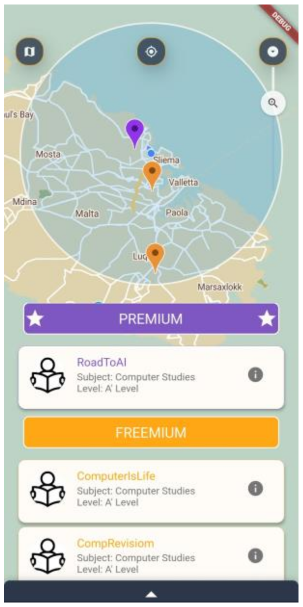
MJ-Tuition
Subject: Computer Studies
Level: Intermediate Level

Italian Tuition
Subject: Italian
Level: O' Level

Risk Management
Subject: Business Studies
Level: A' Level

RoadToAI
Subject: Computer Studies

Figure 1: Initial Search Interface



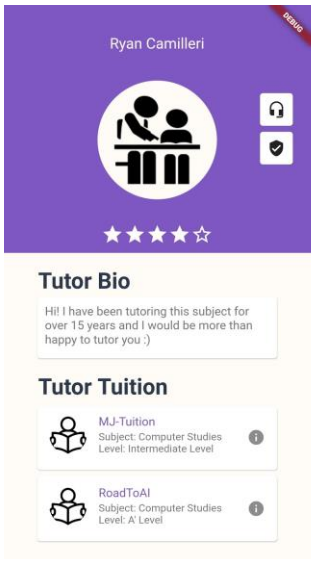
RoadToAI
Subject: Computer Studies
Level: A' Level

ComputerIsLife
Subject: Computer Studies
Level: A' Level

CompRevision
Subject: Computer Studies
Level: A' Level

Figure 2: Search Interface after applying filters

Let us assume that the user is interested in the premium tuition currently being displayed on the map in Figure 2. After tapping on the information button, they are presented with the tutor's profile screen that contains further information about the tutor, such as their tuition and where their tuition is situated through a marked map. This is previewed in Figure 3 below. This screen also shows that tuition is available via online means, and that the tutor is warranted. Along with the bio, a rating section is also shown that displays the tutor's average rating from all student reviews which are shown in Figure 4.

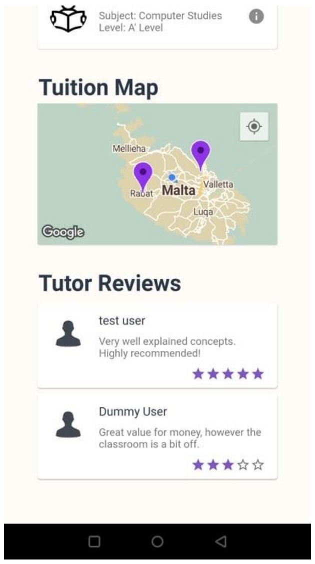


Hi! I have been tutoring this subject for over 15 years and I would be more than happy to tutor you :)

MJ-Tuition
Subject: Computer Studies
Level: Intermediate Level

RoadToAI
Subject: Computer Studies
Level: A' Level

Figure 3: Tutor Profile with details related to it



Subject: Computer Studies
Level: A' Level

test user
Very well explained concepts. Highly recommended!
★★★★★

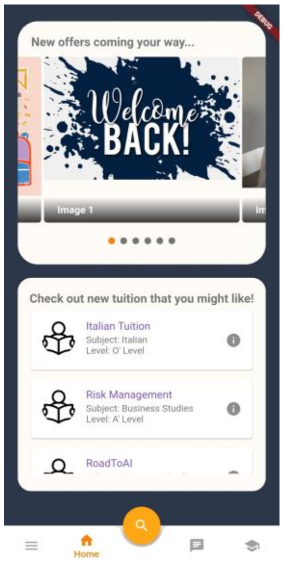
Dummy User
Great value for money, however the classroom is a bit off.
★★★☆☆

Figure 4: Reviews done on the tutor profile by enrolled students

From this section it was concluded that the user's interaction with the map was optimal. A variety of tuition was presented on the map along with the ability to filter based on multiple factors. These proved to work seamlessly as the filtering results conformed to the applied filter. The tutor's profile was also loaded well with all the expected details shown accordingly. The reviews were also listed as expected, containing a short description and the final rating.

4.2 Recommender System

The recommender system suggests tuition that might be of interest to the user. For this account, the user manually defined their interest in Mathematics. Based on their search history in the previous section (Computer Studies & A' Level) the system should also derive the interest in Computer Studies, giving priority to those tuition which are A' Level. Figure 5 shows the homepage. Since at this point in time they had not yet added any interests or even searched for tuition, the recommender view therefore displays a list of the most recently added tuition. Figure 6 shows the contents of a scrolled recommender view after the user adds the Mathematics interest and then proceeds to search for Computer Studies A' Level tuition.

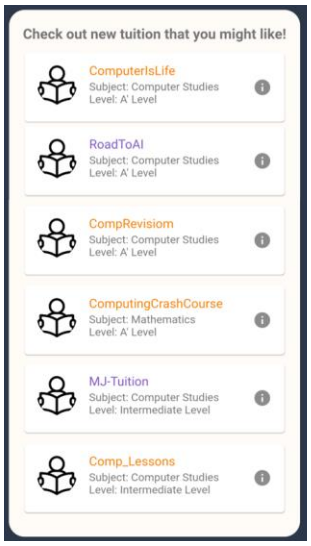

DEBUG
New offers coming your way...
Welcome BACK!
Image 1
Check out new tuition that you might like!
Home

Italian Tuition
Subject: Italian
Level: O' Level

Risk Management
Subject: Business Studies
Level: A' Level

RoadToAI

Figure 5: Homepage recommender view showing latest tuition



ComputerIsLife
Subject: Computer Studies
Level: A' Level

RoadToAI
Subject: Computer Studies
Level: A' Level

CompRevision
Subject: Computer Studies
Level: A' Level

ComputingCrashCourse
Subject: Mathematics
Level: A' Level

MJ-Tuition
Subject: Computer Studies
Level: Intermediate Level

Comp_Lessons
Subject: Computer Studies
Level: Intermediate Level

Figure 6: Fully scrolled recommender view contents

The output of the results is as expected. Figure 5 displays the latest tuition since the user's interests have not yet been recorded. The tuition list in Figure 6 shows available tuition for the recently searched category, Computer Studies. A Mathematics tuition is also in the list, as the user had recently included Mathematics as an area of interest. The level of interest is also taken into account as the tuition at A' Level takes priority over other tuition of other levels. Overall, this system works as intended, in turn facilitating the searching procedure for the user.

5. CONCLUSION AND FUTURE SCOPE

This paper highlights the developmental procedures undertaken by the Tutors+ team to create a viable application prototype, capable of simplifying tuition search and promotion. The paper helps in showing what decisions were made and why such decisions were taken. It also provides a clear explanation of what features were implemented towards better facilitating the need for such tuition search and advertising. By using the Google Maps API along with a recommender system and other methodologies, the user is ensured a reliable and efficient experience that promotes up-to-date information coupled with a strong infrastructural back-end. Testing was performed within a closed community of different users making use of the app. This led towards better understanding the requirements for future development and determining how the app can be further enhanced in specific sectors. The work can be further extended by adding new features for facilitating communication between the tutor and user by providing a messenger system. Moreover, to better keep account of user enrolled tuition, a timetable-like structure can be implemented to track the time each tuition is held. Similarly, tutor profiles can be populated with timetables showing remaining student availability for each tuition, along with the time each tuition is held during the week.

6. DISTRIBUTION OF WORK

Ryan Camilleri:

- Front-end screens
- Back-end streams & data flow
- Firebase Database Management
- Recommender System
- Video Demonstration
- Report

Francesco Borg Bonello:

- Google Maps API
- Tuition Search & Filtering
- Front-end map screen
- Presentation Slides
- Report

REFERENCES

[1] Moroney, L., 2017. The Firebase Realtime Database. In: The Definitive Guide to Firebase., Apress, Berkeley, CA.

[2] Alsalemi, A. et al., 2017. Real-Time Communication Network Using Firebase Cloud IoT Platform for ECMO Simulation. 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData).

[3] Ohyver, M., Moniaga, J., Sungkawa, I., Subagyo, B. and Chandra, I., 2019. The Comparison Firebase Realtime Database and MySQL Database Performance using Wilcoxon Signed-Rank Test. Procedia Computer Science, 157, pp.396-405.

[4] Stonebraker, M., 2010. SQL databases v. NoSQL databases. Communications of the ACM, 53(4), pp.10-11.

[5] Khawas, C. and Shah, P., 2018. Application of Firebase in Android App Development-A Study. International Journal of Computer Applications, 179(46), pp.49-53.

[6] O'connor, P., 2008. User-generated content and travel: A case study on Tripadvisor. com., pp.47-58.

[7] Hu, S. and Dai, T., 2013. Online Map Application Development Using Google Maps API, SQL Database, and ASP.NET. International Journal of Information and Communication Technology Research, 3(3), pp.102-110.

[8] Dong, L., Hai, T. and Gaoshi, Y., 2010. Android Based Wireless Location and Surrounding Search System Design. 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science.

[9] Martínez, G. and López, P., 2018. ADVANCED DATABASES PROJECT: REAL-TIME DATABASES AND FIREBASE. Cs.ulb.ac.be.

[10] Cubranic, D., 2020. US8195695B2 - Apparatus and method for categorical filtering of data - Google Patents. Patents.google.com.

[11] Schiaffino, S. and Amandi, A., 2009. Intelligent User Profiling. Lecture Notes in Computer Science, pp.193-216.

[12] Middleton, S., Shadbolt, N. and De Roure, D., 2003. Capturing Interest Through Inference and Visualization: Ontological User Profiling in Recommender Systems. Proceedings of the international conference on Knowledge capture - K-CAP '03, pp.62-68.

[13] Schafer, J., Konstan, J. and Riedi, J., 1999. Recommender systems in e-commerce. Proceedings of the 1st ACM conference on Electronic commerce - EC '99, pp.158-165.

[14] Google Developers. 2020. Google Developers: Android Studio. Retrieved from: https://developer.android.com/studio

[15] 2020. Flutter. Retrieved from: https://flutter.dev/

[16] 2020. Dart. Retrieved from: https://dart.dev/

[17] Flutter. 2020. Flutter: StreamBuilder Widget. Retrieved from: https://api.flutter.dev/flutter/widgets/StreamBuilder-class.html

[18] Firebase. 2020. Firebase: Cloud Firestore. Retrieved from: https://firebase.google.com/docs/firestore

[19] flutter.dev, 2020. google_maps_flutter | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/google_maps_flutter

[20] baseflow.com, 2020. geolocator | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/geolocator

[21] Anon, 2020. Marker class - google_maps_flutter library - Dart API. Pub.dev. Retrieved from: https://pub.dev/documentation/google_maps_flutter/latest/google_maps_flutter/Marker-class.html

[22] akshathjain.com, 2020. sliding_up_panel | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/sliding_up_panel

[23] Tracey, T. and Robbins, S., 2005. Stability of interests across ethnicity and gender: A longitudinal examination of grades 8 through 12. Journal of Vocational Behavior, 67(3), pp.335-364.

FLUTTER PACKAGE REFERENCES

Firebase Authentication package for registration and login
flutter.dev, 2020. firebase_auth | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/firebase_auth

Cloud Firestore package for cloud-hosted, noSQL database support
flutter.dev, 2020. cloud_firestore | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/cloud_firestore

Provider package for dependency injection and state management
dash-overflow.net, 2020. provider | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/provider

A collection of loading indicators animated with flutter
jeremiahogbomo@gmail.com, 2020. flutter_spinkit | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/flutter_spinkit

Flutter package for creating selectable, input chip tags
gyorgio88@gmail.com, 2020. flutter_tags | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/flutter_tags

Form package to remove the boiler plate when creating material forms
danvickmiller@gmail.com, 2020. flutter_form_builder | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/flutter_form_builder

Package for celebrating in-app achievements using confetti
funwith.app, 2020. confetti | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages?q=confetti

Font Awesome Icon pack providing additional icons
fluttercommunity.dev, 2020. font_awesome_flutter | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/font_awesome_flutter

Package for providing a rating slider for displaying rating and reviewing
infimatet@gmail.com, 2020. smooth_star_rating | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/smooth_star_rating

Package for integrating Google Maps in iOS and Android applications
flutter.dev, 2020. google_maps_flutter | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/google_maps_flutter

Package for providing a cross-platform API for generic location functions
baseflow.com, 2020. geolocator | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/geolocator

Package providing a material design slider used in map range slider
flutterdev.site, 2020. flutter_xlider | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/flutter_xlider

Package providing a draggable flutter widget used in search view
akshathjain.com, 2020. sliding_up_panel | Flutter Package. Dart packages. Retrieved from:https://pub.dev/packages/sliding_up_panel

Package providing a carousel slider widget used in the homepage
serenader.me, 2020. carousel_slider | Flutter Package. Dart packages. Retrieved from: https://pub.dev/packages/carousel_slider