AN INTELLIGENT MOBILE APPLICATION TO CONNECT TENNIS PLAYERS BY SKILL, LOCATION, AND AVAILABILITY USING FLUTTER AND FIREBASE

Mingcheng Tian[1], Rodrigo Onate[2]

[1]Santa Margarita Catholic High, 22062 Antonio Parkway, Rancho Santa Margarita, CA 92688
[2]California State University, Fullerton, 800 N State College Blvd, Fullerton, CA 92831

Abstract

Tennis players often struggle to find compatible people to play with because they need someone nearby, at a similar skill level, and available at the same time. This project proposes a mobile application that helps players find rally partners through structured profiles and search filters. The app was built using Flutter for the interface and Firebase for authentication and profile storage [11]. The main components are the profile system, search and filter system, and settings system.

Users can store details such as UTR rating, location, gender, preferred courts, playing hand, available days, and available times. Challenges included keeping profile data consistent across settings, search, and profile displays. Experiments tested search accuracy and availability updates. Results showed strong performance overall, although city and availability formatting affected some results. This idea is useful because it gives tennis players a faster and more organized way to connect.

KEYWORDS

Tennis, Community, Social, Exercise, Mobile Application, Flutte

1. Introduction

Many tennis players struggle to find people to play with who match their skill level, location, and schedule. This problem is important because tennis is connected to many health benefits, including improved aerobic fitness, lower body fat, and stronger cardiovascular health [5]. Tennis is different from many other fitness activities because it usually requires another person. A player may have access to courts and equipment, but still be unable to practice if they cannot find a reliable partner. This is especially difficult for beginners, students, competitive juniors, adults with busy schedules, and people who recently moved to a new area. Research has also found that tennis and other racquet sports are associated with longer life expectancy, which makes consistent access to playing partners even more valuable [6].

of athletic outcomes, with training intensity alone showing a strong statistical association with performance ($\beta = 0.55$, $p < 0.001$) [2]. Yet unlike heart rate or motion, breathing mechanics such as breathing rate, chest expansion, and breathing effort are rarely tracked outside laboratory or

David C. Wyld et al. (Eds): BMLI, CACIT, EDUAIT, InWES – 2026
pp. 43-67, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.161805

professional settings, since instruments capable of measuring them, such as spirometers and metabolic carts, are costly and impractical for daily training use. This problem disproportionately affects high school and adolescent athletes, who are actively developing lifelong training habits but typically have no access to school-level or professional-level respiratory monitoring, unlike college and elite programs that increasingly deploy dedicated performance-monitoring infrastructure [3]. Left unaddressed, this gap may cause inefficient training, unrecognized dyspnea, or early disengagement from sport among a population of hundreds of thousands of young athletes each year.

The background of this problem comes from the way tennis communities are usually organized. Players often depend on school teams, clubs, group chats, coaches, or word of mouth to find matches. These methods can work, but they are often limited, unorganized, and not always available to everyone. Tennis has also become more popular in the United States, with over 23 million people participating according to the Sports & Fitness Industry Association [1]. As more people play, there is a greater need for easier ways to connect players.

This problem is important because regular practice helps players improve, stay active, and remain interested in the sport. In the long run, difficulty finding partners can cause people to play less often, improve more slowly, or stop playing altogether.

Location-based recommendation systems try to suggest useful nearby places or users based on location and behavior [12]. Their weakness is that they usually focus on places, not player compatibility. My project improves this by including tennis-specific factors such as UTR, courts, and schedule.

Online matchmaking systems try to pair players by skill level and wait time. They work well for games, but they do not handle real-world needs like travel distance or court availability. My app adds location and availability.

Stable matching systems try to create fair pairings based on preferences. However, they usually require fixed rankings and one-to-one matches. My app improves this by allowing users to search flexibly and choose different partners.

My solution is an intelligent mobile application that helps tennis players find local rally partners using skill, location, gender, and availability filters. Users can create profiles with details such as UTR rating, preferred courts, playing hand, available days, available times, gender, age, and location. The app is built with Flutter, which supports cross-platform mobile application development from one codebase [7].

This app solves the problem by organizing important player information in one place. Instead of relying on random texts, group chats, or asking around at courts, users can search for players who match what they need. Firebase Authentication supports this system by connecting each profile to a secure signed-in user account [8]. For example, someone can look for a nearby player with a similar UTR rating who is available on the same day and time.

This solution is effective because it focuses on the main factors needed for a successful rally: similar skills, nearby courts, and shared availability. Hybrid recommender research shows that recommendation systems can combine different types of information, such as user preferences and item details, to make suggestions more useful [9]. It is better than traditional methods because it is faster, more organized, and helps users discover new players outside their usual

circle. Location recommendation research also shows that geographic and social information can improve suggestions for nearby places or activities [10].

The experiments tested whether the app could match tennis players accurately and keep important profile information updated. The first experiment focused on the search filter system. I created sample player profiles with known UTR ratings, cities, genders, available days, and available times. Then I tested filters such as UTR range, same city, Monday availability, morning availability, and combined filters. The strongest result was that most filters returned the expected players. The only lower result came from the city filter, which showed that location information can be inconsistent.

The second experiment tested whether availability settings worked across the app. I changed test accounts to different days and times, then checked whether those choices appeared correctly on profiles and in search results. Most tests were successful, but one account displayed correctly on the profile while failing in search. This suggested that saved data must use consistent labels.

2. Challenges

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

2.1. Improving Player Search and Filtering Accuracy.

One major component of the program is the player search and filtering system. A challenge with this component is making sure the filters return accurate results without hiding too many useful players. For example, users may search by UTR rating, distance, gender, available days, or time of day, but some players may have incomplete profiles. This could cause good matches to be excluded. To address this, the program could allow flexible filtering, where missing information does not automatically remove a player unless the filter requires it. Clear profile fields and simple filter options could also help users get better results.

2.2. Simplifying Player Availability Management

Another major component is the availability system. A challenge is making sure usercan clearly set when they are free and that other players can easily understand those times. Availability can become confusing if users enter different formats or if the app allows too many specific options. To resolve this, the program could use organized choices such as days of the week and general time periods like morning, afternoon, and evening. This would make availability easier to compare across profiles. The app could also let users update availability whenever their schedule changes.

2.3. Designing an Effective Player Profile System

A third major component is the user profile system. A challenge is making profiles helpful without making them too long or difficult to complete. If users are asked for too much information, they may skip important details, but if profiles are too simple, other players may not know enough to decide whether to connect. To solve this, the program could separate profile information into clear categories, such as personal information and tennis information. Required fields could focus on the most important matching details, while optional fields could allow users to add extra information when they want.

3. Solution

The main structure of my program is a mobile tennis matchmaking application that connects users through profiles, search filters, and availability settings. The three major components are the user profile system, the player search/discover system, and the settings/availability system. These components work together so that users can create accurate tennis profiles, find other players, and update the information used for matching.

The flow of the program begins when a user signs in or creates an account. After entering the app, the user can set up a profile with personal information such as age, gender, and location, along with tennis information such as UTR rating, preferred courts, playing hand, available days, and available times. Once this information is saved, the user can go to the discover or search sections to find other players. The search system allows filtering by UTR, distance, gender, days of availability, and time of day. When a player is selected, the app shows their full profile, including when they are available to play. The settings section allows the user to update important matching information, such as UTR rating and availability. This keeps the profile accurate over time. The program was made using Flutter for the mobile application interface and Firebase for backend services such as authentication, user data storage, and real-time profile updates.

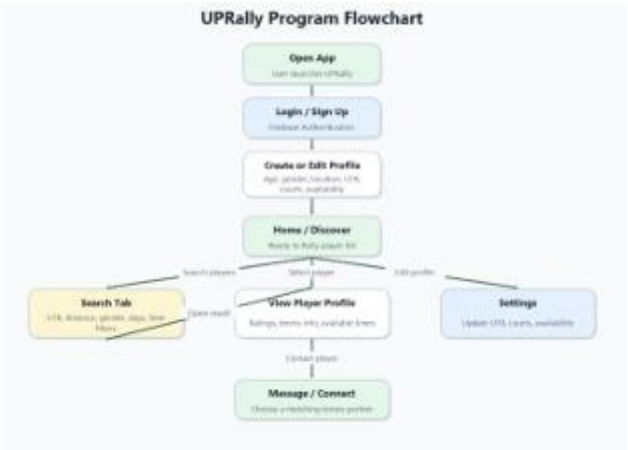

Figure 1. Overview of the solution

The profile system stores and displays important information about each tennis player. Its purpose is to help users understand another player's background, skill level, location, court preferences, and availability. This component uses Firebase to save user data and Flutter widgets to display it. It relies on authentication because each profile must connect to a specific user account. F

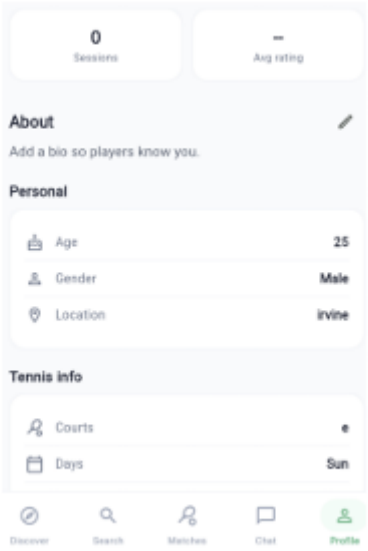

Figure 2. Player profile page displaying user data from Firestore

```
final user = FirebaseAuth.instance.currentUser;
if (user == null) return const SizedBox.shrink();

return StreamBuilder<DocumentSnapshot<Map<String, dynamic>>>(
  stream: FirebaseFirestore.instance
      .collection('users')
      .doc(user.uid)
      .snapshots(),
  builder: (context, snapshot) {
    final data = snapshot.data?.data() ?? const <String, dynamic>{};
    final name = data['name'] as String? ?? user.displayName ?? 'Player';
    final city = data['city'] as String? ?? 'Location not set';
    final bio = data['bio'] as String? ?? 'Add a bio so players know you.';
    final gender = data['gender'] as String? ?? 'Not shared';
    final handedness = data['handedness'] as String? ?? 'Not shared';
    final courts = data['favoriteCourts'] as String? ?? 'Not set';
    final photoUrl = data['photoUrl'] as String? ?? '';
    final availableDays = _stringList(data['availableDays']);
    final availableTimes = _stringList(data['availableTimes']);
    final birthday = data['birthday'] is Timestamp
        ? (data['birthday'] as Timestamp).toDate()
        : null;
```

Figure 3. Real-time profile data retrieval using StreamBuilder

This code runs when the profile page loads or updates. Its main purpose is to get the current user's profile information from Firebase and display it in the app.

First, the build methodchecks who is signed in using FirebaseAuth.instance.currentUser. If no user is logged in, it returns an empty widget called SizedBox.shrink(), so the app does not try to show a profile that does not exist.

Next, the code uses a StreamBuilder. This listens to the signed-in user's document inside the Firestore users collection. The document is found using user.uid, which is the user's unique ID. The .snapshots() method means the profile updates in real time whenever the database changes.

Inside the builder, the code gets the Firestore data with snapshot.data?.data(). If no data is found, it uses an empty map [13]. Then it creates variables such as name, city, bio, gender, handedness, courts, and photoUrl. Each variable gets information from Firebase, but also has a default value if the field is missing.

The availableDays and availableTimes variables convert saved Firebase lists into normal string lists. The birthday variable converts a Firebase timestamp into a Dart date, and the age variable calculates the user's age from that birthday.

The search and filter system helps users find tennis players who match what they are looking for. Its purpose is to narrow down players by UTR rating, distance, gender, availability days, and time of day. This component uses Flutter for the interface and Firebase data from player profiles. It relies on filtering logic, which compares each player's saved information to the user's selected filters.

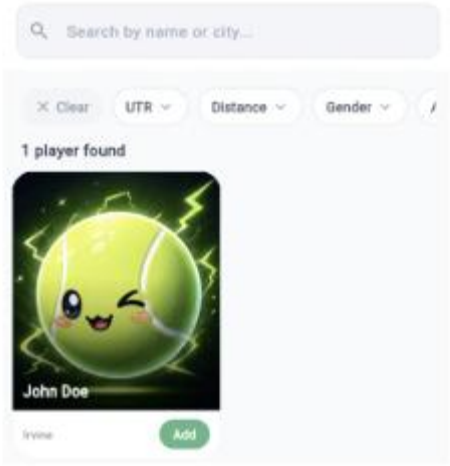

Figure 4. Search and filter interface for player discovery

```
List<UserModel> get _filtered {
  return _users.where((u) {
    final matchesQuery =
        _query.isEmpty ||
        u.name.toLowerCase().contains(_query.toLowerCase()) ||
        u.city.toLowerCase().contains(_query.toLowerCase());
    if (!matchesQuery) return false;
    if (_utrFilterActive) {
      final rating = u.selfReportedRating;
      if (rating <= 0 || rating < _utrRange.start || rating > _utrRange.end)
        return false;
      }
    }
    if (_distanceFilter == 'Same city') {
      if (_currentCity.trim().isEmpty ||
          u.city.toLowerCase() != _currentCity.toLowerCase()) {
        return false;
      }
    }
    if (_genderFilter != 'Any' &&
        u.gender.toLowerCase() != _genderFilter.toLowerCase()) {
      return false;
```

Figure 5. Filtering logic for UTR, distance, gender, and availability

This code runs on the Search page whenever the player list needs to be displayed or updated. The _filtered method takes all loaded users and returns only the players who match the current search

and filter choices. First, it checks matchesQuery, which compares the search text to each player's name and city. If the player does not match the search, the method returns false, removing them from the results.

Next, the code checks the UTR filter. It creates a rating variable from the player's saved selfReportedRating and makes sure it falls inside the selected UTR range. Then it checks distance by comparing the player's city to the current user's city. After that, it checks gender.

The _matchesAvailability method checks whether the player is available on the selected day. The _matchesTimeOfDay method checks whether the player is available during the selected time, such as morning, afternoon, or evening. Firebase provides the user profile data, and this code filters it inside the app.

The settings and availability system let users update the information used on their profile and in search filters. Its purpose is to keep details like available days, available times, courts, and UTR rating accurate. This component uses Flutter form widgets and Firebase Authentication/Firestore. Authentication connects updates to the correct user account, while Firestore stores the saved profile data.

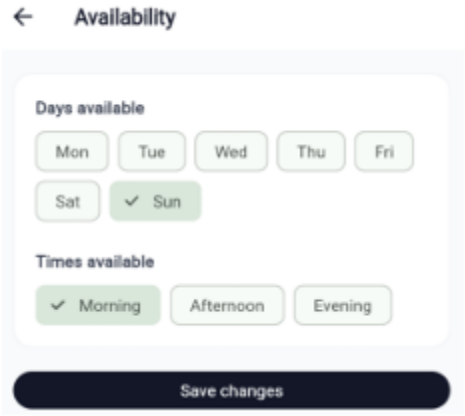

Figure 6. Availability settings interface

Figure 7. Availability day/time selection using FilterChip widgets

This code runs when the user opens the availability settings page. It builds the part of the settings screen where users choose when they are available to play tennis. The main widget shown is a Column, which stacks the availability sections vertically.

First, the code displays the label Days available. Then it creates a Wrap widget, which places the day buttons neatly across multiple lines if needed. The list ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'] is mapped into FilterChip widgets. Each chip checks whether _availableDays contains that day. When a chip is selected or unselected, setState updates _availableDays.

Next, the code repeats the same idea for Times available. It creates chips for Morning, Afternoon, and Evening. These update _availableTimes. Later, this data is saved to Firebase so profiles and search filters use the updated availability.

4. Experiment

4.1. Experiment 1

A possible blind spot in my program is whether the search and filter system always returns the correct tennis players. This is important because users depend on the filters to find players who actually match their UTR level, location, gender preference, and availability.

To test this, I would create a controlled group of sample player profiles with known information. Each sample profile would include a name, UTR rating, city, gender, available days, and available times. This would act as my control data because I would already know which players should appear for each filter combination before running the experiment. For example, I could create one player available on Monday morning with a 4.5 UTR, another available on Tuesday evening with a 6.0 UTR, and another in a different city. Then I would use the app's search filters and compare the actual results to the expected results. I would test filters one at a time, then test combinations such as UTR plus availability or city plus time of day. This setup is useful because it tests the real purpose of the search system: showing the right players and hiding the wrong ones.

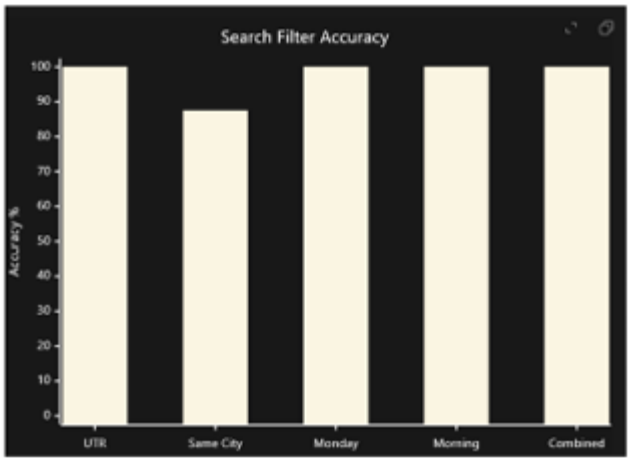

Figure 8. Search filter accuracy across test conditions

The accuracy values from the experiment were 100%, 87.5%, 100%, 100%, and 100%. The mean accuracy was 97.5%, which shows that the search filter system performed well overall. The

median was 100%, because most of the test cases returned perfect results. The lowest value was 87.5% for the Same City filter, while the highest value was 100% for the UTR, Monday availability, morning availability, and combined filter tests.

The most surprising result was the Same City filter because it did not return every expected player. I expected this filter to be simple, but location data can be inconsistent. For example, one profile might list a city with different capitalization, spacing, or a nearby neighborhood instead of the exact city name. This could cause the app to miss a valid match. The biggest effect on the results is the quality and consistency of profile data. Even if the filtering logic works correctly, inaccurate or incomplete user profiles can lower the final accuracy.

4.2. Experiment 2

A second possible blind spot is whether users can correctly update their availability in settings and have it appear accurately on their profile. This is important because availability affects both profile viewing and search results.

To test this, I would create several test accounts and update each account's availability settings with different days and times. For example, one account could be set to Monday and Wednesday mornings, another to Friday evenings, and another to Saturday afternoons. After saving the settings, I would check the profile page to see whether the selected availability appeared correctly. I would also test whether those users appeared in search results when filtering by the same days and times. The control data would come from the exact availability choices entered into the settings page. This setup is useful because it tests the full path of the feature: user input, saving to Firebase, loading on the profile, and filtering in search.

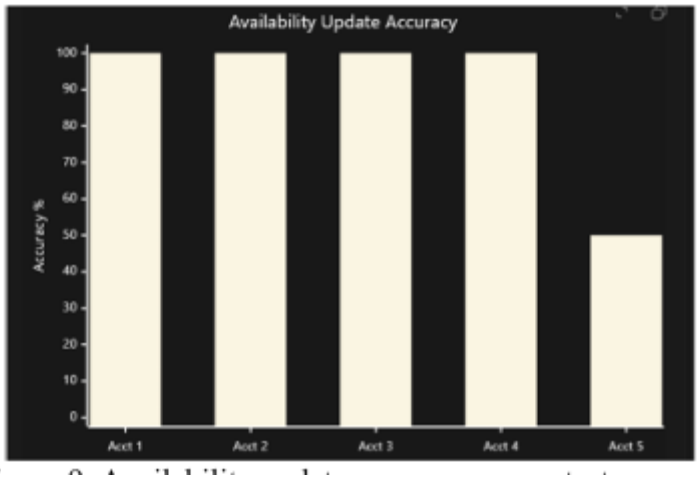

Figure 9. Availability update accuracy across test accounts

The accuracy values were 100%, 100%, 100%, 100%, and 50%. The mean accuracy was 90%,and the median was 100%. The lowest value was 50%, while the highest value was 100%. The surprising result was Account 5 because the availability displayed correctly on the profile but did not appear correctly in search. This may have happened because the profile and search system depend on the same saved data being formatted consistently. If "Sunday" was saved differently than the search filter expected, such as Sun versus Sunday, the profile could still display it while the filter would not match it. The biggest effect on the results is consistency between settings, Firebase storage, profile display, and search filtering.

5. Related Work

One scholarly source is Sánchez and Bellogín's survey, "Point-of-Interest Recommender Systems Based on Location-Based Social Networks" [2]. This source studies systems that recommend places by using location, user behavior, and social data. The solution works by analyzing where users are, where they have been, and what similar users prefer. This is effective because location is very important when recommending something nearby. However, its limitation is that it focuses mostly on places, not people. It may ignore personal schedule, skill level, and whether two users are actually compatible. My project improves on this by matching tennis players using location plus UTR, availability, gender, and profile information.

One scholarly source is Park's study on adaptive matchmaking systems [3]. This source studies how matchmaking systems can recommend people by using both explicit preferences, which are choices users directly enter, and implicit preferences, which are patterns learned from user behavior. The solution works by adjusting how much importance different profile traits receive over time. This is effective because users do not always know exactly what they want at first. However, its limitation is that it focuses on general matchmaking and does not consider tennis specific needs. My project improves this by using UTR, location, availability, and court preferences.

One scholarly source is Pizzato et al.'s study on people-to-people recommendation systems [4]. This source explains reciprocal recommenders, which are systems where a match must make sense for both users, not just one. This solution works by considering whether two people are likely to be interested in each other, instead of recommending a person the same way an app recommends a product. It is effective because social matching depends on mutual interest. However, its limitation is that the study focuses mostly on online dating, not sports. My project improves on this by applying people-to-people matching to tennis and adding UTR, location, court preferences, and availability

6. Conclusions

One limitation of my project is that it depends on users entering accurate profile information. If a user does not update their UTR, location, or availability, the search results may not be fully accurate [14]. Another limitation is that the app currently uses broad availability options, such as days of the week and morning, afternoon, or evening, instead of exact times. This makes the app simple to use, but it may not be detailed enough for scheduling real matches. The project could also improve by adding stronger messaging, notifications, and court availability features.

If I had more time, I would implement exact time scheduling so users could choose specific time blocks. I would also add automatic reminders and push notifications when someone sends a match request. Another improvement would be better location matching maps or distance calculations instead of only city-based comparisons. These changes would make the app more accurate, useful, and reliable.

Overall, this project shows how a mobile app can make tennis more accessible by helping players find compatible partners [15]. By combining profiles, UTR ratings, location, and availability, the app creates a more organized way for players to connect and schedule rallies.

REFERENCES

[1] Liang, Guoli, et al. "Failure and revival: physical education and youth sport in China." Asia Pacific Journal of Sport and Social Science 1.1 (2012): 48-59.

[2] Sánchez, Pablo, and Alejandro Bellogín. "Point-of-interest recommender systems based on location-based social networks: a survey from an experimental perspective." ACM Computing Surveys (CSUR) 54.11s (2022): 1-37.

[3] Park, Yoon-Joo. "An adaptive match-making system reflecting the explicit and implicit preferences of users." Expert Systems with Applications 40.4 (2013): 1196-1204.

[4] Pizzato, Luiz, et al. "Recommending people to people: the nature of reciprocal recommenders with a case study in online dating." User Modeling and User-Adapted Interaction 23.5 (2013): 447-488.

[5] Yang, Fanghuan, and Yi-Sub Kwak. "A study of the tennis trust-based system on the psychology and cognition of amateur tennis enthusiasts." Behavioral Sciences 14.11 (2024): 1019.

[6] Pluim, Babette M., et al. "Health benefits of tennis." British journal of sports medicine 41.11 (2007): 760 Nanavati, Jay, et al. "Critical review and fine-tuning performance of flutter applications." 2024 5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI). IEEE, 2024.

[7] Moroney, Laurence. "Using authentication in firebase." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Berkeley, CA: Apress, 2017. 25-50.

[8] Burke, Robin. "Hybrid recommender systems: Survey and experiments." User modeling and user-adapted interaction 12.4 (2002): 331-370.

[9] Ye, Mao, Peifeng Yin, and Wang-Chien Lee. "Location recommendation for location-based social networks." Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems. 2010.

[10] Tashildar, Aakanksha, et al. "Application development using flutter." International Research Journal of Modernization in Engineering Technology and Science 2.8 (2020): 1262-1266.

[11] Park, Moon-Hee, Jin-Hyuk Hong, and Sung-Bae Cho. "Location-based recommendation system using bayesian user's preference model in mobile devices." International conference on ubiquitous intelligence and computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[12] Kesavan, Ram, et al. "Firestore: The nosql serverless database for the application developer." 2023 IEEE 39th international conference on data engineering (ICDE). IEEE, 2023.

[13] Kurtz, Jennifer A., et al. "Ability for tennis specific variables and agility for determining the Universal Tennis Ranking (UTR)." Sport Journal 24 (2019).

[14] [14]Islam, Rashedul, Rofiqul Islam, and Tohidul Mazumder. "Mobile application and its global impact." International Journal of Engineering & Technology 10.6 (2010): 72-78.

©2026 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.