ISSN: 2752-3829 Vol. 3 No.2, (December, 2023)

Stochastic Modelling and Computational Sciences

IMPLEMENTATION OF REQUIREMENT BASED SERVICE PROVIDING SYSTEM

Avanti Banginwar[1], Shashikant V. Athawale[2], Soham Bhirange[3], Pranav Bokade[4] and Aditya Dayal[5]
[1,345]Research Scholar and [2]Associate Professor, Department of Computer Engineering, AISSMS COE, Pune, India
Savitribai Phule Pune University

ABSTRACT

This paper proposes the development of the Requirement-Based Service Providing System (RBSPS), a digital platform designed to connect individuals with essential services during times of need. The RBSPS aims to address the growing demand for swift and efficient access to critical services in an increasingly unpredictable world. The platform caters to a diverse range of needs, encompassing medical assistance, roadside support, emergency responders, and home repairs. By functioning as a centralized hub, the RBSPS seeks to bridge the gap between individuals requiring assistance and qualified service providers equipped to address their specific needs.A sophisticated algorithm facilitates real-time service matching, connecting individuals with vetted service providers possessing the necessary expertise and resources. Geolocation integration ensures the efficient and prompt location of service providers, minimizing response times and maximizing impact.To foster transparency and accountability, the RBSPS incorporates a robust rating and review system. This system allows users to share their experiences and guide others towards reliable service providers. In this System, recommendation plays a vital role in suggesting a new user for the best service and existing user for the additional service based on their requirement. This System uses CBCF (Content Based Collaborative Filtering) algorithm for recommendation. The Requirement-Based Service Providing System aims to provide a reliable and user-friendly solution to ensure the well-being and safety of individuals in times of need.

Key-words: Recommendation, real-time service matching, geolocation integration, review and rating system, payment processing, Content Based system, Collaborative Filtering System.

I. INTRODUCTION

In today's fast-paced world, swift and reliable access to essential services during times of need is a growing concern. Enter the Requirement-Based Service Providing System (RBSPS), a digital platform that bridges the gap between individuals seeking immediate assistance and qualified service providers equipped to address their specific needs. This paper delves into the core components, functionalities, and potential impact of the RBSPS on user experience and overall service accessibility.

The Building Blocks of the RBSPS

The RBSPS operates seamlessly thanks to a powerful combination of technology solutions:

- **Flutter and Dart:** This dynamic duo forms the backbone of the RBSPS mobile application. Flutter is a cross-platform framework that empowers developers to create a single codebases for both Android and iOS devices. This translates to streamlined development and maintenance, reducing resources needed to cater to diverse user bases. Dart, an object-oriented programming language, acts as the building block for the application's logic and functionalities, providing the essential instructions for the app to run smoothly.
- **Firebase:** This comprehensive suite of tools from Google Cloud Platform offers the RBSPS critical functionalities. It provides services for:
  - **Real-time Database Management:** This ensures that user information, service provider details, and service requests are constantly updated and readily accessible, facilitating smooth operation.
  - **User Authentication:** Firebase securely manages user logins and registrations, safeguarding user data and ensuring only authorized individuals can access the system.

- ○ **Push Notifications:** This feature enables real-time communication between users and service providers. For example, users can receive notifications when their service request is accepted, and providers can be alerted about new requests.

- **Android Studio:** As the official integrated development environment (IDE) for Android app development, Android Studio provides the platform for building, testing, and deploying the RBSPS Android application. Its comprehensive tools and functionalities include code editing, debugging, and emulator functionalities, streamlining the development process and ensuring a polished and user-friendly experience for Android users.

Recommendation Engine:
A crucial aspect of the RBSPS is its recommendation engine, powered by a hybrid approach called Content Boosted Collaborative Filtering (CBCF). This innovative technique combines the strengths of two individual recommendation algorithms:

- **Content-Based Filtering (CBF):** This user-centric approach analyzes an individual's profile, including past service selections and interests, to recommend similar services they might find valuable. Unlike traditional methods, CBF operates independently of user-item interaction data, relying solely on the rich information gleaned from individual user profiles. This enables the system to personalize recommendations even for new users who haven't interacted with the platform before.

- **Collaborative Filtering (CF):** This method leverages the collective intelligence of user groups to suggest services. It analyzes user-item interaction data, such as ratings and reviews, to identify users with similar service preferences. By understanding the preferences of users with similar service history, CF can recommend services that these users have enjoyed, even if they differ from the individual's previous selections.

The Power of Combining CBF and CF:
By combining the personalized recommendations of CBF with the broader user group insights of CF, the CBCF approach delivers a more refined and accurate service suggestion experience for RBSPS users. This hybrid approach ensures that users receive recommendations tailored to their individual needs while also benefiting from the collective wisdom of the user community.

The RBSPS, with its innovative combination of technology and user-centric design, holds immense potential to revolutionize service accessibility and user satisfaction. By fostering seamless connections between individuals and service providers, it can empower individuals to easily find the help they need, when they need it most. Additionally, the system can strengthen the service ecosystem by promoting efficiency and effectiveness, allowing providers to reach a wider clientele and fulfill service requests in a timely manner. As the system evolves and gathers more data, the recommendation accuracy of the CBCF algorithm will continue to improve, solidifying the RBSPS as a valuable tool for navigating the ever-changing service landscape.

II. RELATED WORK

A. Design and Development of Mobile Healthcare Application prototype using flutter. [1]

The core components used to develop a reliable and responsive mobile application are explained in this paper. This paper gives insights of language and frameworks used to implement the mobile application. Flutter and Dart can be used for implementing the frontend of the application while coding it on Android Studio which is the official IDE for android development. Compared to Native app development languages like React Native or Kotlin, Flutter Provides one language solution for developing an app compatible on all platforms.

B. A Service-Based Recommendation System to Assist Decision Making in a Small and Medium Company [2].

The paper proposes a platform to facilitate resource sharing between collaborative companies, through a service-based recommendation system. The recommendation system uses a vector space model to represent the services and the user requests, and applies three similarity measures (Jaccard, Dice, and Cosine) to generate a list of similar services that match the user's needs. The paper also introduces a decision support module that allows the user to evaluate the recommended services, negotiate with the service providers, and give feedback to the system. The paper evaluates the performance of the three similarity measures using several criteria, such as precision, recall, F-measure, and user satisfaction. The results show that Cosine similarity produces the best results in terms of recall and prediction errors.

C. Recommendation Systems with Machine Learning [3]

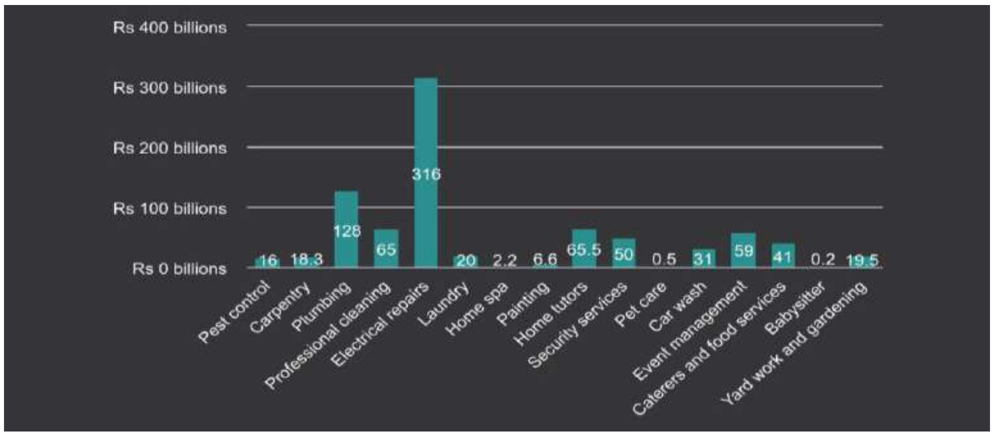
The paper presents the development and comparison of two movie recommendation systems, using different machine learning algorithms and methods. The first system is based on collaborative filtering, using matrix factorization in ML.NET framework2. It predicts the ratings of movies for users based on their past ratings and the ratings of other users with similar preferences. The second system is based on a hybrid approach, using Matchbox recommender in Azure Machine Learning Studio3. It predicts the ratings of movies for users based on their past ratings, the ratings of other users, and the genres of the movies. The paper evaluates the performance of the two systems using the MovieLens 20M dataset, which contains 20 million movie ratings and 465,000 tag applications 45. It uses the root mean squared error (RMSE) as the evaluation metric. The paper finds that the collaborative filtering system achieves a lower RMSE value than the hybrid system, but the hybrid system solves the cold start problem and can make use of additional features. The paper suggests that the best recommendation method depends on the type of data and the problem to be addressed.

D. A Novel Gamification Approach to Recommendation Based Mobile Applications [4]

The paper proposes a novel gamification approach to enhance the usability and usefulness of recommendation based mobile applications, such as e-commerce, travel, or education apps. The paper uses content-boosted collaborative filtering (CBCF), a hybrid of content-based and collaborative filtering, to generate personalized recommendations for users based on their preferences and ratings.

E. Systematic Review on Recommendation Systems [5]

The paper is about recommendation systems, which are systems that suggest relevant items to users based on their preferences, behavior, or ratings. The paper reviews the traditional recommender approaches, such as content-based filtering, collaborative filtering, and hybrid filtering, and discusses their advantages and disadvantages. The paper also presents a proposed approach based on item-item similarities, which uses cosine similarity or conditional probability to measure the similarity between items and recommend items that are similar to the user's previous purchases or interests. The paper evaluates the performance of the proposed approach using various metrics, such as root mean square error, hit rate, coverage, and average reciprocal hit rank, and compares it with the user-based recommendation approach. The paper concludes that the proposed approach provides more accurate and efficient recommendations than the user-based approach, and can handle the issues of sparsity, scalability, and offline calculation.



Graph 1: Sales of Most Popular Services By 2024

III. METHODOLOGY

The journey of crafting a user-centric application begins with a deep dive into user needs. This involves meticulous requirements gathering, where the desired services and expectations are meticulously identified. Next, a comprehensive plan and design phase takes shape, meticulously outlining the application's architecture, user interface, and feature set. The development stage then comes alive, utilizing the most suitable technology stack and adhering to established best practices. To ensure a seamless user experience, rigorous testing becomes paramount, encompassing various methodologies like unit testing, integration testing, and user testing itself. The process doesn't stop there. By incorporating a continuous feedback loop and an iterative approach, the application undergoes constant refinement, guaranteeing that it consistently delivers the intended services and remains perfectly aligned with user expectations.

Phase 1: Component selection for application development

Crafting an application tailored to user needs necessitates a structured, multi-stage process. The initial stage, requirements gathering, lays the foundation by thoroughly understanding the target audience. This crucial step involves engaging stakeholders and potential users through various methods, such as interviews, surveys, and user research. By actively listening to their needs, expectations, and pain points, developers gain invaluable insights that ensure the final application effectively addresses user demands. All this crucial information is meticulously documented to serve as a roadmap throughout the development process, guaranteeing the final product aligns seamlessly with user desires.

Moving forward, the planning and design stage takes center stage. During this crucial phase, the app's architectural blueprint is meticulously crafted, outlining how data will be stored, processed, and accessed. This stage also focuses on crafting the user interface (UI) and user experience (UX), ensuring the application is not only visually appealing but also intuitive and user-friendly. Additionally, this stage involves defining the specific features the app will offer, often visualized through wireframes or prototypes that showcase the app's intended flow and functionality. By meticulously planning and designing the application at this stage, developers can avoid potential roadblocks and ensure a smooth development process that prioritizes user needs and expectations.

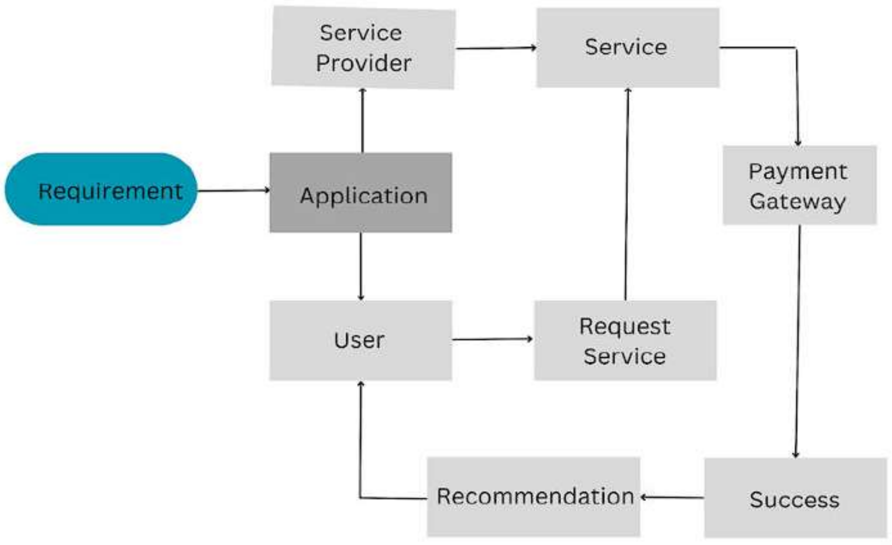


Figure 1: System Architecture Diagram

Phase 2: System and software requirement and installation

The development environment should be equipped with a modern operating system, such as Windows, macOS, or Linux, to support Flutter and Firebase. A development machine with a minimum of 4GB RAM and a multi-core processor is recommended for smooth operation.

Install the Flutter software development kit, which includes the Flutter framework and Dart programming language. Choose an IDE for Flutter development, with popular options being Android Studio, Visual Studio Code, or IntelliJ IDEA, each of which supports Flutter plugins for efficient coding, debugging, and testing.

To integrate Firebase services seamlessly, install the FlutterFire plugins. These plugins include Firebase Authentication, Cloud Firestore, and Cloud Messaging. Configure the Firebase project with the desired authentication methods, database, and cloud messaging settings. Install the Firebase CLI to interact with Firebase services from the command line. The CLI allows for features like deploying applications and managing Firebase projects.

Phase 3: Prototyping and developing system architecture

The application prototype acts as a vital first step, showcasing the core functionalities and user interface design. This preliminary model focuses on key features and user interactions, providing a clear picture of the app's flow and layout. The prototype prioritizes user-friendliness and intuitiveness, allowing users to navigate through essential actions like registration, authentication, service selection, and even viewing personalized recommendations based on their preferences and behavior. Users can interact with the prototype to simulate the process of requesting services and receiving these recommendations.

Visually, the interface will incorporate essential components like navigation bars for seamless movement within the app, forms for user input to gather necessary information, and service listings showcasing available options. These components will be built using the Flutter framework. The prototype will also simulate user authentication, highlighting the security and personalization aspects of the application. Additionally, it will showcase the integration of Firebase for backend services, demonstrating functionalities like real-time data synchronization, data storage, and user authentication. Finally, the prototype will incorporate a recommendation system, offering users suggestions based on their needs and preferences.

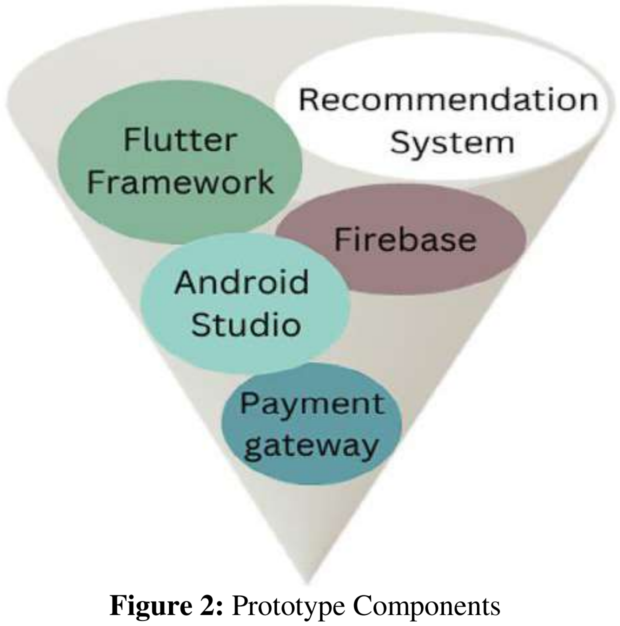


Figure 2: Prototype Components

Phase 4: Integrating machine learning for recommendation and application development

1. Machine Learning Training:
The journey begins with training and fine-tuning machine learning models using relevant data. This data could encompass user interactions, preferences, service details, or any other information crucial for generating recommendations. By meticulously analyzing this data, the models learn to identify patterns and relationships, allowing them to make insightful predictions.

2. Model Deployment:
Once trained, these models are then serialized, essentially converting them into a format suitable for deployment on a backend server. This makes them accessible to the application and enables them to analyze user data and generate recommendations in real-time.

3. Recommendation Engine:
A powerful recommendation engine lies at the heart of this integration. One approach, Content-Based Collaborative filtering, leverages the combined strengths of two established recommendation techniques:

Content-Based Filtering: This method analyzes a user's past interactions and preferences to suggest similar services. For instance, if a user frequently utilizes a particular service category, the system might recommend other services within the same category based on shared attributes.

Collaborative Filtering: This method leverages the collective wisdom of the user base. By analyzing the ratings and reviews provided by other active users, the system can identify trends and recommend services that have received positive feedback from users with similar preferences.

By combining these techniques, Content-Based Collaborative Filtering can deliver highly personalized recommendations that cater to individual user needs and preferences, significantly enhancing the user experience and increasing the likelihood of user engagement with the application.

This paper uses the Hybrid recommendation technique i.e Content Boosted Collaborative Filtering. It has a power of both the techniques which helps to increase the accuracy of recommendation.

Algorithm for Content-based using cosine similarity which is used for recommending similar products depending on similar users:

1. Import the necessary libraries: numpy and cosine_similarity from sklearn.metrics.pairwise.

2. Define the data: users, products, and ratings.

3. Convert the ratings to a numpy array for easier computation.

4. Define a function recommend_similar_products (user, top_n) to recommend similar products for a given user.

5. Inside the function:

- Find the index of the user in the user list.
- Calculate cosine similarity between the user's ratings and all other users.
- Sort users based on similarity (excluding the user itself).
- Find products rated highly by similar users.

6. Test the recommendation function for each user:

- Call recommend_similar_products (user) for each user.
- Print the recommended similar products for each user.

Algorithm for SVD Collaborative Filtering for recommending using user's history:

1. Retrieve the shape of the dataframe df.

2. Group the dataframe by 'product_id' and count the ratings for each product, then sort them in descending order to get the most popular products.
3. Plot the top 30 most popular products in a bar chart.
4. Select the most popular product.
5. Create a ratings utility matrix from the dataframe, where rows represent users and columns represent products.
6. Transpose the ratings utility matrix.
7. Perform dimensionality reduction using TruncatedSVD with 10 components on the transposed ratings utility matrix.
8. Given a specific product ("Product16"):

a. Retrieve the index of the product.

b. Calculate the correlation of the given product with other products.

c. Select products with a correlation greater than 0.90 as recommendations.

d. Remove the given product from the recommendations.

9. Return the top 9 recommended products.

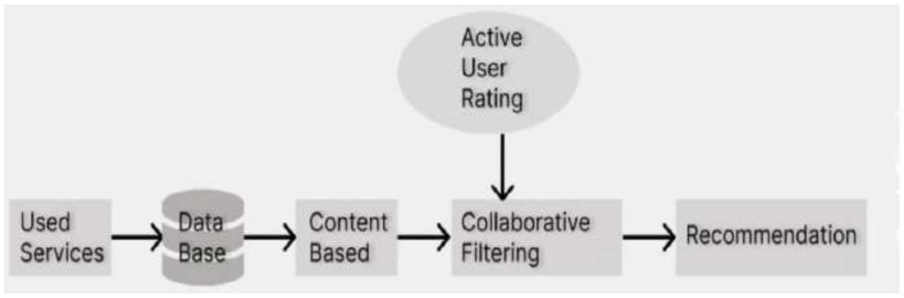


Figure 3: recommendation system workflow architecture.

Testing an example where 5 service providers are rated by 5 users,

Product_id	Cleaning1	Cleaning2	Cleaning3	Electronic1	Electronic2	Mechanic1	Mechanic2	Mechanic3
Users								
Alice	2	0	5	4	2.5	4	5	5
Bob	4	3	4	4	3.5	5	3.5	3
Charlie	4	0	5	2.5	4	0	4	2
David	3	4	4	2	0	4	3.5	4
Eve	0	5	2	2	4	4	2	5
Frank	4	3	0	4.0	3.0	4	3.5	3

Table 1: example ratings given by user for different services.

Using the above algorithm, the rating table is converted into Rating Utility Matrix.

Users	Alice	Bob	Charlie	David	Eve	Frank
product_id						
Cleaning1	2.0	4.0	4.0	3.0	0.0	4.0
Cleaning2	0.0	3.0	0.0	4.0	5.0	3.0

Cleaning3	5.0	4.0	5.0	4.0	2.0	0.0
Electronic1	4.0	4.0	2.5	2.0	2.0	4.0
Electronic2	2.5	3.5	4.0	0.0	4.0	3.0
Mechanic1	4.0	5.0	0.0	4.0	4.0	4.0
Mechanic2	5.0	3.5	4.0	3.5	2.0	3.5
Mechanic3	5.0	3.0	2.0	4.0	5.0	3.0

Table2: rating utility matrix.

The final recommendation list is updated according to the correlation matrix in which the rating value above 0.95 is selected. Below is shown the related array with different services and their recommendation.

Booked Services	**Correlation array**	**Recommended services**
Electronic2	array([0.99881155, 0.78083476, 0.81342701, 0.99960138, 1, 0.98046891, 0.96741874, 0.98997226])	['Cleaning1','Electronic1', 'Mechanic1']
Cleaning1	array([1 , 0.74945771, 0.84081014, 0.99978943, 0.99881155, 0.96971798, 0.97860883, 0.99568068])	['Electronic1', 'Electronic2', 'Mechanic1']
Mechanic1	array([0.96971798, 0.88845392, 0.68314105, 0.97452544, 0.98046891, 1, 0.89872971, 0.94285446])	['Cleaning1', 'Electronic1', 'Electronic2']
Electronic1	array([0.99978943, 0.76288549, 0.82952477, 1, 0.99960138, 0.97452544, 0.9741811 , 0.99356583])	['Cleaning1', 'Electronic2', 'Mechanic1']
Mechanic3	array([0.99568068, 0.68475311, 0.88743757, 0.99356583, 0.98997226, 0.94285446, 0.99348265, 1])	['Cleaning1', 'Electronic1', 'Electronic2']

Table3: recommendations and correlation array of booked services.

Phase 5: Testing and deployment

To ensure a high-quality application, rigorous testing is crucial throughout the development process. This involves verifying individual components (unit testing), ensuring seamless integration (integration testing), and gathering real user feedback (user testing). The application should be tested thoroughly for functionality, performance, security, and compatibility across various devices and platforms. For deployment on Android, a release build and an APK file (Android app package) are required using Flutter. Similarly, for iOS deployment, an IPA file (iOS App Store package) needs to be created using Flutter to compile the app for iPhone and iPad. Regardless of the platform, both thorough testing and quality assurance are essential to guarantee the app's functionality and a smooth user experience. Moreover, ongoing maintenance and updates are vital to address issues, add new features, and keep the application up-to-date with changing user needs and evolving platforms.

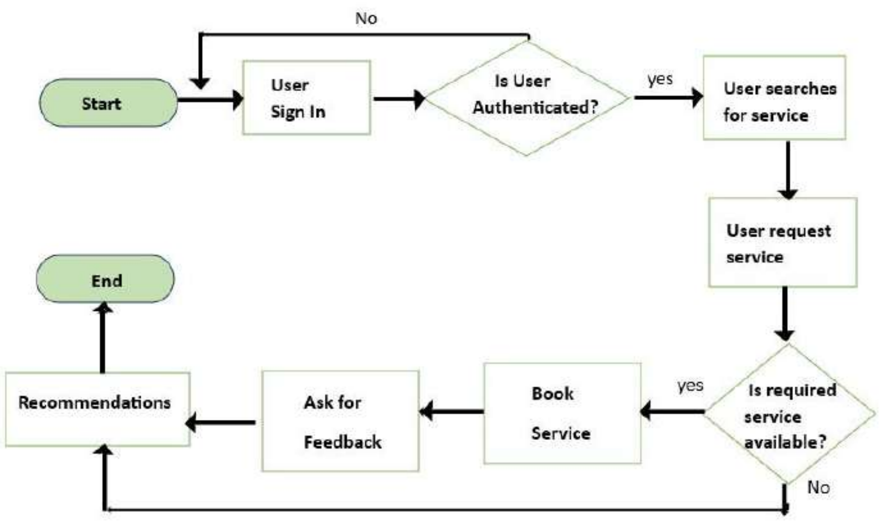


Figure 4: overall workflow of system.

IV. CONCLUSION

This application goes beyond just making it easier to find services. It also offers several valuable technical features that can benefit society. The use of machine learning, specifically Collaborative Filtering, helps personalize recommendations, making them more accurate and user-friendly. Additionally, the application utilizes cloud services like Firebase to ensure data security, real-time updates, and easy expansion, which are essential in modern app development. Furthermore, building the app with Flutter allows for cost-effective and rapid development, making it accessible to a broader audience regardless of their device or operating system.

This application's ability to gather and analyze user interactions and feedback provides valuable insights into user behavior and service preferences. These insights can be used to improve the recommendation system, making it even more precise and responsive. Additionally, the data collected can be used for data-driven decision-making, not only for individual users but also for service providers and policymakers. This data can help improve service quality, resource allocation, and pave the way for innovation.

In essence, this application combines cutting-edge technology with user-centered design, offering personalized recommendations, data security, cross-platform access, and valuable data insights. By doing so, it not only simplifies service discovery but also contributes to a more efficient, secure, and informed society, benefiting both individual users and the broader community.

V. REFERENCES

[1] Ragda Mamoun, Mohamed Nasor, Sahar H. Abulikailik "Design and Development of Mobile Healthcare Application prototype using flutter.", in the International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE) IEEE Xplore 2020.

[2] Djihene Bourenane, Noria Taghezout, Nawal Sad Houar, Lylia Abrouk "A service-based recommendation system to assist decision making in a small and medium company" in the Third International Conference on Intelligent Sustainable Systems [ICISS] IEEE Xplore 2020 .

[3] Alexandra Fanca, Adela Puscasiu, Dan-Ioan Gota, Honoriu Valean "Recommendation Systems with Machine Learning " IEEE Xplore 2021.

[4] S. Neeraj, C. Oswald, B. Sivaselvan "A Novel Gamification Approach to Recommendation Based Mobile Applications " in Ninth International Conference on Advanced Computing (ICoAC) IEEE Xplore 2017.

[5] Anam Khan, Anusha Yadav, Aman Chand Rahi, Jaya Sinha "Systematic Review on Recommendation Systems" in 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN) IEEE Xplore 2020.

[6] Kunal Shah , Akshaykumar Salunke , Saurabh Dongare , Kisandas Antala "Recommender Systems: An overview of different approaches to recommendations" in International Conference on Innovations in Information Embedded and Communication Systems (ICIIECS) 2017.

[7] Sunkuru Gopal Krishna Patro,Brojo Kishore Mishra, Sanjaya Kumar Panda, Raghvendra Kumar, Hoang Viet Long, David Taniar, Ishaani Priyadarshini "A Hybrid Action-Related K-Nearest Neighbour (HAR-KNN) Approach for Recommendation Systems" in IEEE Access 2020.

[8] Farhan Ullah , Bofeng Zhang , Rehan Ullah Khan, Tae-Sun Chung, Muhammad Attique, Khalil Khan, Salim El Khediri, Sadeeq Jan "Deep Edu: A Deep Neural Collaborative Filtering for Educational Services Recommendation" in IEEE Access 2020

[9] Zhihua Cui, Xianghua Xu, Fei Xue, Xingjuan Cai, Yang Cao, Wensheng Zhang, and Jinjun Chen "Personalized Recommendation System based on Collaborative Filtering for IoT Scenarios" in IEEE Transaction 2020

[10] Akhilesh Kumar Sharma, Bhavna Bajpai, Rachit Adhvaryu , Suthar Dhruvi Pankajkumar ,Prajapati Parthkumar Gordhanbhai, Atul Kumar "An Efficient Approach of Product Recommendation System using NLP Technique" in Elsevier 2021

[11] Phatpicha Yochum , Liang Chang , Tianlong Gu, And Manli Zhu "Linked Open Data in Location-Based Recommendation System on Tourism Domain: A Survey " in IEEE Access 2020.

[12] Yanli Guo And Zhongmin Yan " Recommended System: Attentive Neural Collaborative Filtering" in IEEE Access 2020.

[13] Suja Cherukullapurath Mana, T.Sasipraba "A Machine Learning Based Implementation of Product and Service Recommendation Models" in IEEE Xplore 2021

[14] Safia Kanwal , Sidra Nawaz, Muhammad Kamran Malik , And Zubair Nawaz " A Review of Text-Based Recommendation Systems" in IEEE Access 2021

[15] Byeongjin Choe , Taegwan Kang , And Kyomin Jung, "Recommendation System With Hierarchical Recurrent Neural Network for Long-Term Time Series" in IEEE Access 2021.