International Journal of Intelligent Information Technologies
Volume 20 • Issue 1

Event-Based Social Networking System With Recommendation Engine

G. Manikandan, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India*
https://orcid.org/0000-0001-6962-5197

Reuel Samuel Sam, College of Engineering Guindy, Anna University, India

Steven Frederick Gilbert, College of Engineering Guindy, Anna University, India

Karthik Srikanth, College of Engineering Guindy, Anna University, India

ABSTRACT

For over a decade, social networking has been ruling over the internet and plays a vital role in day-to-day life. However, for a new network to survive in this market, exclusivity is a necessity. As a result, the goal of this work is to create a network for hosting and managing volunteering and events. Furthermore, the network will feature a recommendation system to provide users with events based on their interests and preferences according to how they interact with the platform. The proposed system is exclusively meant for event post creation and management and also it focuses on event posts with interactions such as replies, likes, interest, and disinterest options. This system has been implemented and deployed with the title 'Evento' with the recommendation engine boasting an average purity index of 0.8031 for approximately 30 users. The results for recommendations have been chosen considering purity index and fisher optimization criterion metrics. Based on the experimental results, the proposed social network system with the recommendation engine has been found to be sufficient.

KEYWORDS

Clustering, Collaborative Filtering, Recommendation System, Social Networking System

1. INTRODUCTION

With the ever-growing influence of the Internet in today's society and world, the influence that Social Media has over the people cannot be understated. Many people depend on these platforms for their daily dose of news and information. Along with this, these platforms provide entertainment which further grabs hold of the user's attention. Social Networking Systems refer to some form of online community of individuals that can interact with each other by sharing of textual or pictorial information. A social network is an abstract structure consisting of people who are linked together by one or more forms of relations, such as friendship, shared information, or similar interests (Zheng et al., 2009). Social network analysis is one of the research areas that has become increasingly relevant

DOI: 10.4018/IJIIT.334232 *Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

as the amount of social network data grows. The traditional recommendation system believes that users are self-contained and spread evenly, and it ignores the social interaction and connections that exist between them (Cai et al., 2019). Users are in desperate need of recommender systems that can connect them with potential friends or interesting items. In real-world applications, social (i.e. friend) and behaviour (i.e. item) recommendation are two types of popular services (Zhao et al., 2020).

Upon examining the successful social media platforms of the recent past, it becomes evident that the success of these platforms are directly tied to need for these applications increased along with the specific function that these platforms provide. Taking Twitter, now called X, as an example, it focuses primarily on textual content. There was a need for a platform that allowed for an individual's voice to be heard and their opinion made known. Twitter met this market by providing a platform where users were able to do just this by posting their thoughts and interacting. The existing social networking platforms have a plethora of limitations associated with them. Platforms like Facebook have reached a saturation point where there is an abundance of content and functionalities giving it an extremely general view. This has in part contributed to users flocking to other platforms like Twitter and Instagram. Niche platforms are required for specific functions and tasks. This is where an event-based platform would aid users in solving their concerns regarding real world engagement through social media/online interaction.

In consideration of all this, the objective and purpose of this work is to design a Social Media platform exclusively meant for creating, attending, managing and interacting with Events. The product aims to help a wide demographic of people, from students to adults to companies. Additionally, a Recommendation System is created to cater to the interests of each individual by suggesting other users based on common interest. The Recommendation Engine follows a Collaborative Filtering approach where suggestions are made based on the user's interaction with the system. This system was selected since users seem to prefer recommendations from people they know or who are similar to them, and trust-based recommendation techniques outperform those based just on user similarity (Belkhadir et al., 2019; Golbeck, 2009).

The recommendation system proposed uses an unsupervised learning approach called K-Means algorithm. Here, for each user, an array is maintained containing values that quantifies the user's interest in each tag. These tag values are then passed as input into the clustering algorithm. The users with similar tastes are clustered and recommended to each other as those users with similar interests. Recommender systems (RSs) are software that analyses data and makes recommendations based on the user's preferences.

The rest of the paper has been organized into chapters. Section 2 covers various other related works that cover the idea of a Social Media System and a Recommendation Engine. The following chapter, Section 3, describes in detail the system design that has been followed for the implementation of the proposed work. In addition, Section 4 showcases the hardware and software specifications, experimental results obtained and analysis. Finally, Sections 5 summarizes the work conducted in this paper.

2. RELATED WORK

This section describes the other works that have focused on a form of Recommendation Engine. These papers have described in detail the types of filtering that have been considered before reaching to a conclusion on how the system is built.

Anandhan et al. (2019) presented a paper that discusses open research issues that come with social media recommender systems. They categorize the various approaches to recommending schemes. Content-Based (CB) filtering looks at the text information that the user provides. There are many methods for creating communities based on user preferences that can be automated. However, due to a lack of resources, there are drawbacks in assigning attributes. Public forums are a drawback because new users with few reviews would not be able to provide reliable recommendations. Collaborative

Filtering (CF) generates a list of suggestions based on users' previous experiences with objects. The accuracy of recommendations can be improved by calculating the similarity between users' preferences of each friend. This can also be done clustering suitable groups to calculate pairwise user similarities. However, this approach is constrained by the availability and expectations of the consumer, resulting in cold start issues and a higher level of complexity. Knowledge Based (KB) allows for the discovery of additional information about a user's interests. The user's suggestions are more precise and diverse when using KB. The restriction is focused on the specific user profile; the behaviour of other related users is not considered.

Li et al. (2019) have presented a proposal regarding a social recommendation model based on user interaction in complex social networks. Interactions on social media, according to the authors, provide important information about e-commerce as well as the expectations of target users for their recommendation framework. They introduced the SRUI model, a novel social recommendation framework that provides a foundation for improving the recommender system's performance. The SRUI model's framework is mostly finished in three phases. They work together to evaluate user behaviour in social media and determine target users' interests. Their model is primarily focused on item recommendation. To find similar users, they use weighted social network interaction and full mining routes. While the approach may be superior to other models, it is subject to user dynamic changes, which means that user experiences may change dramatically over time, necessitating a new set of recommendations.

Ben-Shimon et al. (2007) proposed a recommender system from personal social networks. They constructed the social network into a graph for each user. The user's friends are in the first layer, and suggestions are built based on the user's personal social network, which is a social tree for each user that reflects a snapshot of the entire network with user's connections. Then, using BFS, suggestions are generated by summing the impact of both positive and negative reviews from the personal social network. CF only considers the item's unique identifier, while CB only considers the attributes. To solve the drawbacks, this approach proposes a hybrid method. The model's efficiency can be improved by increasing the attenuation factor 'K.'

For heterogeneous social networks, Xiong et al. (2020) provided an effective point-of-interest recommendation paradigm. They address a POI recommendation framework that is critical in assisting users in discovering interesting places while travelling or in new areas. They propose a latent probabilistic generative model called Heterogeneous Information based LDA (HI-LDA), which can adapt to questionable relationships and boost the effectiveness of POI recommendations, especially for out-of-town visitors. By taking into consideration the information on LSBNs, they precisely capture users' inputs on CBSNs, including geographical impact as well as the excessive information on LBSNs, social connections and users' interaction and comments. The authors have proposed a geographical clustering approach via DBSCAN on popularity.

Silva et al. (2010) developed a study for a graph-based recommendation using a genetic algorithm approach. Since the algorithm is graph-based their proposed system is based on the structural properties of the social network. Characteristics such as information and metric of the social network are derived from the theory of the complex network. For recommendation the system uses the graph topology to perform filtering and ordering of nodes of the graph with relation to a given node as its recommendations and connected to it to improve the characteristics of the target node. Filtering is separation of the highly possible nodes in relation to the target node from the list of all possible nodes. Ordering is putting relevant nodes with higher priority i.e moving them higher up on the list. Relevancy is determined by considering certain properties of the target node. There are certain variables known as indexes which are derived from adjacent lists of nodes, density of nodes, etc. The Genetic Algorithm comes into play when calibrating these indexes into a single value. The final value from this algorithm will be used to determine the final list of nodes to take as recommendations.

The study by Yigit et al. (2015) proposes an extension to the topology network for recommendation systems. The extended topology not only uses data from the dataset like relationships between users

but also includes user actions to generate more precise and better recommendations. The first step in the approach is clustering of users by dividing the dataset into 4 classes. The first iteration will be to define the users to be recommended. Next are the target followees in the next cluster, then the followers of that cluster and finally the total followees and collected for recommendation.

The works of Abbasi-Moud et al. (2021) bring to attention the use of geographical tags that have aided in providing tourism recommendations. Similarly, the system proposed in our paper uses a tag-value mechanism to provide recommendations. This system has been further described in the next section. Two major drawbacks that come into question when discussing social networks have been discussed in Jung (2011). The authors here raise the issue of data privacy and isolation of social networks. The extraction of data from a network is risky due to an abundance of information that may or may not be sensitive. Furthermore, fetching data from such social networks may not result in complete data but a subset. The proposed work in our paper avoids these issues by using data from a newly developed platform allowing the developers full freedom to choose the subset of data. Additionally, no sensitive data is being used as the only data required for the recommendations is the set of tag values for each user and event.

Sankar et al. (2015) discusses about a glaring issue that exists with recommendation systems based on a form of Social Network. Since these recommendations are drawn for interactions between users and products, it becomes fairly easy to manipulate the data with false tweets and fake information. This is highly penalizable when it comes to stock recommendation as it influences the market. However, our proposed system attempts to combat this by only considering the tag values of each user and event, thereby ignoring any form of textual content that can then influence the recommendation.

Chen et al. (2021) describes a two-stage hotel recommendation system where the authors' first form a set of association rules between hotel features and user preferences. Only then do they create and establish a preference model that can be used to estimate and suggest hotels. Our work implicitly has a similar two-stage approach where the clusters are formed based on associations drawn from users and existing events.

Collaborative filtering (CF), as one of the most promising recommender engines, seeks to investigate a user's preferences on unknown things based on evaluations from comparable neighbors (Wang et al, 2017). Almost all CF engines consider ratings to be indicators for essentially exploring users' tastes and preferences (Shams & Haratizadeh, 2018). Similarity measurements can aid in the search for neighbors and play a crucial part in CF prediction (Pirasteh et al., 2015). When detecting neighbors in a complicated user environment, traditional methods such as cosine similarity, Pearson correlation coefficient, and distance-based similarity are ineffective (Choi & Suh, 2013). As a result, this paper resorts to using Fisher's optimization criterion and purity index for evaluating the quality of the clustering algorithms.

3. PROPOSED EVENTO SYSTEM FRAMEWORK

The following section describes in detail the system design and construction of the proposed work. Just like in every Social Network, the proposed system is divided into two core sections - a frontend and a backend. The frontend has been created solely with the use of ReactJS1 and Bootstrap Material UI2, whereas the backend uses MongoDB3, Express and Node.js4.

3.1 The Social Networking System

The main purpose of a Social Networking System is the proper management of data. To do so, this system stores User and Event Data in a secure NoSQL Database provided by MongoDB Atlas. All sensitive information is encrypted and stored using uniquely generated Salt Strings along with a JWT Secret.

The backend, driven primarily by Node.js, acts as an interconnect between the frontend that is visible to the Users of the System and the database. Each User is authenticated at every step so as

to not permit access to pages where the User does not have the authorization to do so - for example, User A is not allowed to edit details of User B. To do so, a JSON Web Token is generated and stored in the browser's local buffer on sign in. This token is then compared with the data in the backend to authenticate if it is the right user. This approach permits the use of Sessions that allows a user to remain signed in even on closing or in the event of relaunching the browser and/or website.

The loading of any page can be expressed in a few steps as follows - front end triggering a request to the backend, backend sending the request to the database, database fetching the value and returning the data, processing of data by backend, returning of response to frontend, rendering of data response.

Fig 1 provides a high level view of how the system functions. In Fig 1, there are two main actors - User and Admin. The User provides input to the Frontend (made using React). The Frontend in turn processes the data and sends a HTTP request to the Backend in JSON format. The Backend (made using Node.js and Express) receives this request and reroutes the API to the respective methods. The methods in turn process the request and send their own fetch request to the Database. The Database (handled on MongoDB Atlas) uses a NoSQL storage where a cluster is maintained with 2 main collections - Users and Posts. The requested data is fetched and returned as a response to the Backend. The Backend processes the response accordingly and generates its own JSON response which is returned back to the Frontend that receives the response, processes the data and renders the requested information to the User.

The second actor in Fig 1, the Admin, has direct access to the Backend and Database. Admins are able to deal with data without dealing with the abstraction that is provided by the Frontend. In doing so, they can manage the routing and data without having to deal with the hassle of working through an interface. The requests sent are standard HTTP requests with the 4 basic HTTP Methods - Post, Get, Put, Delete. Based on the method and the request, a response is returned and handled by the frontend to render the details properly.

All requests and responses are made with JSON formatted messages. JSON (JavaScript Object Notation) is a lightweight data-exchange format. It is convenient for interpret and construct by eye. It is also simple for machines to parse and generate.5

Fig 2a shows an example of an HTTP request that has been formatted into JSON. This request is sent to the backend and then returns the corresponding response that has been shown in Fig 2b. The response is also formatted in JSON formatting. This formatting has been influenced by the storage format of objects in MongoDB. MongoDB Atlas stores objects in cluster and collections. Each object in a collection is stored in formatting similar to the adopted JSON format. Therefore, by maintaining a JSON message format through the system, communication has been vastly simplified.

Figure 1. Proposed Evento system framework

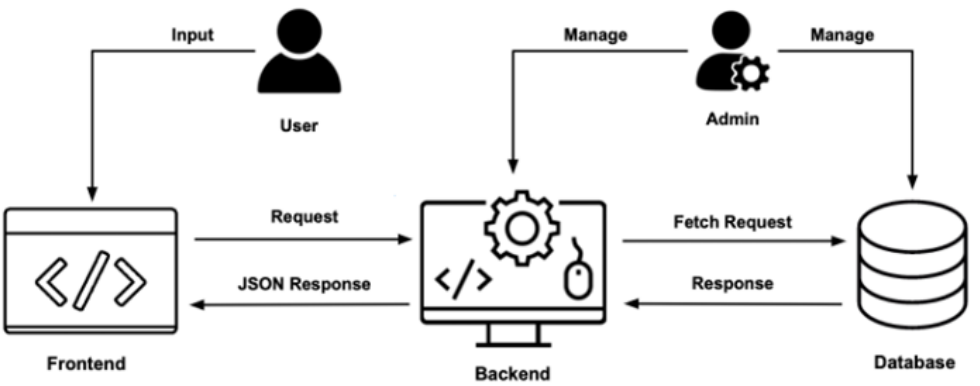


Figure 2. (a) Sample JSON input, (b) corresponding sample JSON output

```
return fetch(`${process.env.REACT_APP_API_URL}/user/${userId}`,
{
    method: "GET",
    headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
        Authorization: `Bearer ${token}`
    }
}
```

(a)


```
{
    "_id": "604f0d5e61a6548490f440d3",
    "name": "Reuel Sam",
    "email": "reuelsam@gmail.com",
    "created": "2021-03-15T07:31:42.866Z",
    "__v": 1,
    "about": "All these squares make a circle... Timeline? Time is not made of lines. That is why clocks are round",
    "updated": "2021-05-30T08:46:02.286Z"
}
```


(b)

3.2 Descriptive Methodology

The methodology for the Social Networking System revolves around effective data management and security, making use of MongoDB Atlas as the NoSQL database and a backend powered primarily by Node.js. The system stores User and Event Data in a secure NoSQL Database provided by MongoDB Atlas. NoSQL databases are chosen for their flexibility in handling unstructured data, which is common in social networking applications.

Following the database, we then have the backend which serves as an intermediary to the frontend that is accessible to users. It plays a pivotal role in processing requests and ensuring data security and integrity. User authentication is a fundamental component of the backends' functionality. Each user is authenticated at every step of their interaction with the system to prevent unauthorized access to restricted pages and data.

For example, User A is not allowed to edit details belonging to User B. To achieve this, a JSON Web Token (JWT) is generated upon user sign-in. This token is securely stored in the user's browser's local buffer. The backend then utilizes this token to verify the user's identity and permissions before granting access to specific system functionalities. The use of JWT tokens facilitates session management, enabling users to remain signed in even after closing or relaunching the browser or website, providing a seamless user experience.

To ensure data security, all sensitive information is encrypted. Unique Salt Strings are generated and used for encryption. Additionally, a JWT (JSON Web Token) Secret is employed for safeguarding sensitive data at rest. This acts as a data pre-processing step to ensure data security throughout the system.

The user interacts with the system through the front end that was built on React.js. The process of loading a page starts with the frontend triggering a request to the backend due to a user action. This request may include user-specific data or information about the requested page. This request is then processed by the backend, ensuring that the user has the necessary authorization to complete this

action. The requested data is then fetched from the database before the data is then processed by the backend. This is then returned to the frontend which display the information to the user accordingly.

This methodology ensures efficient data management, security, user authentication, and data retrieval for the Social Networking System. It provides a seamless and secure user experience while maintaining data integrity and privacy.

3.3 Recommendation Engine

The highlight of this work is the inclusion of User Recommendation based on shared interests that is calculated as the User interacts more with the System. For this recommendation, an Unsupervised Clustering Machine Learning Algorithm – K-Means – has been followed. The User and their tag values are provided as input to the Clustering Algorithm and the output is fetched as the cluster the requesting User belongs in. Before returning the list of suggested Users, every user that is already being followed is removed from the cluster as well.

The K-Means procedure implemented has been detailed in Fig 3. Whenever the user requests to view Recommended Users, the Frontend sends a formal request to the backend along with the userID of the logged in User. The backend then fetches all users in the system, populates their individual tag arrays and then begins to cluster them. The algorithm given in Fig 4 clearly shows the steps involved in the Recommendation Engine. Fig 3 provides a pictorial description for the K-Means algorithm and how it has been integrated into the software while Fig 4 provides an algorithmic overview.

Once the user creates an account, interest tag vector is initialized. The 8 interests are as follows: Environmental, Social, Educational, Sport, Recreational, Music, Political and Festive. The tag arrays are 8-dimensional vectors of float values that vary between the range of 0 and 1. Each time a User expresses interest in an event, the value of the tag for the respective event tags is increased by 5%. Similarly, each time a User expresses disinterest in an event, the value of the tag for the respective event tags is increased by 5%. Whenever a certain tag value is greater than a threshold value (0.5), the user is said to have an interest in that tag. With this method, varying data for the clustering algorithm has been produced.

Going back to Fig 3, once the collection of all Users are populated with their tag values, their names, IDs and tag values are extracted and sent to the K-Means algorithm. The algorithm assigns 4 centroids and iterates over all users, calculating distance and repositioning the centroids until they stop moving. The cluster containing the logged in User is taken and the accounts present in the cluster are filtered to remove the users that are being followed by the logged in User. This filtered collection is then returned and provided as the recommended users to the logged in Users.

As users interact more with the system, they tend to have vastly different tag arrays than when they would have started with. The main idea is to generate clusters for each user using these tag array

Figure 3. User recommendation engine

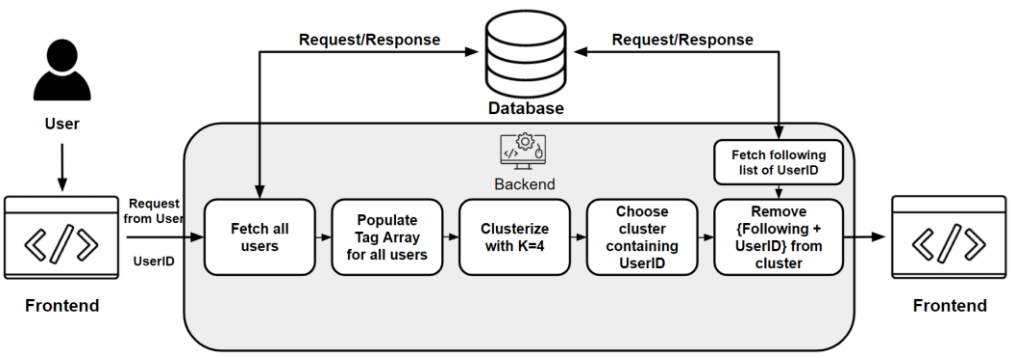


Figure 4. Recommendation engine algorithm

Input: Logged In User ID – uID, List of all Users – allUsers with 8 dimensional tag array of floating point values between 0 and 1, List of all users followed by Logged In User – following

Procedure:

1)→Logged In User's tag value is adjusted. If interest is expressed in a particular event, respective tag values for the user are increased by 5%. On expressing of disinterest in a particular event, the corresponding tag values for the user are decreased by 5%

2)→Populate (8-Dimensional) Tag Array for all users Each Tag Array contains 8 float values ranging between 0 and 1.

3)→allUsers List along with Tag array for each user is given as input to K-Means Clustering algorithm with K = 4

4)→List of Clusters returned by K-Means

5)→Choose Cluster CLS containing uID

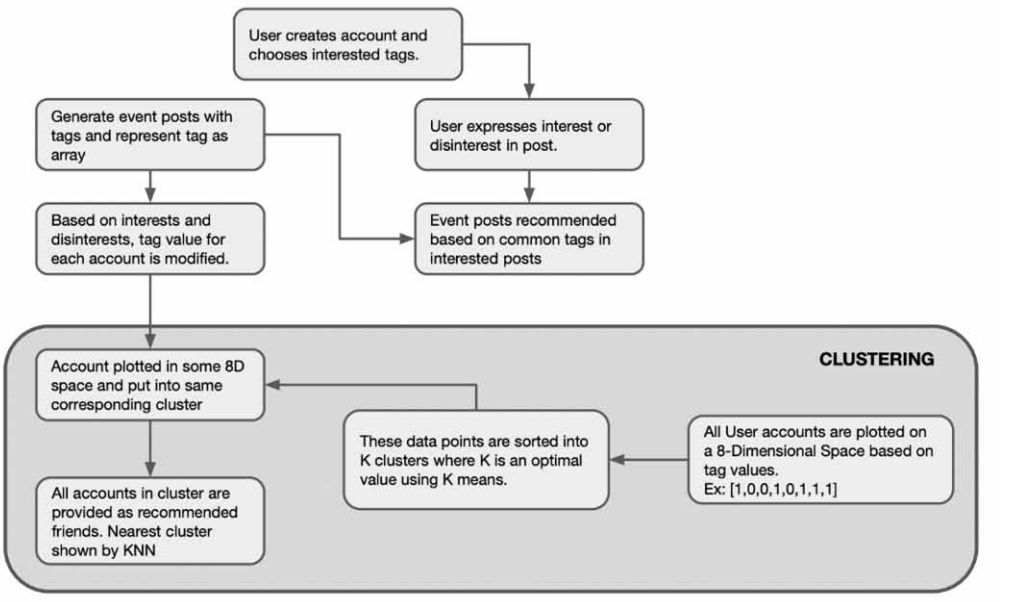
6)→Remove all users in following List from CLS

7)→Remove Logged In User from CLS

8)→Return CLS

Output: CLS – list of recommended Users

Figure 5. Recommendation flowchart



values. Therefore, the users are clustered together based on how similar their tag values. Those users in the same cluster are recommended to each other. A flowchart describing the above discussion regarding the Recommendation Framework and Engine has been provided in Fig 5.

3.4 Limitations and Challenges

One major limitation that can be identified with this approach is the choosing of number of clusters for the recommendation engine. As of now, due to a limited userbase of 50~100 users, we have settled for 4 clusters after conducting a comparative study that can be seen in Section 4.4. As the userbase grows, a re-examination of the k-value will be needed since a k-value of 4 will not scale as the userbase grows. Another limitation that has been identified has to do with the speed of recommendations. These systems tend to work on a small platform with under a few thousand users. But as the number of users grow, there is an inevitable problem of slow recommendations. A future scope could be to introduce a way to avoid re-computation of clusters with each new action that the user makes. Maintaining such a system is also a challenge in the long term since this system is not self-sustaining with regards to funding and revenue. As a result, alternate methods of revenue such as advertisements and subscription models would need to be explored. Furthermore, another concern that has risen in our minds is the need of servers and server space to be handle all the data that flows in as the system grows. A need for distribution computing may arise, requiring the implementation of the system to undergo further change.

Yet another concern that we have identified is the potential privacy concerns that come with collection of user data. The potential for misuse or unauthorized access to any sensitive data that is provided is a considerable worry. However, this system does not collect any sensitive data such as user's address or even the activities. The only form of information that is collected are the interest or disinterest that the user displays to the various events that are posted. This is an unavoidable collection of information since the essence of this work is based on knowing what other users to recommend based on the user's interest.

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

The following section describes the results of the experiment carried out in this work. Details are provided on the implementation and deployment of this work. The results are analysed and inferences are drawn using Fisher Optimization Criterion and Purity Index. The hardware and software specifications regarding the development platform along with the deployment instance has also been expanded upon in this section.

4.1 Hardware Specifications

This section describes the specifications of the software adopted and the hardware the implementation has been carried out on. Various forms of testing have been undertaken inclusive of numerical testing as well as practical testing via Beta Testing.

The entire system has been deployed as an EC2 instance of Amazon Web Services (AWS)6. Amazon Elastic Compute Cloud (Amazon EC2)7 is a cloud-based web service that delivers secure, resizable computational capacity. It is intended to make web-scale cloud computing more accessible to programmers. There are many different tiers and version of EC2 instances available, but keeping in mind the scale of the SNS, the website is running on a t2.micro instance type. The t2.micro instance type features an x86_64 architecture running on single virtual CPU. The memory allocated is 1GB. The storage is dynamic and can be scaled. This instance runs on Ubuntu OS as it is open source and reliable with a lot of support for web applications and processes. Operating system used here is Ubuntu 20.04 LTS8. Table 1 gives details regarding the specifications of the Ubuntu platform that has been instanced for the deployment of this work.

Table 1. Hardware specifications

System	Version
Architecture	x86_64
CPU(s)	1
Memory (GB)	1
Storage	Dynamic
Operating System	Ubuntu

4.2 Software Specifications

For the backend, the system uses Node.js with version 12.22. This version has updates for stable ES Modules and a new API to monitor event loop utilization by threads. For the frontend the, ever important ReactJS with version 17.0.2, has been used. For the elements of the website like cards, lists, dropdown, Bootstrap 5 API is used. The database is handled by MongoDB, a cloud based DBMS. All the data of the users are securely stored in the cloud and not locally. The version of MongoDB employed is 4.4.

4.3 Implementation

This section focuses on the experimental results obtained from the creation and implementation of the proposed system. The system proposed has been implemented along with a recommendation that suggests events to users based on their interests and interaction with the system. This system has been termed as 'Evento' by the authors of this paper. As discussed in prior sections, an entire Social Media System has been created from scratch that implements all the features required of a Social Media platform. All the different pages of the website such as but not limited to the home page, login page, signup page, profile page, event creation page, upcoming events, users page, etc. have been implemented.

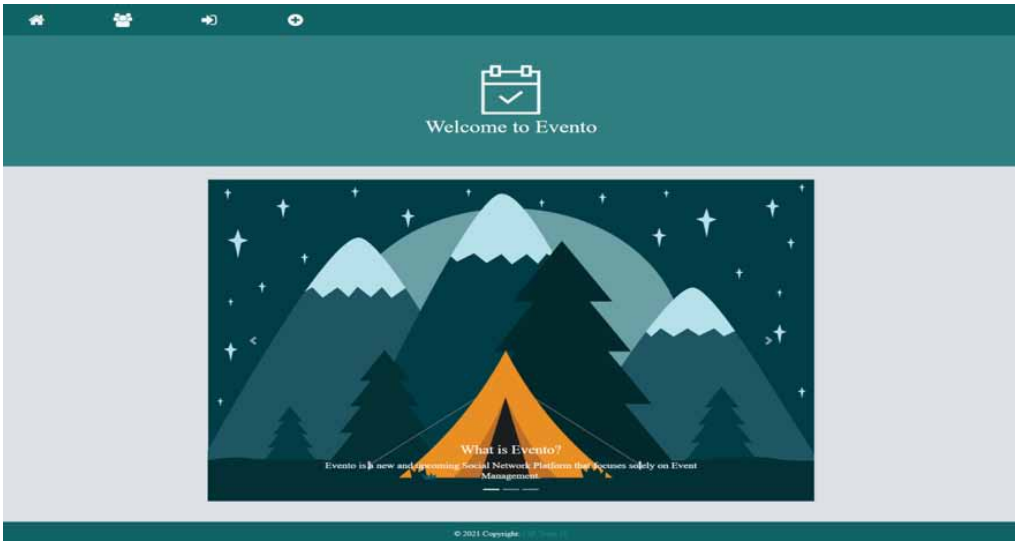
The homepage, as seen in Fig 6, describes the features of the platform while providing information on how one may sign up and get involved with this work. When logged in, the homepage includes some details regarding the events posted by the users already existing on the platform. Events are shown in the orders of recently posted and vice-versa. Popular events from following can also be viewed easily by using the filter option.

Login and signup are fairly simple tasks for the user. The application includes a social login option whereby a user can use one of his/her Google accounts to seamlessly log into the platform. On the profile page, seen in Fig 7, the user can view profile details and info such as followers, following and

Table 2. System specifications

System	Version
Ubuntu	20.04 LTS
Node.js	v12.22.0
ReactJS	17.0.3
Bootstrap	5
MongoDB	v4.4.4
Nginx	1.18.0
PM2	4.5.6

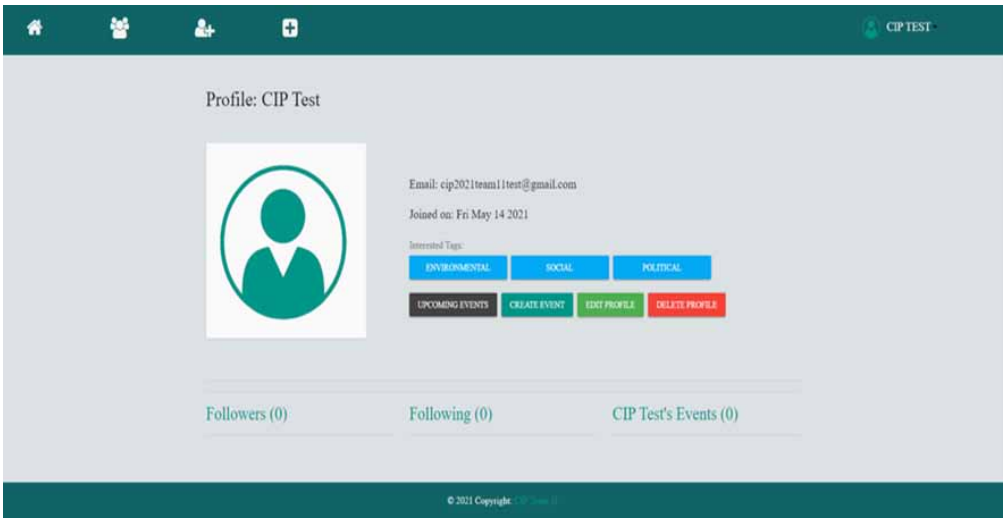
Figure 6. Home page of the application 'Evento'



created events. The user can also edit info about the user like name, email as well changing password. Upcoming events can be viewed easily by the user. Coming to event creation, the user can enter the relevant page and add the required information about the event. The user can add a title, a venue, picture, a description, the relevant tags and date and time. After posting the event, it will be present in the home page of the users and can be viewed. Other users can comment on the event.

The work described in this paper has been successfully deployed as a fully functional web application available on the internet. As described earlier, Amazon's Web Service has been used to achieve this deployment using their EC2 instances.

Figure 7. Sample profile page of the application 'Evento'



4.4 Result Analysis

This section summarizes the inferences drawn from the results. To get a better understanding of the system, the value of 'K' in the K-Means algorithm was varied until an optimal value was found. The value of K is considered to be a performance metric as this is what decides how the users are recommended to each other.

An empirical method that calculates the value of K is given in equation (1).

$$Number\ \ of\ \ clusters = \sqrt{\frac{N}{20}} \text{ where N is a data point} \tag{1}$$

However, since the proposed work has roughly 25 users at this stage, this formula returns an inefficient value, that is 1. It is obvious why having just a single cluster would not be ideal. Therefore it became evident to resort to other measures such as trial and error.

Table 3 describes the different experimental results that were noted with trial and error. On analysing the result, considering the number of users on the platform, K-value of 4 seems optimal for this application.

For numerical testing, two values have been calculated - purity index and Fisher optimization criterion. Purity is an external evaluation criterion of cluster quality. It is defined as the ratio of correctly classified objects to the total number of objects.9 It assesses the homogeneity of clusters in terms of class labels. The formula for Purity index has been provided in equation (2).

$$Purity = \frac{Correctly\,classified\,data\,points}{Total\,number\,of\,data\,points} \tag{2}$$

The main idea of Fisher Optimization Criterion is to find the right direction of classification is to obtain it based on projected data. It is simply a ratio of the between-class variance to the within-class variance, with the goal of maximizing the ratio.10 The idea is to maximize the ratio of the between-cluster variance to the within-cluster variance. The spread of each class is compared relative to the spread within the class. Equation (3) provides the formula applied for calculation of Fisher Optimization Criterion.

Table 3. K-means result analysis

K-Value	Trial No.	Number of Users in Cluster	Number of Users Suggested
2	1	5	5
	2	12	8
3	1	9	5
	2	8	4
4	1	7	5
	2	5	3
5	1	6	4
	2	5	5

$$Fisher\,optimization\,criterion = \frac{\frac{1}{n}\sum_{i=0}^{n}\frac{1}{m}\sum_{j=0}^{m}\left|x_{ij} - c_i\right|}{\frac{1}{n}\sum_{i=0}^{n}\left|\, c_i - \mu \,\right|} \tag{3}$$

where,

n =number of clusters
m = number of data points in each cluster
x = data points
c = cluster centroids
μ = mean of cluster centroids

Since the dataset used is generated by the users by creating accounts and setting up their profiles with tags and with joining and rejecting events, there is no clear cut distinction of correct or incorrect data as with other labelled datasets. So in order to differentiate between correct and incorrect data points (users) within a cluster, a filter is set up such that if within a cluster if any data points (users) have at least 3 common tags with the target user for recommendation is classified as correct else it is classified as incorrect and used in the purity function.

The two metrics that are calculated are commonly used in evaluating clustering algorithms. The Fischer Optimization Criterion aims to find clusters that are well-separated from each other and tightly packed within each cluster. The higher the value, the better the algorithm seems to perform. Purity index has a range of 0 to 1 with a value closer to 1 indicating a better performing algorithm. The value of number of clusters to be formed have been made by examining the values of both these metrics and choosing a k such that the metrics return a higher positive value.

Examining the purity index and Fisher optimization criterion results (see Table 4), the following inferences have been drawn. Initially when K was set to 2, the results highlight the relatively well-separated clusters, based on high Fisher Optimization Criterion of 0.9792 in the first run. However,

Table 4. Purity and Fisher optimization criterion

K-Value	Trial No.	Within Cluster	Between Cluster	Fisher Optimization Criterion	Purity Index	Average Fisher Criterion	Average Purity Index
2	1	3.2741	3.2060	0.9792	0.4615	0.9792	0.5923
	2	3.3234	3.2525	0.9787	0.7272		
	3	3.3262	3.2586	0.9797	0.5882		
3	1	3.2500	3.1565	0.9712	0.6363	0.9714	0.6723
	2	3.3635	3.2678	0.9716	0.6666		
	3	3.3266	3.2315	0.9714	0.7142		
4	1	3.3852	3.2843	0.9702	0.8	0.9694	0.8083
	2	3.3705	3.2698	0.9702	1.00		
	3	3.3819	3.2730	0.9678	0.625		
5	1	3.3421	3.2315	0.9669	0.5	0.9661	0.3888
	2	3.3263	3.2195	0.9679	0		
	3	3.2863	3.1764	0.9636	0.6666		

the Purity Index varied by indicating the mixed cluster contents. As K increased to 3, the Fisher Optimization Criterion slightly reduced, suggesting less distinct clusters, but the Purity Index consistently improved, reflecting better-defined clusters. For K=4, although the Fisher Criterion decreased further, the Purity Index was notably high, emphasizing the balance between cluster separation and compactness. On the other hand, K=5 exhibited decreasing cluster separations and mixed cluster contents, reflecting in the metrics. It can be noticed that, based on the analysis, K=4 stands out with the highest average Purity Index of the 0.8083, indicating well-defined and pure clusters, making it an attractive choice when considering the balance between cluster separation and compactness. In contrast, K=2 achieves the highest average Fisher Criterion of 0.9792, emphasizing well-separated clusters, albeit with a relatively lower average Purity Index of 0.5923. These results reinforce the notion that the choice of K is a critical decision in K-means clustering, and its impact should align with the specific objectives and requirements of the clustering task.

5. CONCLUSION

The event-based social networking system was created to facilitate social networking revolving around events. Due to the lack of a Social Media platform that is exclusively meant for Event Management, the need for such a platform came to surface. The implemented system allows for interaction between Users and Events. The product will allow the user to view profiles, view Events, express interest, express disinterest, create Events, delete Events, comment on Events, interact with other Users and other trivial functionalities such as login and sign-up.

To stand out in the market, a Recommendation Engine was put into place that provides Users with suggestions of other profiles to interact with based on their interests. The recommendation engine takes in an 8-dimensional vector array as input and returns the recommended users as output. The engine proposed in this work achieved a purity index value of 0.8083. Furthermore, the recommendation engine can be expanded to not only recommend users but also to recommend other important features, like events, groups, chat rooms, pages, etc.

An event-based social network equipped with a K-means-based recommendation engine would make substantial contributions to both theory and practice within the realm of the social networking. Theoretically, grouping people with comparable preferences and the interests using clustering algorithms would increase our understanding of user behaviour. This would make it possible to create more detailed, data-driven user segmentation models, enabling academics to learn more about the dynamics of social interactions. Additionally, incorporating K-means recommendations into the network would make it easier to explore personalized content delivery and the dynamics of social influence in greater detail, supporting the creation of cutting-edge algorithms and insights in the process.

On the practical side, this combination of a K-means recommendation engine would enable users to find other users and possibly events that closely match their interests and preferences, boosting their experience as a whole. It would encourage more meaningful and relevant interactions by providing customized event suggestions based on shared interests within user clusters, ultimately resulting in a more engaging and socially linked digital ecosystem.

The work can be improved upon further in many ways. In an attempt to emulate popular social networks, the application can include many popular features. A messaging system could be added for users to communicate with each other with the provision of groups to facilitate the communication of multiple users in a single space. Instead of just people being able to create accounts, organizations, companies and other entities can also be given the provision to do so. This would keep the platform more engaging and help with the event management for large scale events.

REFERENCES

Abbasi-Moud, Z., Vahdat-Nejad, H., & Sadri, J. (2021). Tourism recommendation system based on semantic clustering and sentiment analysis. *Expert Systems with Applications*, *167*, 114324. doi:10.1016/j.eswa.2020.114324

Anandhan, A., Shuib, L., Ismail, M. A., & Mujtaba, G. (2018). Social media recommender systems: Review and open research issues. *IEEE Access : Practical Innovations, Open Solutions*, *6*, 15608–15628. doi:10.1109/ACCESS.2018.2810062

Belkhadir, I., Omar, E. D., & Boumhidi, J. (2019). An intelligent recommender system using social trust path for recommendations in web-based social networks. *Procedia Computer Science*, *148*, 181–190. doi:10.1016/j.procs.2019.01.035

Ben-Shimon, D., Tsikinovsky, A., Rokach, L., Meisles, A., Shani, G., & Naamani, L. (2007). Recommender system from personal social networks. In *Advances in Intelligent Web Mastering: Proceedings of the 5th Atlantic Web Intelligence Conference–AWIC'2007, Fontainbleau, France, June 25–27, 2007* (pp. 47-55). Springer Berlin Heidelberg.

Cai, C., & Xu, H. (2019). A topic sentiment based method for friend recommendation in online social networks via matrix factorization. *Journal of Visual Communication and Image Representation*, *65*, 102657. doi:10.1016/j.jvcir.2019.102657

Chen, K., Wang, P., & Zhang, H. Y. (2021). A novel hotel recommendation method based on personalized preferences and implicit relationships. *International Journal of Hospitality Management*, *92*, 102710. doi:10.1016/j.ijhm.2020.102710

Choi, K., & Suh, Y. (2013). A new similarity function for selecting neighbors for each target item in collaborative filtering. *Knowledge-Based Systems*, *37*, 146–153. doi:10.1016/j.knosys.2012.07.019

Golbeck, J. (2009). Trust and nuanced profile similarity in online social networks. *ACM Transactions on the Web*, *3*(4), 1–33. doi:10.1145/1594173.1594174

Jung, J. J. (2011). Ubiquitous conference management system for mobile recommendation services based on mobilizing social networks: A case study of u-conference. *Expert Systems with Applications*, *38*(10), 12786–12790. doi:10.1016/j.eswa.2011.04.070

Li, Y., Liu, J., & Ren, J. (2019). Social recommendation model based on user interaction in complex social networks. *PLoS One*, *14*(7), e0218957. doi:10.1371/journal.pone.0218957 PMID:31291288

Pirasteh, P., Hwang, D., & Jung, J. E. (2015). Weighted similarity schemes for high scalability in user-based collaborative filtering. *Mobile Networks and Applications*, *20*(4), 497–507. doi:10.1007/s11036-014-0544-5

Sankar, C. P., Vidyaraj, R., & Kumar, K. S. (2015). Trust based stock recommendation system–a social network analysis approach. *Procedia Computer Science*, *46*, 299–305. doi:10.1016/j.procs.2015.02.024

Shams, B., & Haratizadeh, S. (2018). Reliable graph-based collaborative ranking. *Information Sciences*, *432*, 116–132. doi:10.1016/j.ins.2017.11.060

Silva, N. B., Tsang, R., Cavalcanti, G. D., & Tsang, J. (2010, July). A graph-based friend recommendation system using genetic algorithm. In *IEEE congress on evolutionary computation* (pp. 1–7). IEEE. doi:10.1109/CEC.2010.5586144

Wang, Y., Deng, J., Gao, J., & Zhang, P. (2017). A hybrid user similarity model for collaborative filtering. *Information Sciences*, *418*, 102–118. doi:10.1016/j.ins.2017.08.008

Xiong, X., Qiao, S., Han, N., Xiong, F., Bu, Z., Li, R. H., Yue, K., & Yuan, G. (2020). Where to go: An effective point-of-interest recommendation framework for heterogeneous social networks. *Neurocomputing*, *373*, 56–69. doi:10.1016/j.neucom.2019.09.060

Yigit, M., Bilgin, B. E., & Karahoca, A. (2015). Extended topology based recommendation system for unidirectional social networks. *Expert Systems with Applications*, *42*(7), 3653–3661. doi:10.1016/j.eswa.2014.12.043

Zhao, W., Ma, H., Li, Z., Ao, X., & Li, N. (2020). Improving social and behavior recommendations via network embedding. *Information Sciences*, *516*, 125–141. doi:10.1016/j.ins.2019.12.038

Zheng, Y., Zhang, L., Xie, X., & Ma, W. Y. (2009, April). Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th international conference on World wide web* (pp. 791-800). doi:10.1145/1526709.1526816

ENDNOTES

1 https://reactjs.org/
2 https://mdbootstrap.com/
3 https://www.mongodb.com/
4 https://Node.js.org/
5 https://www.json.org/json-en.html
6 https://aws.amazon.com/
7 https://aws.amazon.com/ec2/
8 https://ubuntu.com/
9 https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html
10 https://online.stat.psu.edu/stat508/lesson/fishersoptimizationcriterion