**AI Event Manager : An event recommendation engine**

# Introduction

The project solves the issues associated with manual process of managing events in global setting by automating the process of determining consumer preferences and based on that recommending suitable events. The number of events that can be recommended to a consumer is configurable and can be changed based on the types and volume of events inducted into the system.

The overall approach adopted in the solution starts with creating the word vector embeddings of consumer preferences and events. The pairwise similarity score is calculated between the word embeddings of word vectors and that of events. Final recommendations are created by rank ordering the events based on the similarity scores. The process is illustrated as below:

A diagram of a computer

Description automatically generated

**Solution Architecture**

The similarity score matrix is flattened by operation depicted as below:

A screenshot of a computer

Description automatically generated

The rank ordering is applied on the flattened similarity data and “*TOP\_K*” (set to 3 by default) unique events are taken as event recommendations for the user to which the interests *<Interest 1, Interest 2, Interest 3, …, Interest N>* belongs.

# Data Handling

There are following types of data used in the proposed solution architecture:

* **Embedding Data :** The word vector is embedding is stored in a custom coded vector database which is optimized to store and similarity search operations. The underlying data is stored in in-memory SQLITE3 database. Following is the data dictionary of the same:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| Index | String | The unique identifier of the word indexed in the vector DB |
| Value | String | The string value of the word for which embedding is generated |
| Embedding | Array(Float) | The word embedding array of the word. |

* **Users Login Data :** The login credentials along with the user metadata is stored in this table and is referenced across the code base to render user and privileges specific data. Following is data dictionary of the table:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| username | String | The name of the user |
| password | String | The password of the user |
| type | String | The 'admin' will be present if the user has admin privileges otherwise 'user' will be present |
| profilePic | String | The path to the profile picture |
| email | String | The email of the user |
| userID | Integer | The unique identifier of the user. It is autoincremented as a new user is signed up |
| isActive | Integer | The flag representing whether a user is active and can login or not. |

* **Interests Mapping :** This is a utility table that maps the users with their specific preferences mentioned via interest tags. Following is the data dictionary of the table:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| userID | Integer | The unique identifier of the user. It is foreign key reference to the userID column of users table. |
| WordIndex | String | The unique identifier of word in vector DB which is associated as interest for the user. |

* **Events Data :** The data associated with events is present in this table. Following is the data dictionary for the same:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| eventID | String | The unique identifier of the event. |
| WordIndex | String | The unique identifier of word in vector DB which is associated as event tag of the event. |
| eventName | String | The name of the event |
| eventDescription | String | The description of the event |
| eventDate | Timestamp | The date at which event will happen. |
| eventAddress | String | The address of the venue for event |
| eventPic | String | The path of image for the event |

* **Event Recommendation Data :** The mapping of events recommendation with the specific user IDs. Following is the data dictionary:

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| eventID | String | The unique identifier of the event. Foreign key reference to eventID column of events table. |
| userID | Integer | The unique identifier of the user. Foreign key reference to userID column of users table. |

# API end point references

## Home Page (/)

This is the landing page of the web application. It contains basic description of the solution.

## View Recommendations (/recommendations)

This is the web page that renders the recommended events for the user. It internally invokes the “/viewRecommendedEvents” endpoint and displays the result in user readable format.

## Login (/login)

This endpoint renders the login form for the user to enter their credentials.

## Signup (/register)

This endpoint renders the signup page for the user to setup their login and initial interest preferences.

## Logout (/logout)

This endpoint is responsible for logging out the user by removing the session variables of the user.

## Admin Panel (/admin)

This endpoint renders the admin panel only for the users who has the admin privileges granted. With this panel admin can view user’s data, activate / deactivate users and view the recommended events to them.

## Register Backend (/registerBack)

The endpoint that handles the backend processing of signup activity.

## Login Backend (/loginBack)

The endpoint that handles the backend processing of login activity.

## Render Users (/users)

The endpoint is accessible to admin users only and is responsible for fetching the data of users in the system.

## Activate (/activate/<user\_id>)

The endpoint activates the user whose user Id is passed as parameter. It is only accessible for admin users.

## deactivate (/deactivate/<user\_id>)

The endpoint deactivates the user whose user Id is passed as parameter. It is only accessible for admin users. After deactivation users cannot login until admin activates them explicitly.

## Settings (/settings)

The endpoint renders the settings panel which provides option to change credentials (username, password and email) and profile picture of the users.

## Analytics Dashboard (/analytics)

The endpoint that renders the dashboard to view interests, add interests, add events (Only for Admin), delete interests and view all events (Only for admin).

## Change Credentials (/changeCredentials)

The endpoint invokes the backend code to change the credentials of the users.

## Change Profile Picture (/changeProfilePicture)

The endpoint invokes the backend code to change the profile picture of the users.

## Add Interest (/addInterest)

The endpoint is responsible for computing the embedding of the interests and storing the same into the database.

## Add Event (/addEvent)

The endpoint is responsible for computing the word vector embeddings of event tags, adding the details into the database and generating the initial recommendations for the users.

## Delete Interest (/deleteInterest)

The endpoint is responsible for deleting the specific interest word from the mapping for the user.

## View Events (/viewEvents)

The endpoint responds with the data of all the events. This is accessible for only admin users.

## View Recommended Events (/viewRecommendedEvents)

The endpoint generates the recommendations for the user and responds with the “*TOP\_K*” event recommendations.

## View Interest (/viewInterest)

The endpoint is responsible for fetching all the interest tags of the user and respond with the list of the same.

## Delete Event (/delete\_event/<event\_id>)

The endpoint is responsible for removing the event with event id supplied as parameter from the database.