**CHAPTER 10**

**INTEGRATION**

**10.1 Integration strategies**

The development part of this project follows Bottom up approach. The bottom up approach is used for building small components, and finally to integrate them into one complete system. Initially the backend implementation was done. Each small component was implemented in python. API's were defined for better interaction and better integration with the frontend as well as for interaction between the backend modules.

To integrate all the components, top down integration approach is used. Top down approach is better for integration, and for every scenario, the corresponding backend API to be used, is also known. The User interface functionalities are linked with backend one after the other one, in a sequential manner. For the frontend, PyQT has been used and the backend coding has been done using Python. As python was used, integration was a challenging task. No python frameworks such as Django were used.

**10.2 User Data Generation**

Using Graph API, the Facebook Access token is obtained. Using this access token, the user profile information and his connected users’ information is fetched in a JSON format. From this JSON data, the required field values is queried and stored in database as User Profile(UP) format.

Get Facebook Access token

Fetch the User’s Information from facebook

Users data in UP format

***Fig. 10.1*** *User data generation*

**10.3 Pools Creation**

From the user profile stored in the database, each user’s information is fetched. This information is parsed and user attributes are retrieved for the different pools. The pools are created by the attribute names of the user fetched from the database. These pools are filled with attribute values. This is done for each of the users’ information and the various pools are generated.

User profile

Get Users Attributes

Parse the Attributes

pool

***Fig 10.2*** *Pool creation*

**10.4 Description Builder**

The pools information present in database is fetched. Each pool is parsed and the attribute values are taken, the description for the attribute value is fetched from the Wikipedia, WolframAlpha and Wordnet.

Get Attributes from Pools

Description for the Attributes

pool

Description builder

***Fig.10.3*** *Description builder*

**10.5 Similarity calculation**

Similarities are calculated between the context entered by the user and the contexts in the database. The user enters the description of the context and the attributes for that context. This context will be matched with the existing contexts in the context database, and the best matching contexts are recommended to the user.

**10.6 Add Context**

In this User Interface component, the Context description is taken as input from the user. First, contexts are recommended from the existing contexts. If the user does not select the context from it, using the Reverse Dictionary the possible context names are suggested. User can select the suggested context name or user can enter the new context name for the description, and the context name and context description are stored in the database. After this, the user is asked to enter the Property description, similar as Context description the property name is suggested to user. Here too, the user can select the suggested property name or he can enter new property name and update the same in the database for that context.

Context database

Get Context Description

Suggest Names

Select the Context Name for the Description

Ask for Property Name and Description

***Fig. 10.4*** *Add Context*

**10.7 Browse Context**

In this User Interface component, the context names present in database are fetched from the database and displayed. User can click any of the context name displayed, to see the context description. The required context name from the context list is to be entered to choose the context.

Suggest Context Names

Select the Context

Context

Database

***Fig.10.5*** *Browse context*

**10.8 Generate Network**

Context description, properties and property description for the corresponding context are fetched. The fetched property description is compared with the pools description. The pool description which matches the properties is fetched from database. These selected pools property values are compared with the user profile. The user profiles which match the property values are selected. The selected user IDs and their names are used to generate the graph. Nodes in the network graph generated represent users congruent to the context. Graph generated contains users who match to the context, containing connections between them. User can click any of the nodes present in graph. The user information of the clicked node is fetched from the database and displayed.

Match (Property Value with User Property Value)

Context

database

Fetch the Properties for the given Context

Match (Property Value, User Property Value)

Users’ id to Graph UI

User Profile

***Fig. 10.6*** *Generate Network*

User Data Collection

User Profile

Pools Creation

Pools

Description Builder

Add Context

Context database

Browse Context

Generate Graph

***Fig. 10.7*** *Integrated System*

The user data collected from Facebook is inserted into MongoDB in required format. From this data, pools are generated. Description of every attribute value in all the pools is generated from Wikipedia, Wordnet, and WolframAlpha and stored. For every pool, similarity between attribute values is calculated and stored.

When a new context is added by a user, the context database is updated accordingly. When a user wants to use an existing context, he can browse and select the context. Once the user has entered a context, his Egocentric congruent network is generated by considering the user’s profile information and the context in which he is in. This network is generated by matching the user’s profile with other users who are also in that context. The Integrated system which generates the network is as shown in the diagram above.