

# **1. INTRODUCTION**

The internet is the largest source of information. Of the internet traffic, according to statista.com<sup>[1]</sup>, 1.79 billion people are users of social networks. The ubiquitous nature of the internet allows for people from all backgrounds to express themselves on social media. This has led to an explosion of data, especially data about and by people.

In this chapter, we discuss how social networks present to us abundant data which, if processed, could provide us a wealth of information. We discuss the different kinds of data we collected, the challenges we faced and the applications of our project.

## **1.1 Social Media Data and Analysis**

Social media presents a diverse range of platforms for personal expression. This makes it a rich source of data about a person. However, the most challenging hurdle in social data analysis is finding relevant information about a person. Given the varied purposes for which each network is to be used, we not only get a fairly holistic view of a person, we also face the challenge of sorting through entirely different types of data. For example, Facebook is a popular social network that is largely used for personal data such as photos, statuses depicting thoughts of users, interests and events that users attend etc. In contrast, LinkedIn is a professional network, restricted to listing jobs, positions and projects that an individual was involved in professionally over the years. Tumblr and Pinterest are blog-based social networks, whereas GitHub is more of a version control system with social aspects. These are merely a few examples of the various networks available today.

With these differences between networks in mind and the sheer volume of data available through them, it is not much of a surprise that despite the potential for

## **Social DaTa Aggregator and Locator of Knowledge**

using social and professional profiles to shortlist individuals for competitions, as a means of background screening, market research etc., people are apprehensive of the daunting task of combing through several irrelevant posts in order to find nuggets of genuine information about a person. Some examples of data that can be obtained from social networks is the number of photos a person posts, the number of statuses, the number of friends he has, how many people he interacts with most – all of which indicate his sociability and people skills. We can also obtain his work and education history, project details, papers, his work preferences – all of which give an insight into his academic and professional capabilities. A lot of information can be derived by analysis of this data.

### **1.2 Collection and storage of Data**

For any project that seeks to extract information, the first pre-requisite is data. Collection of data is often a time-consuming task that requires careful design in order to be easily usable.

For the purpose of this project, we collected data from Facebook, LinkedIn and GitHub. This can be extended to other social media websites and this data set was restricted for purpose of establishing the proof of concept. We collected a subset of these links from various people in order to develop and test our product. Social media data is exposed through REST APIs by the websites. Most social networks also add privacy constraints and require an authentication scheme to gain access to the data.

The collected data needs to be stored in a database to be used for analysis at a later time. The information extracted is also stored for easy access.

### **1.3 Challenges and opportunities**

## **Social DaTa Aggregator and Locator of Knowledge**

Social media has proven to be a large and diverse source of information .It is not just a set of facts; it is a representation of a person as a whole. However, the procuring of data as well as the privacy policies of these media make the processing a challenge. Hence in order to obtain relevant personal information, the user has to agree to provide the data and hence grant permission to access and retrieve his/her personal information from his/her social media profile.

Another challenge faced was the storage and retrieval of data as well as representation in a user-friendly format. The data obtained from social media is highly unstructured and voluminous in nature. Meaningful information has to be combed out of this vast source of data. Similarly the large amount of data has to be stored in an efficient way to allow faster access and retrieval of information.

The third challenge faced was the implementation of a query system. There were several questions to consider. How feasible is it to implement a query processing system from scratch? Are there APIs that can be used? How accurate are we aiming the system to be? What sort of data will be present and will it pose any limitations to the querying system? All of these questions needed to be discussed and debated before the query module was implemented.

Since the varied and diverse nature of this data can be used to build a complete profile of a person, there is no dearth of opportunities. Any use case that requires basic or detailed information about a person without subjecting him to endless surveys which later require manual analysis to determine the complete profile of the person, thereby taking up time on the part of both actors, can make use of our project with a few changes.

## **Social DaTa Aggregator and Locator of Knowledge**

### **1.4 Application**

Our product has a wide range of applications. Any application involving at least two users - a person whose profile is to be evaluated and the person who will evaluate - can make use of this product with a few changes. A few use cases are discussed below in detail.

#### **1.4.1 Use Case 1: Job Screening**

One of the most important applications of using social media data is to determine the basic traits and qualifications of a person that would make him or her suitable to be employed by an organisation. With the increasing number of equally talented applicants and selected candidates, it requires a massive amount of time and effort for interviewers or HR departments to sift through the profiles of all the applicants. Using a survey to determine whether a person is suitable for a particular job or not may result in the applicants submitting false answers. While the veracity of social media can also not be verified, it can at least provide multiple sources of information with to refer to.

According to a survey conducted by CareerBuilder <sup>[2]</sup>, a job portal used across 55 countries, 43% employees use social media in screening their employees. With the vast data present in social media, the consolidation of this data as a single profile of a person will be extremely helpful in giving companies the information required by them and will considerably reduce the time that would have otherwise been spent in looking at each profile individually.

## **Social DaTa Aggregator and Locator of Knowledge**

### **1.4.2 Use Case 2: Hackathon Shortlisting**

As with job screening, hackathons and other competitions also see a large number of applicants and often have limited places to offer. The Aadhaar Hackathon conducted in January 2015 had over 1800 applicants out of which only 140 were selected.<sup>[3]</sup> In such a case, it would be best to determine which applicants would get the most out of such an event and who could give back to the event as well.

Many hackathons today require a candidate to submit their GIT/Linkedin and other profile details, so as to shortlist the candidates from a vast pool of applicants. The social media data can be consolidated to provide judges with information required by them to make a correct selection.

### **1.4.3 Use Case 3: Market Analysis**

It is important to know how a product will be accepted widely in the market. For this, our application can be applied across different profiles of people to assess what characteristics people who use similar products or react favourably to a product have in order to find out who would be the best group of people to market the product to.

Some other use cases are activity monitoring or support group analysis. These are just a few mentioned use cases and these can be extended and adapted to the different needs.

## **2. PROBLEM DEFINITION**

The main aim of this project is to consolidate and analyse data from different social media and present the consolidated information to the user on a single platform based on the requirements of the use case. In addition to this, we allow a user to quickly query information they need. This is an emulation of the graph search feature limited to only facebook. Our query engine works across all the social media tested in this implementation and can be extended to any other social media network as well.

### **2.1 Collection of data from different API**

The three sources of social media that were used are GIT, LinkedIn and Facebook. Data is exposed through APIs for each website. Each API has its own process of invocation and authentication and the format of the data returned is also different. This posed a challenge as we needed to understand the format of each network's data, decide upon and create a uniform template for use in our product and finally adapt the raw data returned from the API to fit the template.

### **2.2 Processing of Data collected**

The above data is in a raw format and has to be converted to information. For this purpose we run a controller that converts the data into information and presents it in human friendly natural language. The controller processes data for different social media sequentially for a person. However, it processes data of upto 5 people in parallel. This can be scaled when the load on the product increases. This information was then consolidated and presented as a generic profile of the person.

## **2.3 Analysis of Information**

We analysed the information that was consolidated above and performed sentiment analysis, text categorization, taxonomy recognition, concept recognition and entity recognition. This helped us provide a deeper profile of the person in addition to information he was directly providing.

## **2.4 Query**

Even though all the information described above is presented to the user , we also provide an additional feature to query the information directly. This information can query through the raw data as well as analysed information and provides an easy interface to quickly get information.

## **2.5 Website**

The interface for our product is a website. There were different views for our two actors. The subject could submit links of his choice for different events. The user can create and manage events, create and manage templates, view and query information of subjects and also accept and reject them through this interface.

## **3. LITERATURE SURVEY**

This chapter has the various papers and books we referred to that speak about mining social media and processing/analysing the data.

### **3.1 Collection of data from different API**

We have collected data from three social media - GIT, Facebook and LinkedIn. We used the following resources to help us understand the process of collection and to achieve this.

#### **3.1.1 Mining the Social Web: Matthew A Russel [4]**

As the name of the book suggests, this book provides various pointers and step-by-step instructions on mining social media and analysing the information obtained from the user's profile. It provides detailed explanation about the various APIs present to obtain data from the user's profile. Among the numerous social networking sites available, the book focuses on extracting and analysing data from LinkedIn, GitHub, Facebook, Twitter and Google+. Each chapter introduces a social website, teaches us how to use its API to fetch data, and introduces some techniques for data analysis. This book directs us in terms of the data to be collected.

We used some of the techniques specified here to collect data from the three social media we used.

#### **3.1.2 Mining Knowledge from Text Using Information Extraction: Raymond J. Mooney and Razvan Bunescu [5]**



## **Social DaTa Aggregator and Locator of Knowledge**

Most data-mining research assumes that the information to be “mined” is already in the form of a relational database. Unfortunately, for many applications, especially those that use data from social media, information is in the form of unstructured natural-language documents rather than structured databases. Consequently, the problem of text mining, i.e. discovering useful knowledge from unstructured text, is becoming an increasingly important aspect of KDD. The simplest method to mine text would be to adopt a bag of words method. But this does not take the context into consideration. Other methods in information extraction (IE) involve recognizing several types of entities in text and identifying some relationships that are asserted between them. This paper describes methods ranging from manually encoding patterns to identify entities to supervised learning methods. It was particularly useful in narrowing down methods that we can use in our project for extraction relevant information from social media data.

### **3.2 Processing Data**

Data got from these different sources needs to be processed to make sense of. To do so, we needed to decide how and which data can be interpreted relevantly. We studied different papers to understand what constitutes useful information in social media before making this decision.

#### **3.2.1 Finding High-Quality Content in Social Media**

**Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, Gilad Mishne <sup>[6]</sup>**

Social media data is often full of humour and unnecessary information. Sarcasm, false information and exaggeration is rampant among the posts. In such a

## **Social DaTa Aggregator and Locator of Knowledge**

scenario, the issue of finding high quality information that actually presents a true picture about a person is an important one. This paper focuses on using user feedback on Yahoo answers to judge the quality of content. This is partially relevant to our system as we can use a similar approach judge from comments on posts, the importance or truth of the post. It uses various factors like user relationships, features in content and comments to judge the quality of the answer. It then presents a mathematical way of evaluating quality. This paper helped us to identify the prospective problem of finding quality in the data collected but didn't offer much help since it seemed to be of little relevance to our problem statement.

### **3.3 Analysing the Data**

The information on social media can indirectly provide information that may be useful to a profiler. Following are some ways to perform the analysis.

#### **3.3.1 Using Social Data as Context for Making Recommendations: An Ontology based Approach:Salma Noor and Kirk Martinez [7]**

Social networking websites are becoming increasingly popular day by day. Facebook has approximately 1.2 billion active accounts. As every minute passes, 293,000 status updates and posts are made. Thus, the social web could provide a better understanding of a user's interests and intentions. But this presents us with two main problems - recommending items of interest to new users and the data overload problem in which highly relevant information is lost due to a lack of filtering. This paper focuses on tackling the data overload problem by filtering information based on context. After extracting the data obtained from the user's profile, the

## **Social DaTa Aggregator and Locator of Knowledge**

authors conduct data filtering by checking it against various ontologies to bridge the vocabulary gap and in finding synonyms of these tags. In the next phase, semantic information acquired about the tags from the web is used to form a common vocabulary which is then mapped to the categories of existing ontologies. These can then be used to send the appropriate recommendation to the user based on his/her areas of interests. This paper was relevant to our project in learning how to extract appropriate information from the vast amount of social data and classify it using an ontology.

### **3.3.2 Studying Facebook via Data Extraction: The Netvizz Application Bernhard Rieder <sup>[8]</sup>**

Facebook is one of the largest and most widely used social networks. This paper describes using the Facebook API to extract information and presents its analysis of the same. It describes the Netvizz application that is used for data extraction (along with the issue of permissions being required to guard user privacy). The app examines personal networks, pages and groups. However, it was more focused on representing and examining the social network as a graph aspect of the data rather than the textual content. This paper was useful in examining the privacy and ethical side to extracting social data from Facebook and in visualizing networks but did not offer much in terms of text processing.

### **3.3.3 Inferring Private Information Using Social Network Data Raymond Heatherly <sup>[9]</sup>**

## **Social DaTa Aggregator and Locator of Knowledge**

A lot of data/information published by individuals is private to external world. However, this information may be crucial for some end users. This paper predicts ways to predict personal information using some learning techniques. For example, social network data could be used for marketing products to the right customers. This paper focuses on the problem of individual private information leakage due to being part of an online social network. Social network data could be used to predict some individual private trait that a user is not willing to disclose (e.g., political or religious affiliation) and explore the effect of possible data sanitization alternatives on preventing such private information leakage.

We are accessing private information of a person and need to be cautious. We use some inferring techniques to find out derived information about a person.

## **4. SOFTWARE REQUIREMENT SPECIFICATION**

### **4.1 Scope**

We define two entities in this system: The “user” is the person querying information. The “subject” is the person who has submitted links to profiles that we extract information from. We distinguish between “data” which is all the data that can be got from a person’s profile and “information” which is the relevant knowledge extracted from the data.

Functionalities within our scope:

- Our project ensures that user querying information about the subject has permission from the subject to access this information.
- Our system will have pre-defined classes that represent commonly occurring traits that most users look for e.g. social skills, technical interests, technical skills, education etc.
- The user can query for information which may not be already classified in the system or wants to know at a quick glance
- The system uses ontology to disambiguate queries and information. Thus, the system is self-sufficient and can answer queries in a generic manner.
- Queries about facts or easily derived information are valid. The system may ask for further clarification for some queries.
- Facilitate saving of templates by the user for future use
- Facilitate handling multiple events by the same user

## **Social DaTa Aggregator and Locator of Knowledge**

- Ensuring that information from only the social networks that the event requires is shown to the user despite other information also being present for a subject.
- Allowing the subject to change his social media profile links if necessary without having to log in to the system.

Important points out of scope:

- The system does not provide any sort of rating or opinion. i.e. it will not be able to answer questions like "is this person suited for a software development position" that require an opinion. The user may ask questions like "does this person know Java" etc. that will allow him to form an opinion.
- The system does not guarantee that the data available to the user is accurate. It can only parse the information. It cannot provide a validation between networks.
- The system cannot provide the latest information from the data. Information is refreshed only once a month from entry of the profile link.
- The system does not provide a personality profile such as classifying a person as introvert, outgoing etc.

## **4. 2. General Description**

### **4.2.1 Product perspective**

This project is an information extraction system that provides a consolidated view of a person's social media presence. It takes data from several social media profiles of a person and categorises relevant information extracted from this data as required by the user in a structured form. It is a means by which a user can view

---

## **Social DaTa Aggregator and Locator of Knowledge**

details such as education, technical skills, social skills, popularity etc. of a person in a single generated profile instead of having to read through numerous profiles. The system also provides a search facility by which the user can directly query information about a person.

As this is a machine learning software, the machine may not at all times understand the question being asked by the user. In order to improve the experience for the user and improve our software's accuracy, we provide the user with an interface to help us create new classes or improve features of existing classes.

### **4.2.2 User Characteristics**

#### **User:**

The user views profiles of subjects. He can log in, manage events, view profiles and take action (accept/reject). He can also save templates and make queries about the profile. His interaction is limited to the functionalities provided on logging in.

#### **Subject:**

The subject is the person of interest whose profile is being reviewed by the user. The subject's interaction with the system is restricted to willingly providing links to their social media profiles. The information from these profiles will be used to answer the questions posed by the user. The subject's data is taken as input and the information is used as output.

## **Social DaTa Aggregator and Locator of Knowledge**

### **4.2.3 General Constraints**

1. Permissions to access profile
2. Security of the data obtained from the subject (Misuse of subject's profile should not take place. Information should be used only for professional use and subject's privacy should be protected).
3. Natural language generation. (Output should be understandable to user)
4. Obtaining large training data set
5. Users accuracy in creating new classes
6. Identification of answer type required by user
7. Accuracy of the ontology used for input disambiguation
8. Downtime of server and external availability of external APIs and resources beyond our control.

## **4.3. Requirements**

### **4.3.1 Hardware requirements**

A Linux server with 30GB storage space and 8GB RAM is required to store and process the large amount of data.

### **4.3.2 Software requirements**

Web server: Apache

Database: MongoDB

Languages: Python, PHP, Javascript, HTML



## **Social DaTa Aggregator and Locator of Knowledge**

A web browser is needed to view the application.

Miscellaneous APIs to get data and for output presentation

### **4.3.3 Functional requirements**

- The product must provide an interface for the subject to register for an event and add/remove/edit links to his/her social media profiles.
- It must allow a user to create a new account to manage multiple events and view profiles.
- The user should be able to create and save multiple templates of profile classes beyond the lifetime of an event.
- The user should be able to view profiles of subjects for multiple events through his account.
- The system must analyse factual data from profiles and categorise them into classes like education, work, projects etc.
- The system must use NLP to analyse data from social profiles and categorise them into classes like social skills, popularity etc.
- The system must provide an interface for the user to define and add new classes to his/her existing templates.
- The user must have the flexibility to create new classes as found appropriate for an event
- Users must have the facility to select/reject candidates and view the candidate's status in an intuitive manner.
- The product must allow the user to enter a query.
- The product must be able to answer a query without much aid from the user.
- The product must be able to distinguish between a query that asks for an opinion or for data.

## Social DaTa Aggregator and Locator of Knowledge

- The system must be self-reliant in disambiguating inputs to the maximum extent possible.
- The software needs to maintain context to a limited extent.
- The software should be able to learn and accommodate new classes as per user input.

### 4.3.4 Non-functional requirements

- **Usability** - The system should be intuitive and user-friendly with simple documentation and quick help links to make user experience as seamless and easy as possible.
- **Performance** - The algorithm must be able to analyse and compile information as fast as possible. It should be able to answer user's queries in particular with minimal delay.
- **Resource Usage** - The resources such as server space, database and memory must be made use of the maximum extent possible to provide the best performance without overusing resources leading to downtime.
- **Reliability** - The system must provide accurate information as far as possible and try to avoid redundancies as some data may be the same across networks. It must also clearly state the assumptions and constraints regarding information collection, compilation and time constraints with respect to freshness of data.
- **Maintainability** - The product should be documented well and code standards followed to ensure maintenance and defect correction is made easy.
- **Availability** - The system is to be designed in a way that will not lead to failures and outages. Data processing should also be done in optimum time so that information about a subject is readily available. Caching must be used to hasten the process of fetching and displaying information.

## **Social DaTa Aggregator and Locator of Knowledge**

### **.3.5 Assumptions and Dependencies**

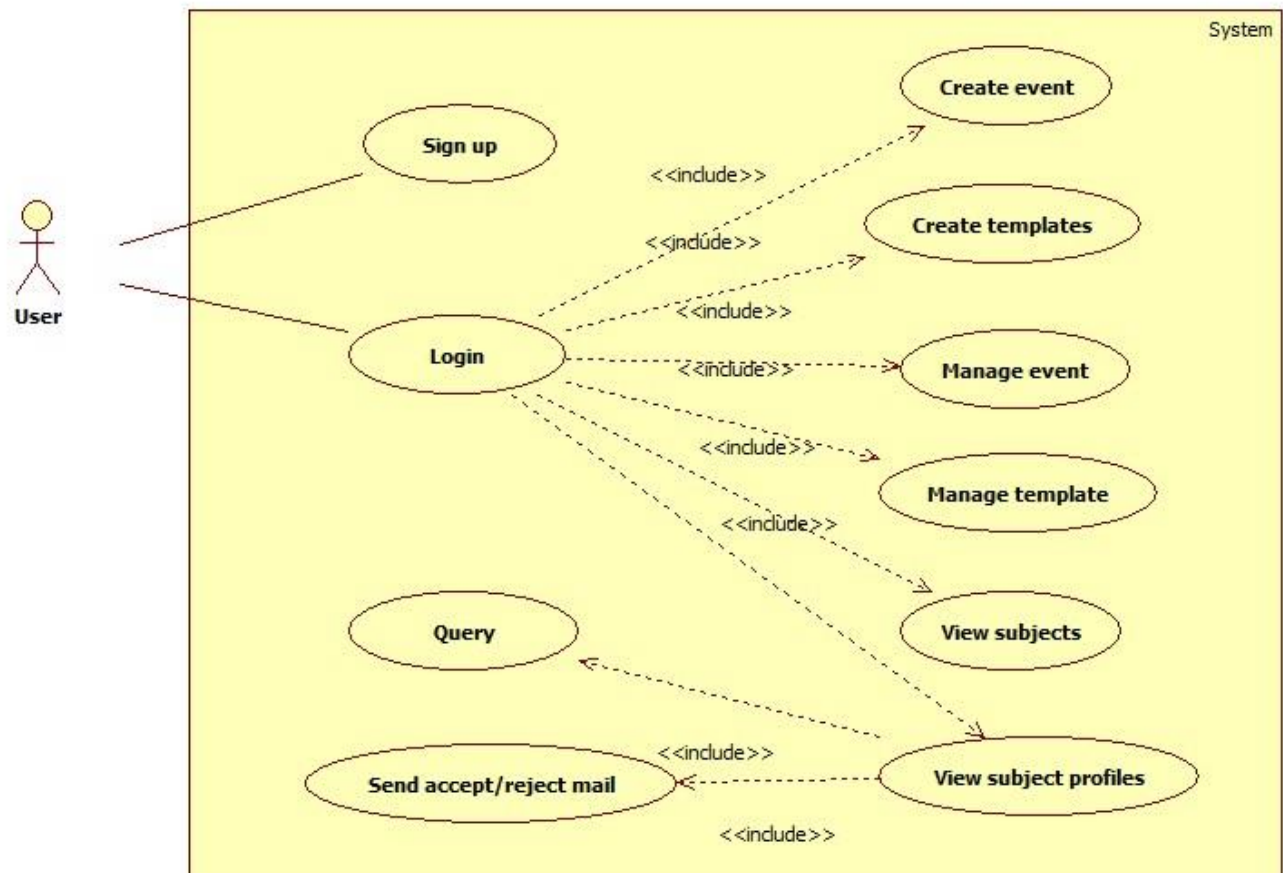
Some of the assumptions are:

1. The subjects are willing to submit their links
2. The subjects have given almost truthful information on the different social media
3. Only one user can host and manage a particular event
4. The Alchemy API has been trained and returns good results
5. The queries made by a user are legally allowed by him.

Some of the dependencies of our system are:

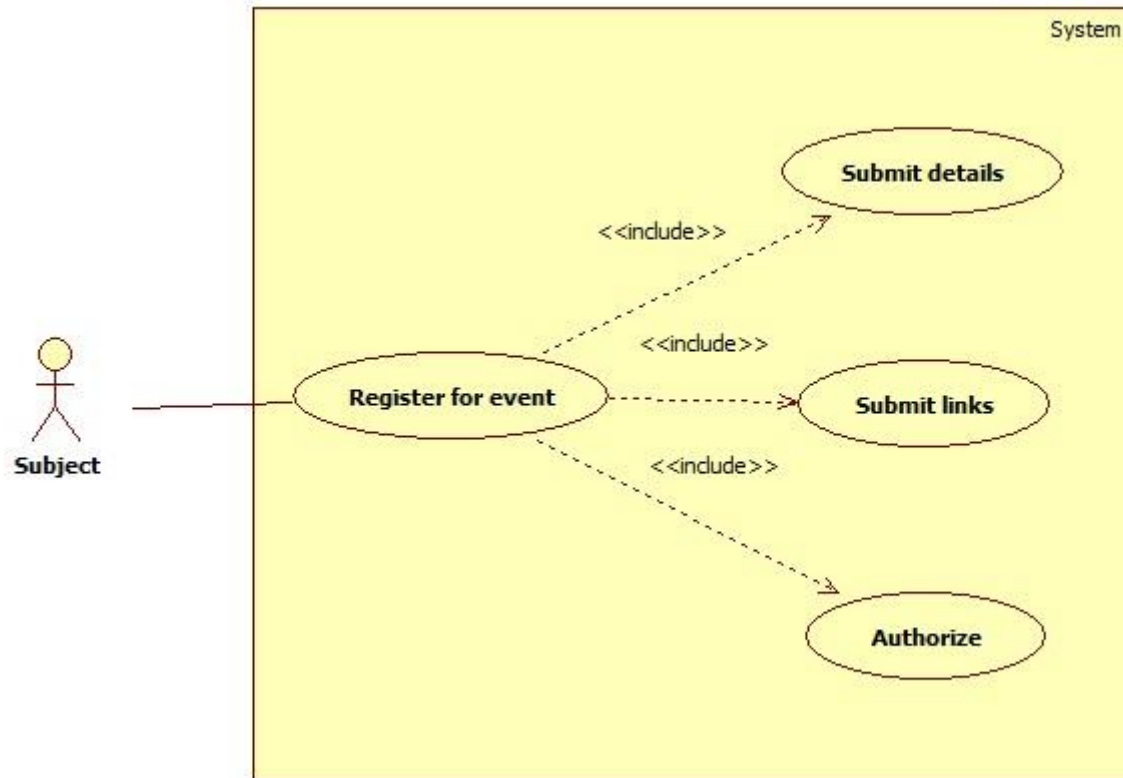
1. The GIT, Facebook and Linkedin API's. If the format of the API and data return format change, some parts of our code would be affected.
2. Alchemy API returns useful information that we process. If the API format is changed, our code may need to be changed.

## 4.4 Use Case diagrams



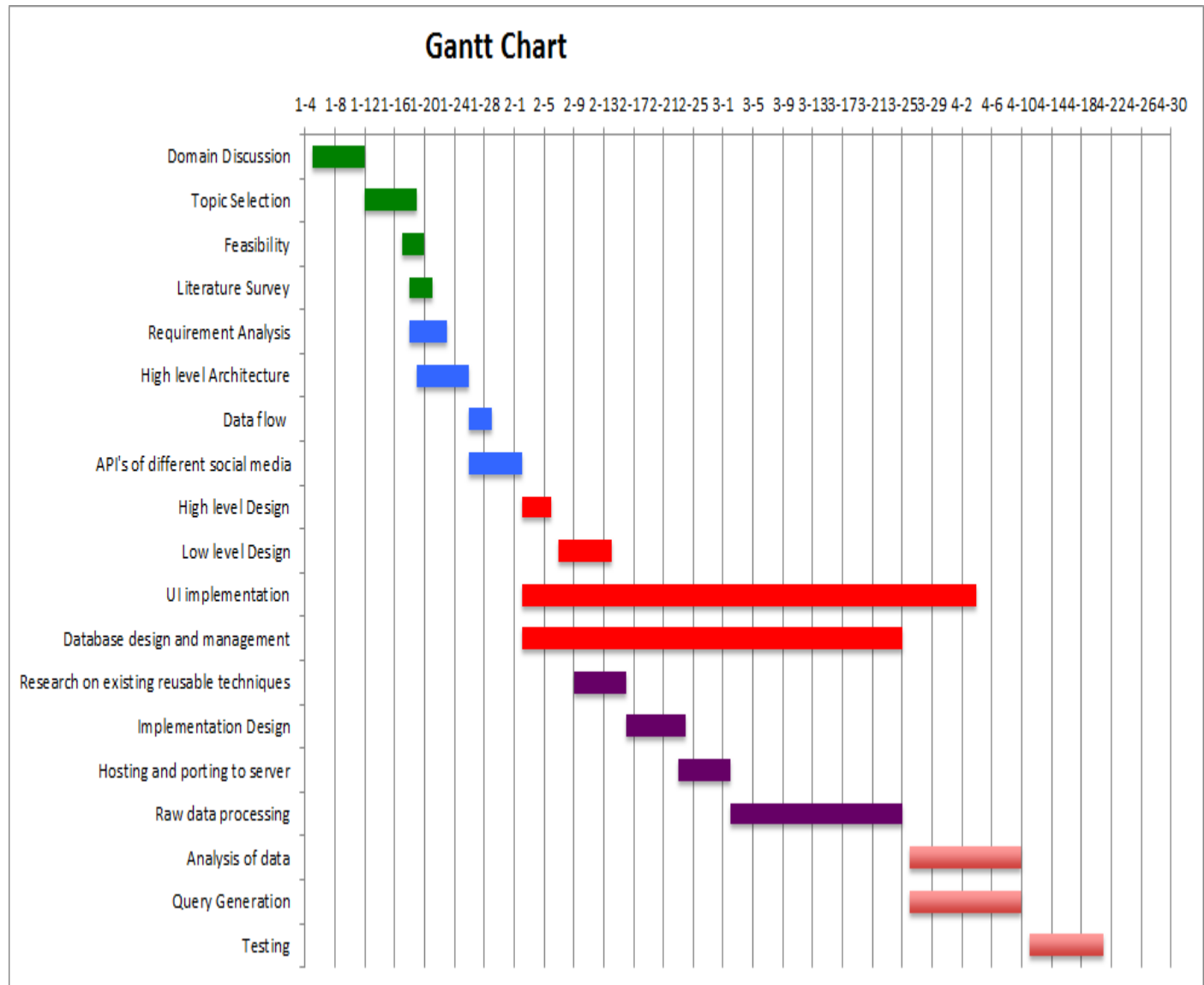
**Fig 4.1** Use case Diagram for the user.

## Social DaTa Aggregator and Locator of Knowledge



**Fig 4.2** Use case Diagram for the subject.

## 5. GANTT CHART



**Fig. 5.1 Gantt Chart**

## **6. SYSTEM DESIGN**

### **6.1. SYSTEM OVERVIEW**

The system provides an online platform, for people to both sign up for and host and select candidates. The system has different types of actors with different privileges and access rights.

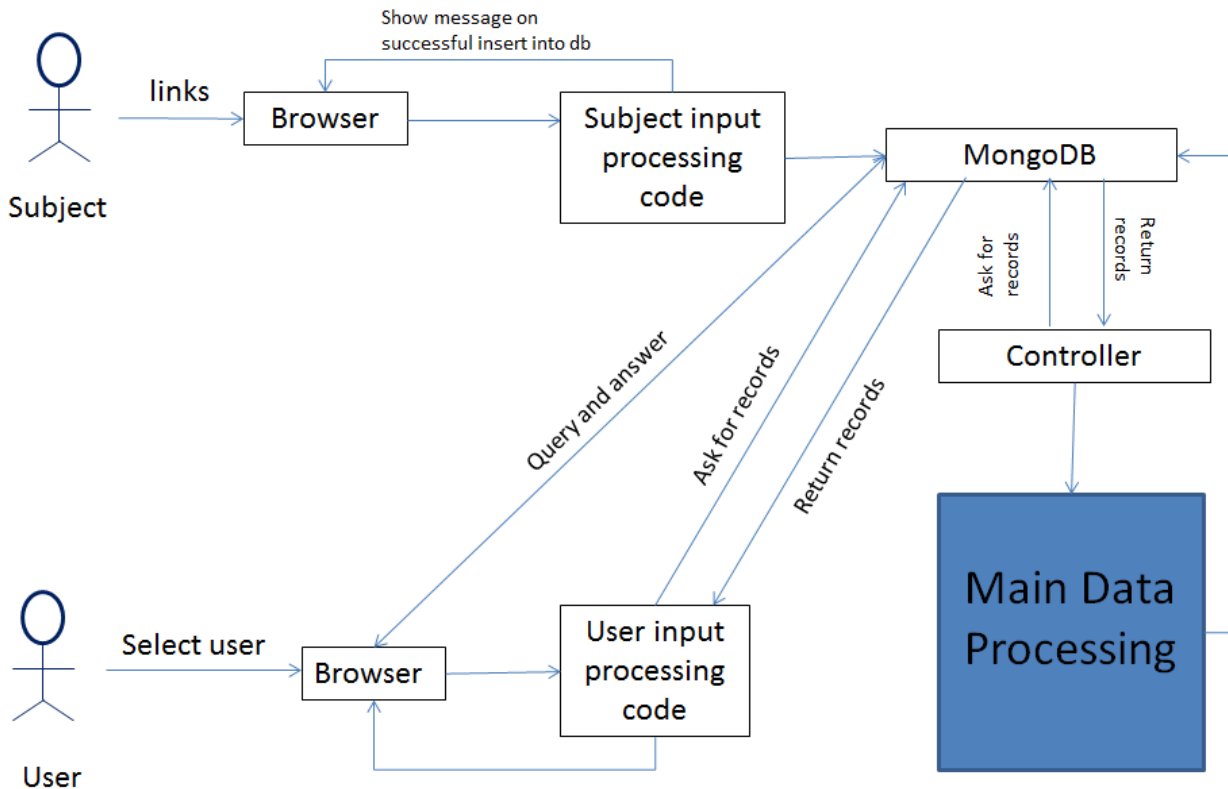
#### **6.1.1. Subject**

The subject is the actor who submits his links- Facebook, LinkedIn, GIT and others for screening. Depending on the links required by the event he is signing up for, he can choose one or more of these social media profiles. He can submit his links for one or more events. The links are stored by the system once submitted, however processing or reprocessing happens only on submission or updation of links by him. In general, whenever the subject signs up for a new event, his profile is regenerated by the system.

#### **6.1.2. User**

The user is the actor who views the profiles of subjects. The user can create different events in our application and allow subjects to register for them. The user can view the profile of any subject he wishes for those corresponding events once the data has been processed and stored by the application. He can accept, reject or park the selection using our site. He can create different templates so he can view information only relevant for him. He can however, use default templates created by us.

## 6.2. SYSTEM ARCHITECTURE



**Fig 6.1** Architecture Diagram depicting the different modules of the system.

The above image describes the various modules that will be used to achieve complete functionality of the system.

Each of these modules has a different function, and each of the modules are independently needed by the system and cannot be merged with any other module.



## **6.3 Decomposition Description**

### **6.3.1 Browser:**

The interface for both the user and the subject is the browser.

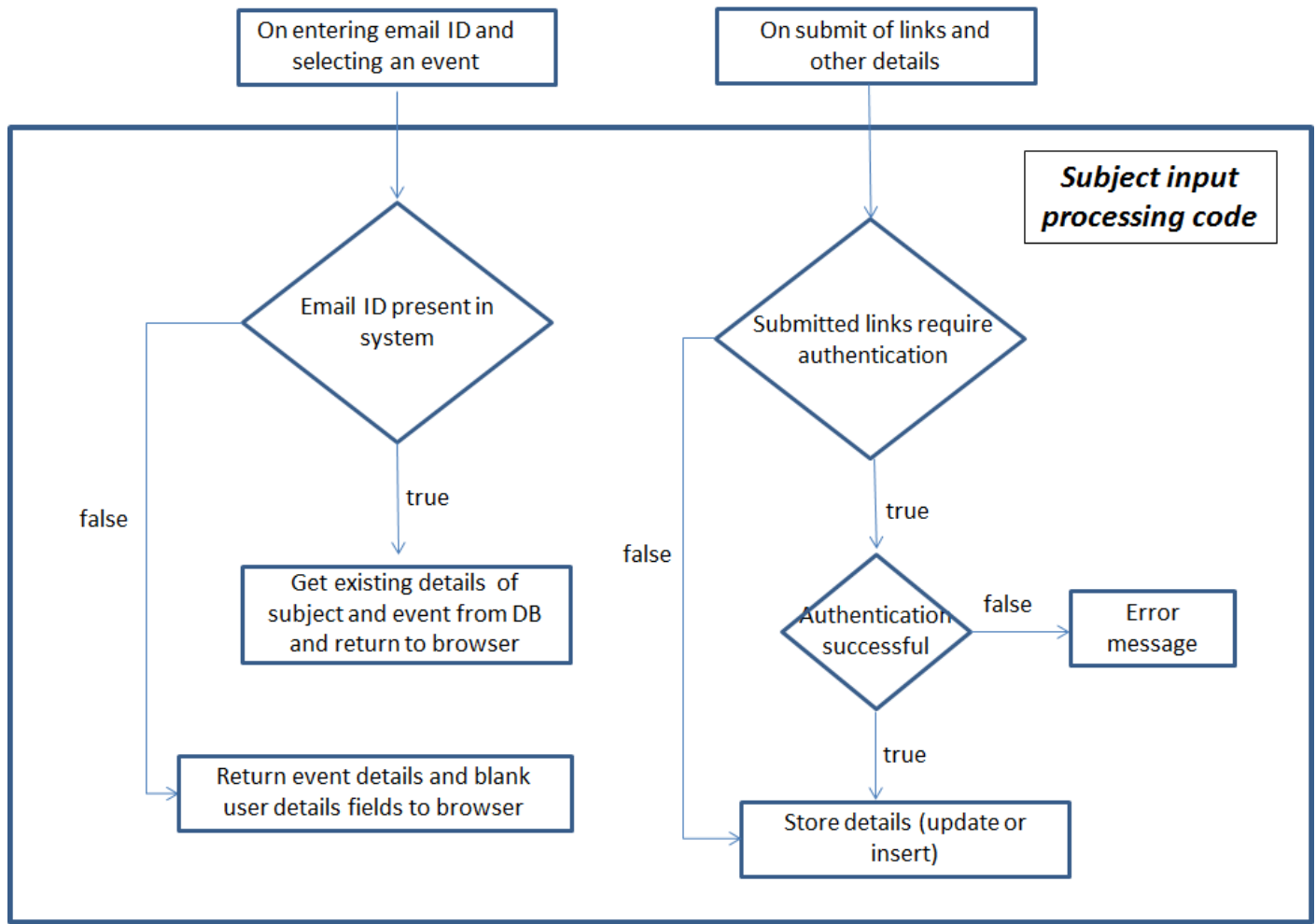
The SUBJECT submits various links and gives permission as to which link can be used for which event.

The USER can create events, select candidates and also choose templates to select the fields he wishes to view in the SUBJECT's profile.

### **6.3.2 Subject Input Processing Code**

This particular module takes the information submitted by the subject and stores it in the database. It also handles automatically filling the subject's details if the email ID entered by the subject is already present in the system. The module flow is as follows:

## Social DaTa Aggregator and Locator of Knowledge



**Fig. 6.2** Activity flow diagram for the system.

### 6.3.3 User Input Processing Code

The user is allowed to have his own account, where all his events and subjects can be viewed. The user creates an account using the “sign up” link. The system checks for a valid email ID using regex matching. Password strength is also checked. Next, once the sign-up process is successfully completed, the user can sign in to his account. He can create an event by filling in the relevant details. The assumption

## **Social DaTa Aggregator and Locator of Knowledge**

made here is that no organisation can have multiple events of the same name. This is checked before the event is inserted into the database. The user also selects the links he wants the applicants of his event to provide. Once subjects enter links and register for the event, the user can see a list of subjects who have signed up. He can view their consolidated profiles once the data has been processed and if the subject has chosen to provide links. The user is allowed to select and reject candidates using the portal itself. He can customize the mail to be sent on acceptance or rejection. on or use the standard email. He can also create templates which define the fields he wants to see about the user that he can regularly use and retrieve whenever he uses his account. He can also query information about the subject. In addition, the user can also perform standard tasks such as editing or deleting an event, editing his profile information etc.

### **6.3.4 MongoDB**

All data is stored in collections in MongoDB. The data design is described in the forthcoming sections.

### **6.3.5 Controller**

The controller module requests for data from different social media and stores the raw data got from the different profiles in our collections. This data is then processed in the Main Data Processing Unit. The controller calls each of the processing unit for each of the social media profiles sequentially.

The controller runs continuously in the background. It spawns multiple processes (currently 5 processes and can be scaled up if necessary depending on the load). These processes run in parallel to process the links of multiple subjects at once.

## **Social DaTa Aggregator and Locator of Knowledge**

Hence, at any given time, five subjects can be processed in parallel. The controller pulls unprocessed subject records from the subject collection by checking a flag and passes those records to the main processing unit.

### **6.3.6 Main Processing Unit**

This contains the python code that is used to convert raw data into natural understandable language. The data is fetched from the API and the corresponding module is used to parse this data into the previously decided format. The pertinent fields are obtained from the raw data and stored in the NLG collection under the field names decided by us. Hence, the raw data from all sources is consolidated, analysed and stored in a standard format.

### **6.3.7 Analysis of Information**

This module contains the python code which performs analysis on the GIT, LinkedIn and Facebook profiles of the user. It is a part of the main processing module. If the data is present and the event asks for the particular profile information, this module converts the unstructured raw data into meaningful structured format and also uses the Alchemy API to perform certain analyses like concept tagging, sentiment analysis, taxonomy, text summarisation etc to obtain meaningful information about the subjects. Other information like the friends the user commonly interacts with, the topics that interest him, his hobbies, his usual teammates etc are all extracted as well.

## **Social DaTa Aggregator and Locator of Knowledge**

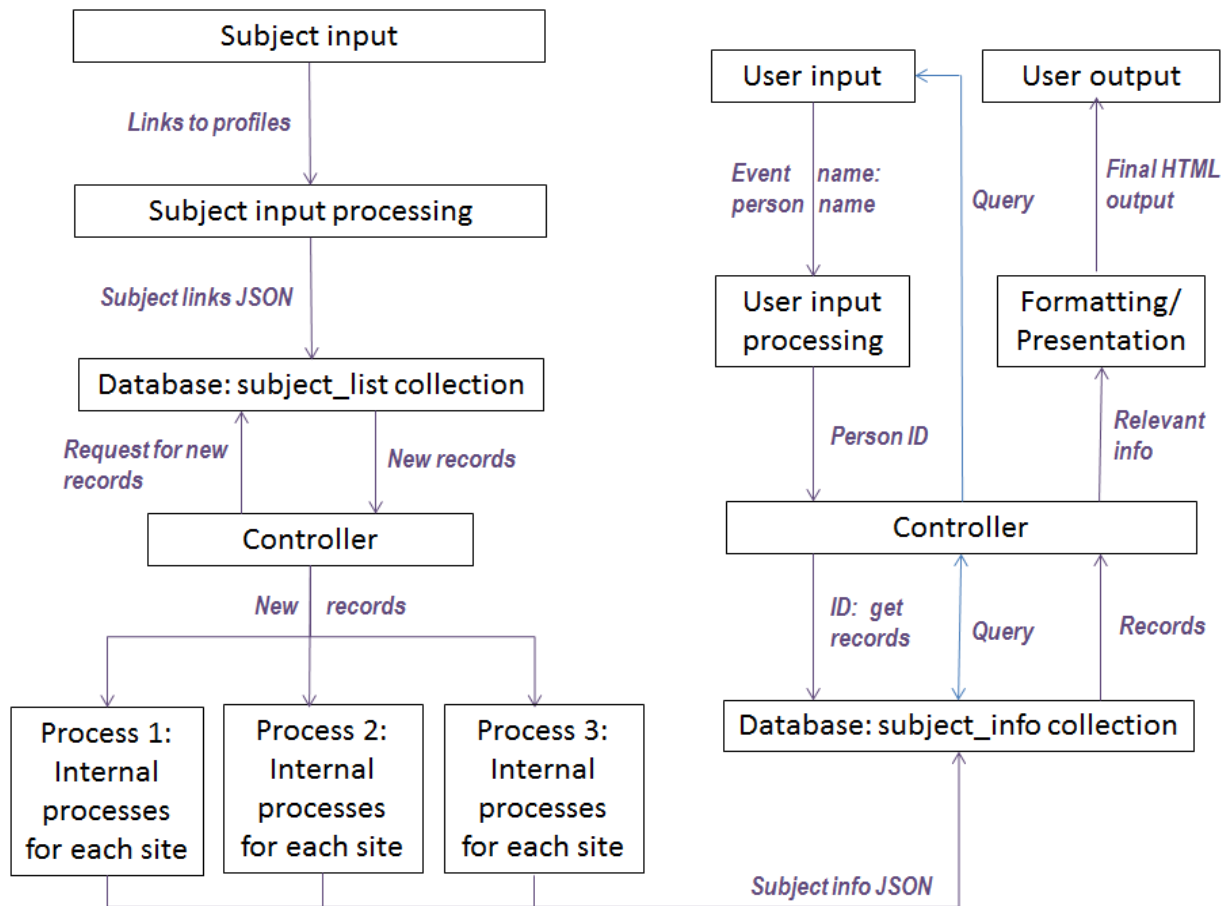
### **6.3.8 Query**

The query module is used to process, understand and reply to user queries. Queries can be written in simple english. The query backend uses Alchemy to obtain taxonomy of the sentence. Using the word in the query and the taxonomy, the class of the query is obtained i.e., is the user asking about a programming language, or a technology that the subject is experienced in; or is he asking about a hobby that the subject might indulge in. This class is then mapped to one of the existing categories from our defined information collection. The corresponding field in the database is checked for the answer to the query and natural language with the final answer is generated.

## **6.4. DATA DESIGN**

### **6.4.1 Data Flow**

## Social DaTa Aggregator and Locator of Knowledge



**Fig. 6.3** Data flow Diagram

The flow of data is as depicted in the diagram above.

The architecture above described the main components shown in the above diagram.

## Social DaTa Aggregator and Locator of Knowledge

### 6.4.2 Data Dictionary

The format of the collection used in our Mongo database is as follows:

#### 6.4.2.1 User

```
{
  "_id" : ObjectId("55113de348484d632e8b4569"),
  "user_id" : "RakBha1427193315",
  "events" : [
    "Fb Hackathon",
    "fb"
  ],
  "templates" : [
    "Template 2,technical,work,social,non-tech",
    "Hackathon Template,education,technical,social"
  ],
  "name" : "Rakshitha Bhat",
  "email" : "rakshitha03@gmail.com",
  "password" :
"$2y$10$1cbsECct4fqTP8vOrvdl3OWmqbGhF26WhezGEqEJtITJ6to4NwZK."
}
```

#### 6.4.2.2 Subject

```
{
  "_id" : ObjectId("5535cf1b48484dc9738b4567"),
```

## Social DaTa Aggregator and Locator of Knowledge

```
"subject_id" : "navbel1429589785",
"links" : {
  "git" : "NaveenBellary",
  "fb" : "https://www.facebook.com/bellary.naveen",

  "linkedin": "https://www.linkedin.com/profile/public-profile-
settings?trk=prof-edit-edit-public\_profile"
},
"email_id" : "bellary.naveen@gmail.com",
"name" : "naveen bellary",
"nlg_done" : "false",
"process_begun" : "false",
"events" : [
  "All Three,true,true,true,Pending"
]
}
```

### 6.4.2.3 Events

```
{
  "_id" : ObjectId("55352e2148484db2688b4567"),
  "event_name" : "All Three",
  "event_org" : "RRS",
  "required_links" : [
    "fb",
    "git",
    "linkedin"
  ]
}
```



## Social DaTa Aggregator and Locator of Knowledge

```
],  
  "desc" : "Let's hope it works",  
  "end_date" : NumberLong(1432267200),  
  "owner" : "sanjanashankar151@gmail.com"  
}
```

### 6.4.2.4 Raw Data

```
{  
  "_id" : ObjectId("551a9e12308b356fde59cbb4"),  
  "gid" : "SanjanaS801",  
  "email" : "sanjanashankar151@gmail.com",  
  "git" : {  
    "general" : {  
      "site_admin" : false,  
      "subscriptions_url" :  
        "https://api.github.com/users/SanjanaS801/subscriptions",  
      "gravatar_id" : "",  
      "id" : 5665631,  
      "followers_url" :  
        "https://api.github.com/users/SanjanaS801/followers",  
      "following_url" :  
        "https://api.github.com/users/SanjanaS801/following{/other_user}",  
      "followers" : 0,  
      "type" : "User",  
      "public_repos" : 2,  
      "gists_url" : .....
```

## Social DaTa Aggregator and Locator of Knowledge

```
    }  
  },  
  "linkedin" : { ..... },  
  "fb" : { ..... }  
}
```

### 6.4.2.5 Nlgdata

```
{  
  "_id" : ObjectId("551a9e12308b356fde59cbb5"),  
  "email" : "sanjanashankar151@gmail.com",  
  "git" : {  
    "gitid" : "SanjanaS801",  
    "name" : "",  
    "blog" : "",  
    "location" : "",  
    "bio" : "",  
    "project_names" : "LogicGateSimulator,RegisterAllocation",  
    "languages" : "Python",  
    "technical_followers" : 0,  
    "technical_following" : 0,  
    "technical_following_links" : "",  
    "technical_subscribed" : 8,  
    "technical_subscribed__links" :  
    "https://api.github.com/repos/rashrag/RegAlloc,https://api.github.com/repos/Sanjan  
aS801/LogicGateSimulator,https://api.github.com/repos/SanjanaS801/RegisterAllocat  
ion,https://api.github.com/repos/rashrag/android_set4,https://api.github.com/repos/
```

## Social DaTa Aggregator and Locator of Knowledge

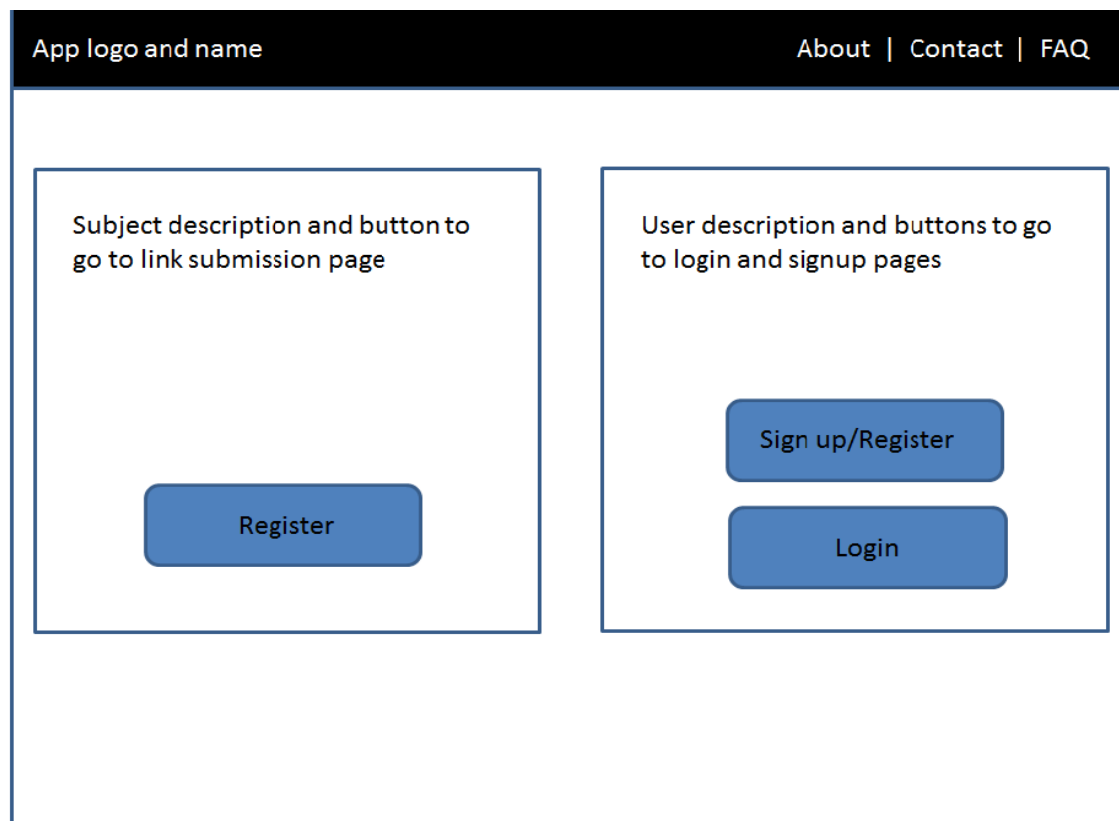
```
Rakshitha03/Set4Multi,https://api.github.com/repos/rashrag/nlp-eval-  
day1,https://api.github.com/repos/rashrag/nlp-eval-  
day2,https://api.github.com/repos/rashrag/PersonProfile", "orgs_subscribed" : 0,  
  "orgs_subscribed__links" : "",  
  "activity_git" : 9  
},  
"linkedin" : { "liurl" : .....},  
"fb" : { ..... }  
}
```

## 6.5. HUMAN INTERFACE DESIGN

### 6.5.1 Screen Images

#### 6.5.1.1 Home Page

The idea was to present a straightforward and simple design. A brief description of the actor's role is presented with the links they will most likely use. A menu bar is provided on every page with other helpful links as is the usual format of most similar web applications.

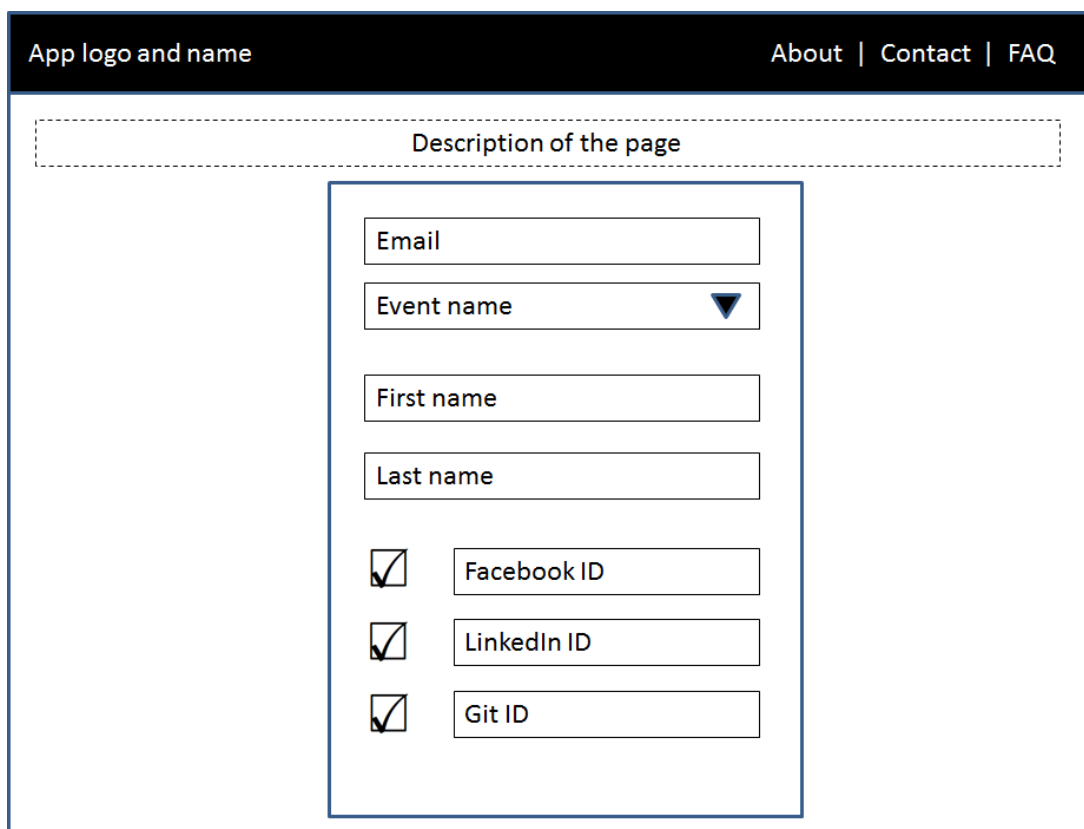


**Fig. 6.4** Home page

## Social DaTa Aggregator and Locator of Knowledge

### 6.5.1.2 Subject Page

The subject registration page is again designed to be minimal with quick and intuitive auto fill of the form on selection of an event. The authentication windows open in separate modal window so as to preserve the form data until submission. Appropriate checks for exceptions have been implemented as well.



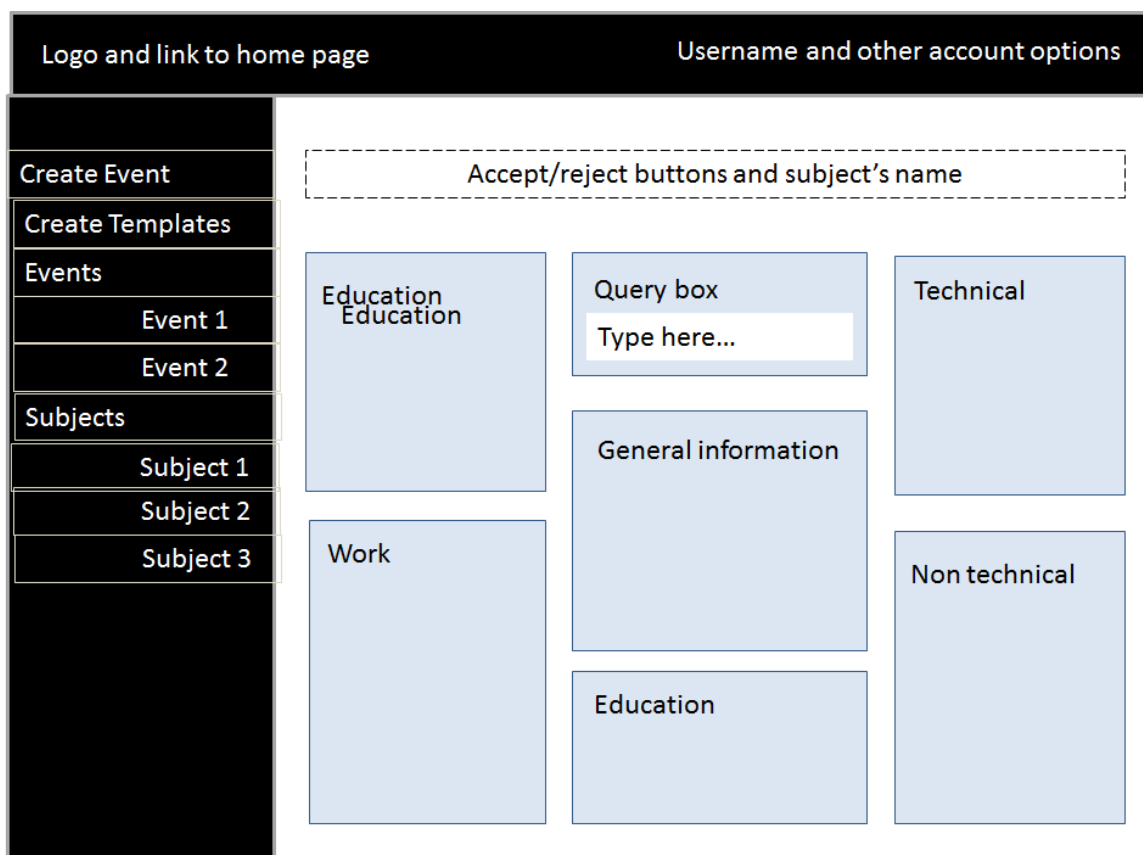
The figure shows a web form for subject registration. At the top, there is a dark blue header bar with 'App logo and name' on the left and 'About | Contact | FAQ' on the right. Below the header is a dashed box labeled 'Description of the page'. The main form area contains a blue-bordered box with the following fields: 'Email' (text input), 'Event name' (dropdown menu with a downward arrow), 'First name' (text input), 'Last name' (text input), and three social media ID fields: 'Facebook ID', 'LinkedIn ID', and 'Git ID'. Each social media ID field has a checked checkbox to its left.

**Fig. 6.5** Subject Page where the subject registers for an event

## Social DaTa Aggregator and Locator of Knowledge

### 6.5.1.3 User Page

The user page was designed to have all important functionalities readily available in the side menu. The user needs to click on the event under the events list to trigger a dropdown of all subjects associated with the event. Clicking on a subject then displays his information. Templates can also be easily selected from the side menu. Only general actions dealing with editing the user's profile and logging need to be selected from the top menu.



**Fig 6.6** User profile page where the user can access his profiles and create an event.

## **6.6 Design Challenges and Decisions**

### **6.6.1 Data collection**

Data collection posed two questions : when and how.

The “how” part of the challenge was easily solved thanks to the availability of APIs and wrappers in several languages for their extraction. The biggest challenge in obtaining data was the lack of a standard process. Each social media network had its own process for data retrieval and had to be separately implemented. In our particular project, Git data required no authentication, LinkedIn used oauth 2.0 and returned a token for use to fetch data within the expiration time of 30 days. However, Facebook, which also required an oauth authentication mechanism, provided a very small window before the oauth token expired. <sup>[10]</sup>

Hence, the decision made was to store the LinkedIn token in a separate collection and fetch the data later, fetch Facebook data as soon as the subject submits and authenticates the links as the token could expire if the data collection was delayed and fetch Git data also before processing begins. We do not fetch Git and LinkedIn data also on link submission as with Facebook since that would result in the user having to wait until data was fetched before a response could be sent to the browser about the status of the submission. This would lead to a bad user experience.

### **6.6.2 Permissions**

The next challenge was to obtain permission from the subjects to use their social media profiles. We decided to allow the user to view the data fields we would be obtaining and gave the the option to authorize our application. This means that our

## **Social DaTa Aggregator and Locator of Knowledge**

data will be taken with voluntary permission from willing subjects as opposed to using convoluted methods of obtaining permission secretly without the subject's knowledge.

### **6.6.3 Privacy**

The next decision to be made was whether the user is allowed to make a few links compulsory for subjects to submit or not. In the end, the decision was not to force the subjects to submit their links. All subjects will have an option to not provide any links, choose a subset of links that the event claims to require, cancel authentication and hence, not provide information from a particular network and also to review and change the permissions asked for.

However, this decision resulted in a new challenge. Our original design was to store all the links provided by the subject. This meant that during processing, information would be taken and consolidated from all the networks to which the subject had authorized our application. However, if he is given the option to choose networks for an event, then the application should not present data from other networks to the user conducting that particular event. For example, consider Event1 created by User1 requiring Facebook and LinkedIn links and another event, Event2 created by User2 requiring Git and LinkedIn links. Now, User1 should not be able to see data from Git in the consolidated profile of the subject presented to him and User2 should similarly not be allowed to see Facebook data of the subject. Our initial design would have presented a consolidated profile from all the links stored.

This issue was overcome by creating separate consolidations. The information collection would now hold separate structures for Git and LinkedIn, Facebook and LinkedIn, Git and Facebook and any other possible combination of the links.

The tradeoff here is between time and memory. This consolidation could have been done in real-time i.e. when the user requested the subject's data and thereby



## **Social DaTa Aggregator and Locator of Knowledge**

reduced the storage of redundant data that the approach we selected requires. However, this would mean that the user would have to wait for several minutes after clicking on the subject's name before his data was available. This would be bad customer experience. In this light, having redundant data was a better approach.

### **6.6.4 Data Storage**

The data from social media networks is large. It is presented in the form of json objects containing all the requested fields. With this in mind, it was a better choice to use a non-relational database like MongoDB. With such a database, it is a simple matter of creating another JSON-like structure with our information fields and inserting that structure into the appropriate collection. This eliminates the need to maintain schemas and structural integrity that relational databases require.

### **6.6.5 Running the controller**

One of the main challenges was that we developed a web application with a frontend in Javascript and HTML interfacing with a PHP backend. However, Python was a better suited language to use for NLP required in analysing social media data. On the other hand, the application did not require a full Python backend which would have necessitated a framework such as Django, adding to the complexity.

However, this brought up a question. How often does the main processing module need to be run? Does it have to be run each time a new subject record is inserted into the database? If so, how does the Python program know that a new record has been inserted? How would we deal with concurrency issues in that case? Can the program be run at specified intervals, if so, how often?

## **Social DaTa Aggregator and Locator of Knowledge**

The solution was to eliminate any interactions between the PHP and Python scripts which would have added unnecessary complexities. The PHP handles the subject record insertion. In that module, two flags are added to the subject record - `nlg_done` and `process_begun`. Both are set to false at the time of insertion. The Python script runs periodically, checks for records with `process_begun` = false and sets this flag to true before beginning processing. Once the NLG record has been inserted, `nlg_done` is also set to true. If there is a failure anywhere, both flags are reset to false so that the record can be processed at a later time.

## **7. IMPLEMENTATION**

### **7.1 Essential Components:**

As part of the environment setup for the project we bought a linux server and a domain name: profile.personprofile.in .

We set up Apache server, MongoDB, PHP and configured the php.ini file to communicate with mongodb.

### **7.2 MongoDB:**

The main database setup by us is named "stalkdb". The following are the collections we used : user, subject, event, rawdata, nlldata, litokens.

The data stored is same as that discussed in the design. We used both PHP and Python to insert data into MongoDB collections.

As we have multiple social media, each with overlapping information, there was a need to ensure that if data from more than one link was requested, duplicate information was not displayed. Hence, we use a cosine similarity measure to check if fields that could overlap among social media contain the same element. For example, school name could be present in both Facebook and LinkedIn. In this case, we check the education fields returned by both Facebook and LinkedIn and combine them. We finally store the combined record as "li\_and\_fb" or a similar combination depending on the social media information being combined.

## Social DaTa Aggregator and Locator of Knowledge

### 7.3 Web Pages:

We have two different actors - user and subject. The subject can select an event, select links to submit for the event and authorize them. The user can sign up for or login to his account after which he can create events and templates, view subjects who have signed up for his events and other general actions.

The pseudo code for the actions performed by each actor is as below:

#### 7.3.1 Subject

Enter email id

Select event

if subject has already signed up for event:

    alert subject

else

    Get links required by the event

    if email id already in subjectdb:

        load corresponding network links if already present

    else:

        allow user to enter links

    Select links subject is willing to submit for this event

    Check if any of the links require authentication

    if yes:

        perform authentication procedure

    else:

## Social DaTa Aggregator and Locator of Knowledge

return

display success/failure message to subject

### 7.3.2.User:

Enter email id and password to enter user page

if authenticated:

Create event:

Enter name, description and end date

Select links required for event

Create Template:

Enter fields user would like to view

Store the template for future use

Select Event:

Click on event name to view list of subjects who have registered.

Alert will be displayed if the event is within seven days of its end date after which it will be automatically deleted with all its data.

Select Subject:

Click on fields to hide unwanted information

Use query box to enter queries about a person

View information

Select Template:

Click on template

Auto refresh subject currently on display to reflect chosen template

## **7.4 DATA COLLECTION**

The data collected was broadly classified into the following fields:

Education, Technical, Projects, Work, Social, Popularity, Non\_technical skills, Tech\_Interests, Non\_Tech\_Interests

### **7.4.1 Controller**

The controller module has two parts - the proc\_test.py file and the controller.py file. The proc\_test file is run as a background process. It checks if there are any unprocessed records in the database by getting those records whose process\_begun flag is set to false. Five records are pulled at a time and up to five processes are spawned. Each process invokes the controller.py file with its assigned record to process.

The controller.py file checks which social network links are present in the record and then sends the record for processing to the corresponding modules. The modules fetch and analyse the data from the networks and insert a consolidated profile NLG record into the database. They also handle eliminating redundancies in information between networks processed earlier. When all the processing is done, they return a boolean value depending on the success or failure to the controller which returns the same value to the proc\_test file.

If all the modules return true, then the controller will know that processing was successful and will set the nlg\_done flag to true. Else, it will reset the process\_begun flag to false to try processing again at a later time. Some common issues that caused such a case were the rate limit for the API being exceeded, logical errors etc.

## Social DaTa Aggregator and Locator of Knowledge

proc\_test:

Background process that is continuously running.

Checks if any records are present in the database with process\_begun flag set to false:

if records with process\_begun > 0:

Pull up to 5 unprocessed records from the database

Create 5 processes

Assign records to the processes:

Set process\_begun to True

Call processing function in controller.py file

If return value of network's processing function is true:

set nlg\_done to true

else:

rollback process\_begun to false

else:

sleep for 90 seconds

controller.py:

Extract email id from input record

Check which links are present in the input record

Pass email id to git processing functions, if git is present

Pass email id to linkedin processing functions, if linkedin is present

Pass email id to facebook processing functions, if facebook is present

if each of the above functions returns true:

return true

else:

send email to creators with error

## Social DaTa Aggregator and Locator of Knowledge

return false

### 7.4.2 Data fetching and processing modules

The individual data processing modules are written specifically for the social networks used. These modules fetch the data if it is not already present. For example, the LinkedIn module fetches data using the token stored in the litokens collection. The Git module fetches data directly from the API since Git does not need authentication.

The modules then map the fields returned by the API to the fields stored in our nlldata collection. A standard format like <field\_name>\_names and <field\_name>\_desc is used for fields that contain a name and a description like education and projects. Using Alchemy, some amount of NLP is also done in terms of taxonomy recognition, text classification, concept recognition etc. which help in understanding the text to obtain more information about the subject. For example, analysing LinkedIn project descriptions gives us more information about the subject's skills and knowledge than the few that have been manually tagged by him.

Some analysis of numerical results is also done like inferring that the subject prefers smaller groups by counting the number of people he frequently interacts with online or deducing who his usual teammates are on projects.

Redundant data removal is also done in these modules. Each module has to ensure that when combining information with another network, the facts are not repeated. This is done by using a cosine similarity measure.

#### 7.4.2.1 GIT DATA

read email id and fetch git id. update information in rawdata and make it more readable in nlldata



## Social DaTa Aggregator and Locator of Knowledge

general information: "[https://api.github.com/users/"+gitid](https://api.github.com/users/)

languages and projects: "<https://api.github.com/users/gitid/repos>"

followers: "<https://api.github.com/users/gitid/followers>"

following: "<https://api.github.com/users/gitid/following>"

subscriptions: "<https://api.github.com/users/gitid/subscriptions>"

orgs: "<https://api.github.com/users/gitid/orgs>"

events: "<https://api.github.com/users/gitid/events>"

store raw data in rawdata collection

store display data in nlldata collection

### 7.4.2.2 LinkedIn Processing:

Linkedin Processing involves:

linkedinProcess:

get oauth token from litokens table using email id passed as input

get data from linkedin profile. Data fields fetched are:

'id', 'first-name', 'last-name', 'headline', 'industry', 'num-connections', 'num-connections-capped', 'summary', 'location', 'specialties', 'positions', 'last-modified-timestamp', 'associations', 'interests', 'publications', 'patents', 'languages', 'certifications', 'courses', 'volunteer', 'three-current-positions', 'three-past-positions', 'num-recommenders', 'skills', 'recommendations-received', 'following', 'job-bookmarks', 'suggestions', 'member-url-resources', 'related-profile-views', 'honors-awards', 'educations', 'projects'

store data as per the design of linkedin record in nlldata

call alchemy API to get more information from project/paper descriptions

insert LinkedIn record into nlldata

## Social DaTa Aggregator and Locator of Knowledge

check if git data already exists

if yes:

remove redundancies:

call cosine similarity functions to check if overlapping  
fields have common values

if values are not common add value to combined field  
add git\_and\_li json to nlldata

### 7.4.2.3 Facebook Processing:

Facebook Processing involves the following two things : i) User Login ii)  
Facebook Processing

userLogin:

Check if the user is already connected to facebook .

logYes:

ask user if he wants to submit data from this account.

yes:

User logged in to the app.

Access token received.

call fbProcess

no:

Log out the current user and ask user to sign in using  
his facebook account.

User logged in to the app.

Access token received

call fbProcess

## Social DaTa Aggregator and Locator of Knowledge

logNo:

Ask user to sign in to his facebook account.

User logged In.

repeat the first logYes block

fbProcess:

Data fields are fetched

The fetched fields are:

'id', 'name', 'email', 'birthday', 'work\_history',  
'education\_history', 'likes', 'statuses', 'interests', 'activities', 'groups', 'posts',  
'photos', 'books', 'movies', 'music', 'friend\_count'

store data as per the design of fb record in nlldata

check if git data already exists

if yes:

remove redundancies:

call cosine similarity functions to check if overlapping

fields have common values

if values are not common add value to combined field

add git\_and\_fb json to nlldata

else:

add fb json to nlldata

check if linkedin data already exists

if yes:

remove redundancies:

call cosine similarity functions to check if overlapping

fields have common values

---

## Social DaTa Aggregator and Locator of Knowledge

```
        if values are not common add value to combined field
        add li_and_fb json to nlldata
    else:
        add fb json to nlldata
    check if git and linkedin data already exists
    if yes:
        remove redundancies:
            call cosine similarity functions to check if overlapping
fields have common values
            if values are not common add value to combined field
            add git_and_li_and_fb json to nlldata
    else:
        add fb json to nlldata
```

## 7.5 Analysis

### 7.5.1. Analysis on GIT Information

GIT data was analysed to obtain the popularity of a person by analysing the number of followers. His interest can be analysed using the links and orgs he has subscribed to.

### 7.5.2. Analysis on LinkedIn Information

LinkedIn data was analysed to obtain the comfort level of the person in working with other people. We obtain the data about his usual project partners and analyse

## **Social DaTa Aggregator and Locator of Knowledge**

whether he prefers to work with the same people or as part of larger teams. We also run the Concept Tagging function of the Alchemy API to extract the concepts or technologies from the project descriptions. This enables us to provide full information that may not have been explicitly added by the subject. The various fields provided in the LinkedIn API are also combined to form a succinct description of the subject's education, projects etc.

### **7.5.3 Analysis on Facebook Information**

Facebook data was analyzed to obtain the social aspect and interests of the subject. We obtain data about his like pages, statuses and groups and analyze this data to find out which books has the subject read or what music has he listened to. Similarly we also give a count of his Facebook friends and also the average number of comments made on his/her post so that we can gauge the popularity of the person. We also use Alchemy API to perform sentiment analysis of the statuses and give a sentiment score of the same. We also collect data about his/her pictures and posts to analyze who is tagged more often with the subject to get an idea about the friend circle the subject is a part of.

## **7.6.QUERY**

The query module is used to accept queries from the user and return an answer in natural language.

```
input query and send to php script
alchemyapi=new AlchemyAPI();
response = alchemyapi->taxonomy('text', query);
```

---

## **Social DaTa Aggregator and Locator of Knowledge**

if response in languages:

    check in permissionobj.language

else if response in education:

    check in permissionobj.education

else if response in work:

    check in permissionobj.work

else if response in projects:

    check in permissionobj.projects

else if response in skills:

    check in permissionobj.skills

breakdown query and display question specific information

## **8. INTEGRATION**

We followed a bottom up approach in integrating the different components. We developed smaller modules and combined them to obtain bigger combined modules. The following modules were developed by us and integrated as shown.

### **8.1 GIT and LinkedIn**

We developed GIT modules to obtain git data and convert it to relevant information. We also developed LinkedIn modules to do the same. This resulted in some of the information being common between the two. We needed to integrate them in such a way that duplicate information would not be stored or displayed. There may also be some events which allowed the use of only one of these two events. These were taken care of when integrating the two. The information collected was stored in mongodb collections.

### **8.2 GIT, LinkedIn and Facebook**

We developed GIT and LinkedIn modules to obtain data and convert it to relevant information. We also developed facebook modules to do the same. This resulted in some of the information being common between the three. We needed to integrate them in such a way that duplicate information would not be stored or displayed. There may also be some events which allowed the use of only one or two of these events. These were taken care of when integrating the two. The information collected was stored in mongodb collections.

## **8.3 Controller, Git, LinkedIn and Facebook**

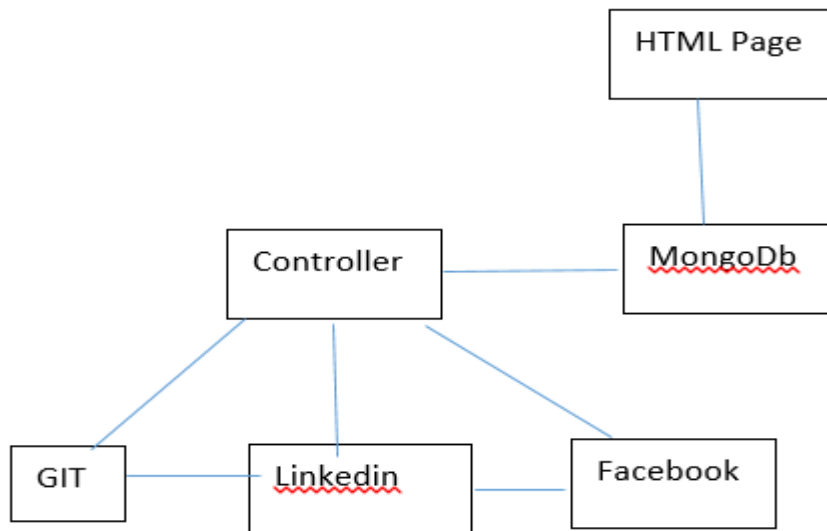
A controller was used to process the GIT, LinkedIn and Facebook information sequentially and as a background process if the data was not yet obtained for this person. Up to five people's data can be processed at once through multi processing. Some of the information may already be stored but the subject may not have allowed a particular event to access information from that link. This event specific information is part of integration process and has been taken care of by us. The information collected was stored in mongodb collections.

## **8.4 HTML and MongoDB**

Front end pages were created using HTML, JavaScript and PHP. The data and information collected and processed above was stored in different mongodb collections. As part of our integration the right data from the collections must be displayed on the front end pages dynamically.



## 8.5 Final Integration



**Fig 8.1** *Integration of various modules*

## 9. TESTING

### 9.1 Testing Strategies

The testing follows Bottom up approach. The bottom up approach is used for testing small components, and finally to test them as one complete system. In this project initially each of the modules were subjected to unit testing and then as they were integrated to higher modules, the interfaces were also tested. Finally, the entire system was subjected to system testing.

### 9.2 Unit Testing

#### 9.2.1 GIT modules

Test case	Expected Results	Actual Results	Conclusion
Location,Bio, Company information not updated in GIT profile	Field added with empty string	Field added with empty string	Success
Repos from GIT	Field added with repo links and languages used in	Field added with repo links and languages used in	Success

## Social DaTa Aggregator and Locator of Knowledge

	those projects.	those projects.	
Followers from GIT	Field added with follower url's	Field added with follower url's	Success
Location,Bio, Company information not updated in GIT profile but API is returning as Null	Field added with empty string	Field added with values as NULL	Failure

### 9.2.2 LinkedIn modules

Test case	Expected Results	Actual Results	Conclusion
Expected field not present in LinkedIn data. For e.g. "projects" not present	Empty string added	Empty string added	Success
Projects from LinkedIn	Project names are added to project_names and the duration, description and	Project names are added to project_names and the duration, description and	Success

## Social DaTa Aggregator and Locator of Knowledge

	team members are added to project_desc	team members are added to project_desc	
Classify skills from LinkedIn as technical, non_technical and languages	All skills are correctly categorized	C, Algorithms do not come under technical and languages	Failure

**Table 9.2**

### 9.2.3 Facebook Modules

Test case	Expected Results	Actual Results	Conclusion
Subject provides permission to access likes, groups and statuses.	The likes, groups and statuses are added to the respective array	The likes, groups and statuses are added to the respective array	Success
The subject does not provide permission to some of the fields in fb say birthday or email.	The respective fields in the fb dictionary will be blank with no data obtained from the API call	The respective fields in the fb dictionary will be blank with no data obtained from the API call	Success

## Social DaTa Aggregator and Locator of Knowledge

The subject provides permission to view the data about the photos in which the subject is tagged in	This information will be stored in the photos array in the fb dictionary	Sometimes, the API call fails and Facebook fails to send back data resulting in an empty photos array	Failure
---	--	---	---------

**Table 9.3**

### 9.2.4 Controller

Test case	Expected Results	Actual Results	Conclusion
Create five processes and assign records to each of them. Print process ID	Five different process IDs are printed	Five different process IDs are printed	Success
Pull five records with process_begun = false when there is at least one such record	Processing should begin	Processing begins	Success
Assign the record to specific social	Only the correct modules are called	Only the correct modules are called	Success

## **Social DaTa Aggregator and Locator of Knowledge**

network modules based on the links submitted			
--	--	--	--

**Table 9.4**

## Social DaTa Aggregator and Locator of Knowledge

### 9.2.5 Query Feature

Test case	Expected Results	Actual Results	Conclusion
Query: Does X know Java?	Yes: Java	Yes	Success
Query: Where does X work?	Work history of X: CISCO, CCBD	Work history of X: CISCO, CCBD	Success
Query: Did X study in RVCE?	NO. Education History of x: PESIT, RVPUC, Innisfree House School	NO. Education History of x: PESIT, RVPUC, Innisfree House School	Success
Query: Has X watched 27 dresses?	Yes	yes	Success
Query: Has X watched Harry Potter?	Yes	This query cannot be understood	Failure

**Table 9.5**

## Social DaTa Aggregator and Locator of Knowledge

### 9.2.6 Subject Functionality

Test case	Expected Results	Actual Results	Conclusion
Inserts fb and linkedin links	Successfully inserted	Successfully inserted	Success
Old subject returns for new event	Old links displayed	Old links displayed	Success
Old subject returned and submits only facebook link but git and linkedin data previously populated	Event creator can view only facebook data	Event creator can view only facebook data	Success

**Table 9.6**

### 9.2.7 User

#### 9.2.7.1 Log-in functionality

Test case	Expected Results	Actual Results	Conclusion
Right username,	Error displayed:	Error displayed:	Success



## Social DaTa Aggregator and Locator of Knowledge

Wrong password	password incorrect	password incorrect	
Wrong username, Right password	Error displayed: username incorrect	Error displayed: username incorrect	Success
Right username,Right password	Successfully logged in	Successfully logged in	Success

**Table 9.7**

### 9.2.7.2 User Functionality

Test case	Expected Results	Actual Results	Conclusion
Create a new event	A new Event is created. Reflected in the Events tab.	A new Event is created. Reflected in the Events tab.	Success
Create a new template	A new Template is created. Reflected in the Templates Tab	A new Template is created. Reflected in the Templates Tab	Success
Select Subject	Profile of the selected subject displayed	Profile of the selected subject displayed	Success

## Social DaTa Aggregator and Locator of Knowledge

Apply Template	Template applied, auto refresh happens	Template applied but auto refresh does not happen. User needs to click on the subject's name to see the template in effect	Failure
Accept/Reject subject for an event	An email is sent with default message	An email is sent with default message	Success

**Table 9.8**

## 9.3 Integration Testing

### 9.3.1 Git and LinkedIn

Test case	Expected Results	Actual Results	Conclusion
Check if languages from LinkedIn overlaps with Git	No duplicate values are stored	No duplicate values are stored	Success
Check if name from LinkedIn overlaps with Git	If no name in Git, store only LinkedIn. Else if names are	Blank string results in name getting added twice	Failure

## Social DaTa Aggregator and Locator of Knowledge

	different, store both		
Check if project names and descriptions are added correctly	Project name to description maps in the same order	Project name to description maps in the same order	Success

**Table 9.9**

### 9.3.2 Git, Linkedin and Facebook

Test case	Expected Results	Actual Results	Conclusion
Check if name from Facebook overlaps with Git	If no name is stored in Git, then store the name of Facebook. Else if names are different, store both.	Facebook name is stored in the nlq data	Success
Check if the education names overlap with LinkedIn	If no education history, then store Facebook education history. Else store the names and	Some of the names are not detected as being similar, hence there is a repetition. For	Failure.

## Social DaTa Aggregator and Locator of Knowledge

	description of the ones which are different.	example, PESIT and P.E.S Institute of Technology are tagged as different names even though one is an acronym of the other	
Check if the work history overlaps with the LinkedIn data.	No duplicates stored	No duplicates stored	Success
Check for name overlapping between Git and linkedIn data and Fb data	No overlap occurs and duplicates are not stored	No overlap occurs and duplicates are not stored	Success

**Table 9.9**

### 9.3.3 Controller, Git, Linkedin and Facebook

Test case	Expected Results	Actual Results	Conclusion
Creating multiple processes	Multiple processes are created	Different processes IDs are printed	Success

## Social DaTa Aggregator and Locator of Knowledge

		indicating multiple processes	
Rollback of flags if processing functions fail	nlg_done and process_begun are set to false	nlg_done and process_begun are set to false	Success
Each process runs asynchronously	Main process does not wait for child to return before spawning next process	Main process does not wait for child to return before spawning next process	Success

**Table 9.10**

### 9.3.4 Front end and MongoDB

Test case	Expected Results	Actual Results	Conclusion
Getting the events list from database and displaying it	All events from the database are displayed in the dropdown menu	All the events from the database are displayed in the dropdown menu	Success
Auto filling subject information if subject has already	The details of the subject are auto filled from the	The details of the subject are auto filled from the	Success

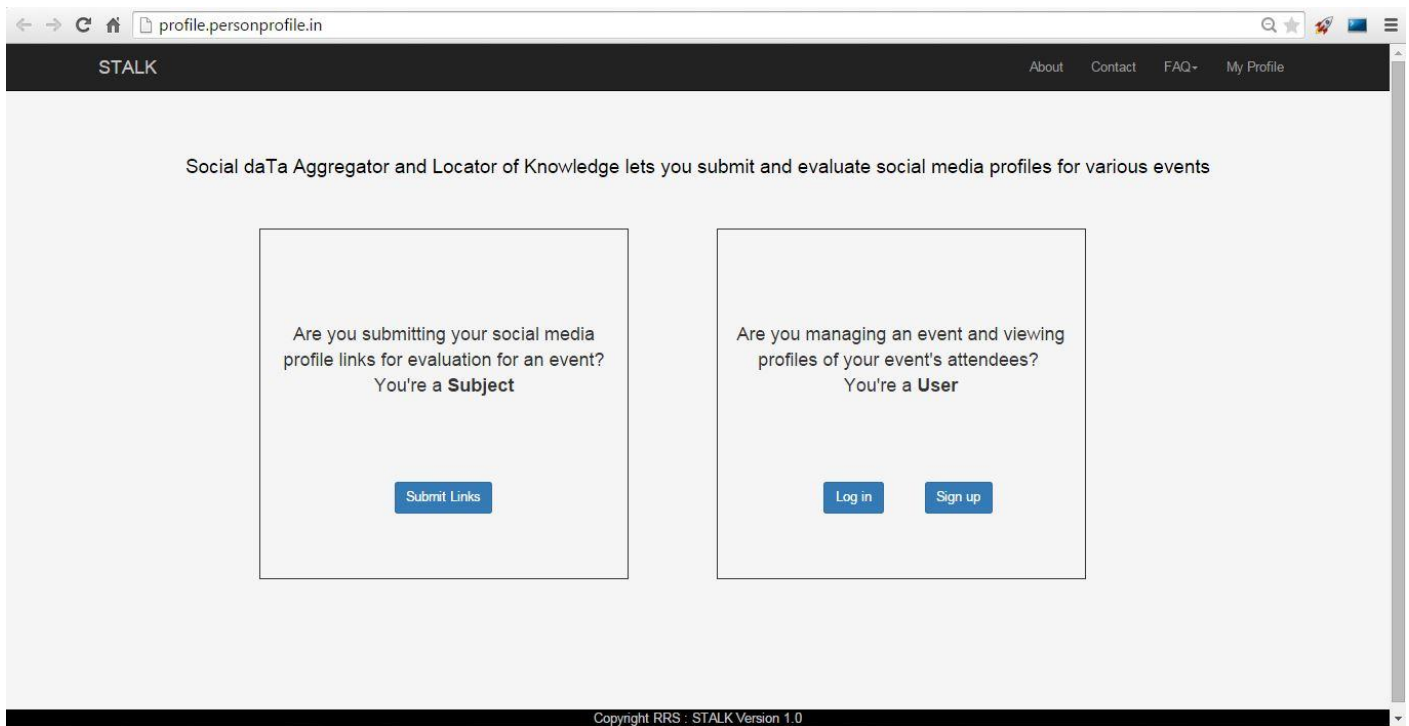
## Social DaTa Aggregator and Locator of Knowledge

registered before	database.	database	
Creation of template/event	Display the new event/ template in the event/template dropdown after creation	Display the new event/ template in the event / template dropdown after creation	Success
Creation of new user	After creation, user must be able to log in and access the functionalities of the user.	The user is able to login and access the functionalities provided to the user.	Success

**Table 9.11**

## 10. STORY BOARD

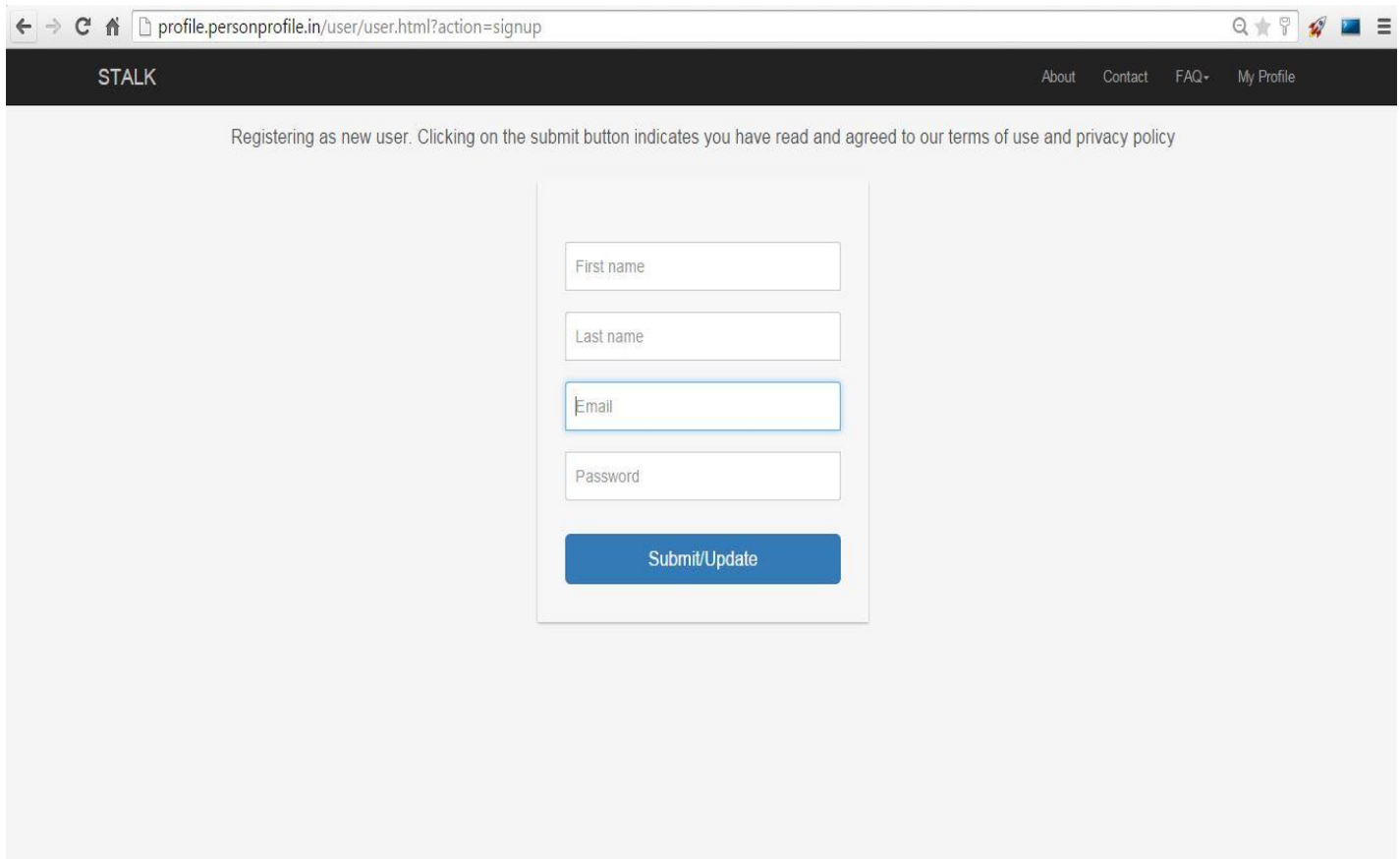
The application begins at the index page



**Fig 10.1**

## Social DaTa Aggregator and Locator of Knowledge

A user wishes to sign up for this application



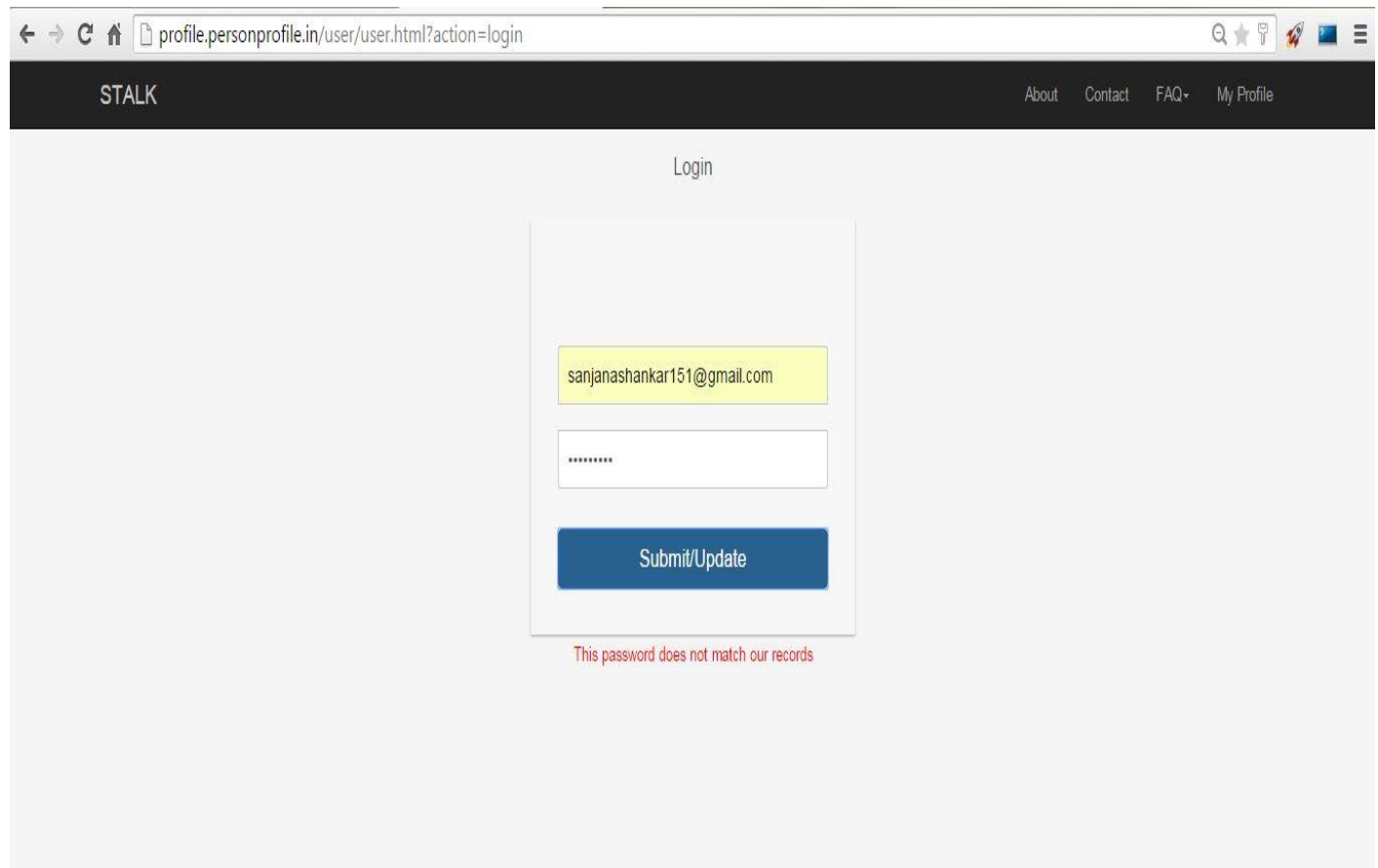
The screenshot shows a web browser window with the address bar displaying `profile.personprofile.in/user/user.html?action=signup`. The page has a dark header with the word "STALK" on the left and navigation links "About", "Contact", "FAQ-", and "My Profile" on the right. Below the header, a message states: "Registering as new user. Clicking on the submit button indicates you have read and agreed to our terms of use and privacy policy". The registration form is centered and contains four input fields: "First name", "Last name", "Email", and "Password". Below these fields is a blue button labeled "Submit/Update".

**Fig 10.2**



## Social DaTa Aggregator and Locator of Knowledge

Once he or she has signed up, they can log in to their profiles



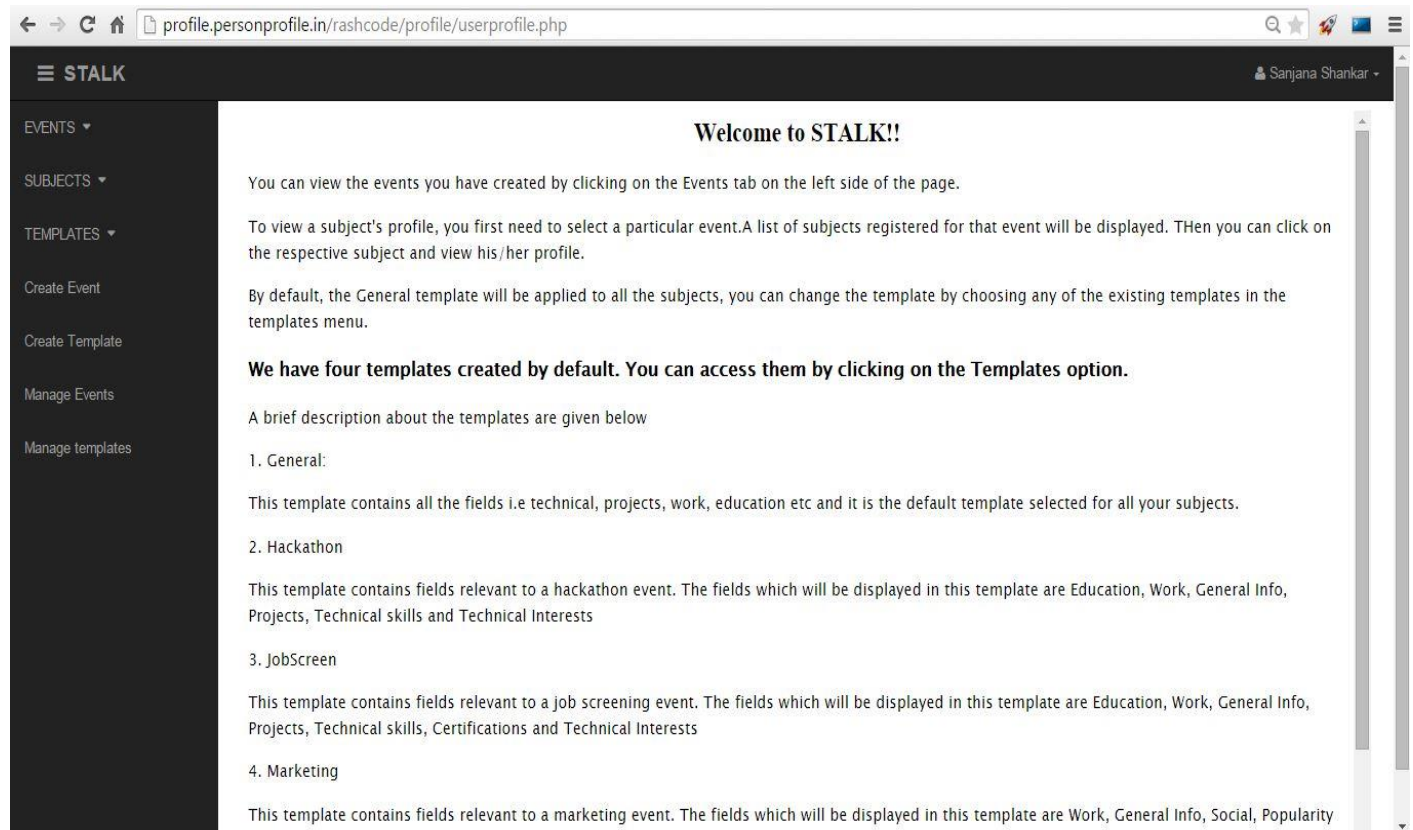
The screenshot shows a web browser window with the address bar displaying `profile.personprofile.in/user/user.html?action=login`. The page has a dark header with the word "STALK" on the left and links for "About", "Contact", "FAQ", and "My Profile" on the right. The main content area is titled "Login" and contains a central form. The form has a yellow input field with the email address "sanjanashankar151@gmail.com", a white input field with masked characters "\*\*\*\*\*", and a blue "Submit/Update" button. Below the form, a red error message states "This password does not match our records".

**Fig 10.3**

The system handles the cases of invalid password or email.

## Social DaTa Aggregator and Locator of Knowledge

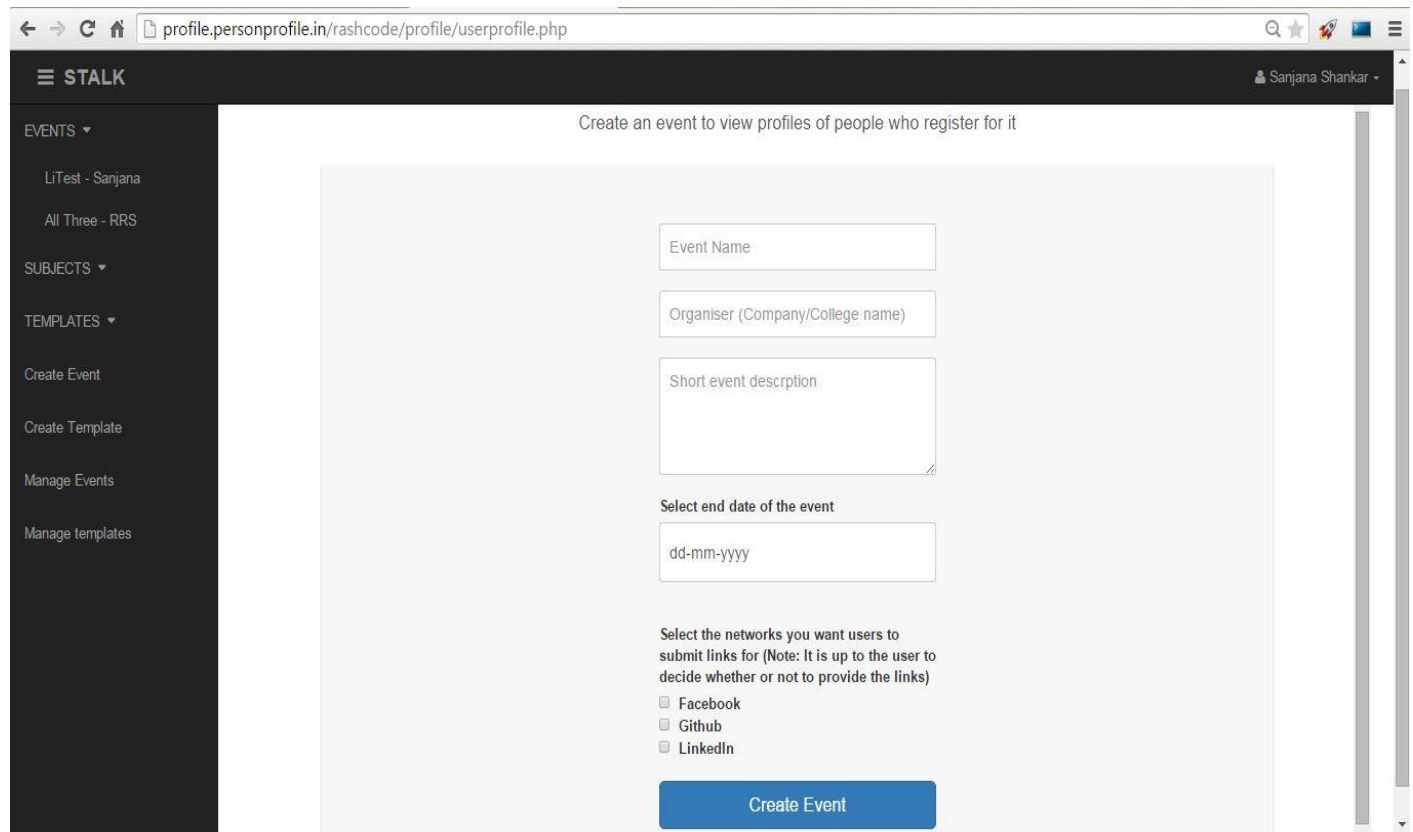
On successful login, the user can see his account page.



**Fig 10.4**

## Social DaTa Aggregator and Locator of Knowledge

He can then create an event

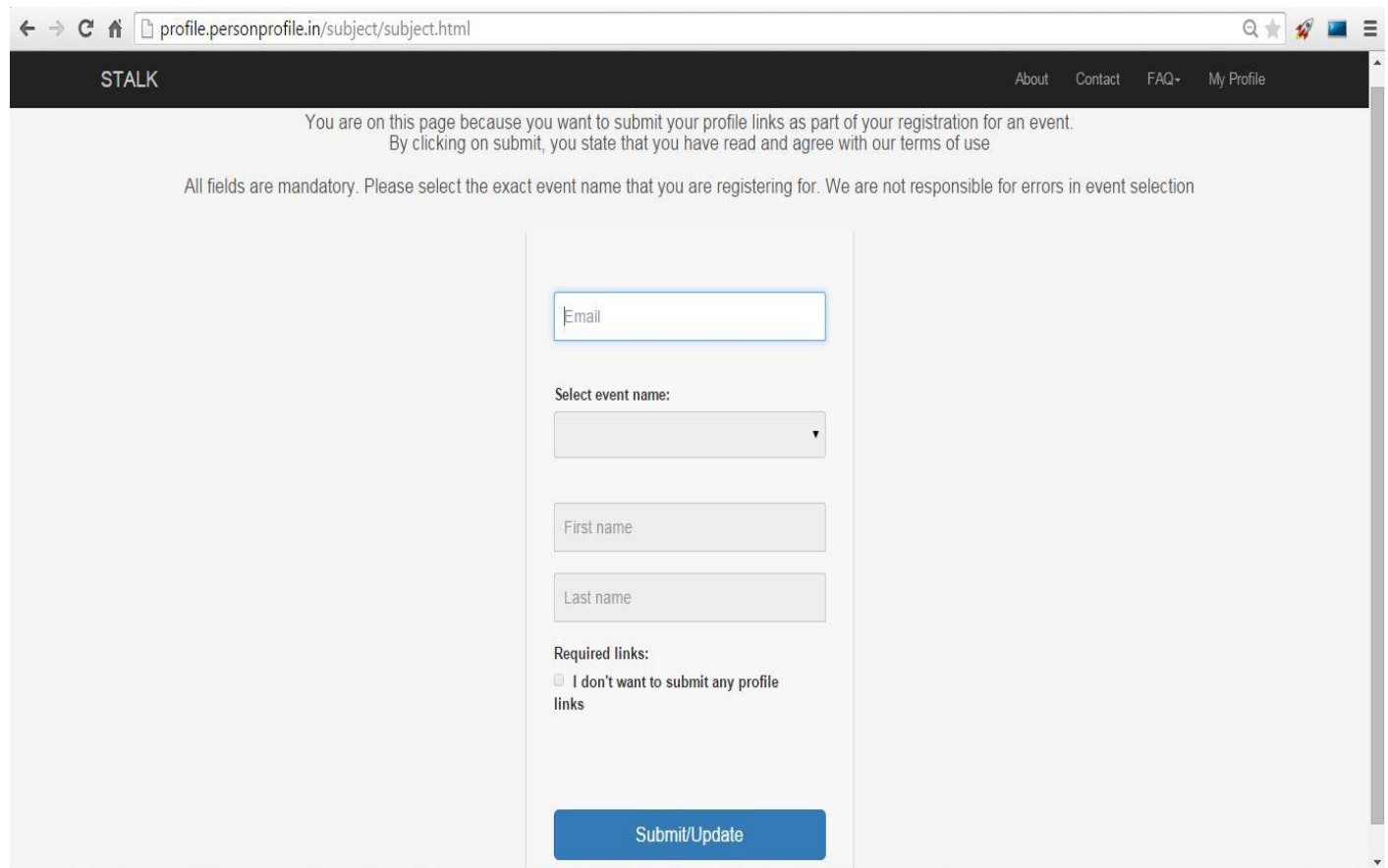


The screenshot shows a web browser window with the URL `profile.personprofile.in/rashcode/profile/userprofile.php`. The page has a dark sidebar on the left with the title "STALK" and a user profile "Sanjana Shankar". The sidebar contains a menu with "EVENTS" (expanded) and "SUBJECTS". Under "EVENTS", there are links for "LITest - Sanjana", "All Three - RRS", "Create Event", "Create Template", "Manage Events", and "Manage templates". The main content area is titled "Create an event to view profiles of people who register for it". It contains a form with the following fields: "Event Name", "Organiser (Company/College name)", "Short event description" (a text area), "Select end date of the event" (a date input field with the placeholder "dd-mm-yyyy"), and a section for selecting networks with checkboxes for "Facebook", "Github", and "LinkedIn". A blue "Create Event" button is at the bottom of the form.

**Fig 10.5**

## Social DaTa Aggregator and Locator of Knowledge

Once this is done, the subject can register for the event



The screenshot shows a web browser window with the URL `profile.personprofile.in/subject/subject.html`. The page has a dark header with the word "STALK" on the left and navigation links "About", "Contact", "FAQ-", and "My Profile" on the right. The main content area contains the following text:

You are on this page because you want to submit your profile links as part of your registration for an event.  
By clicking on submit, you state that you have read and agree with our terms of use.

All fields are mandatory. Please select the exact event name that you are registering for. We are not responsible for errors in event selection

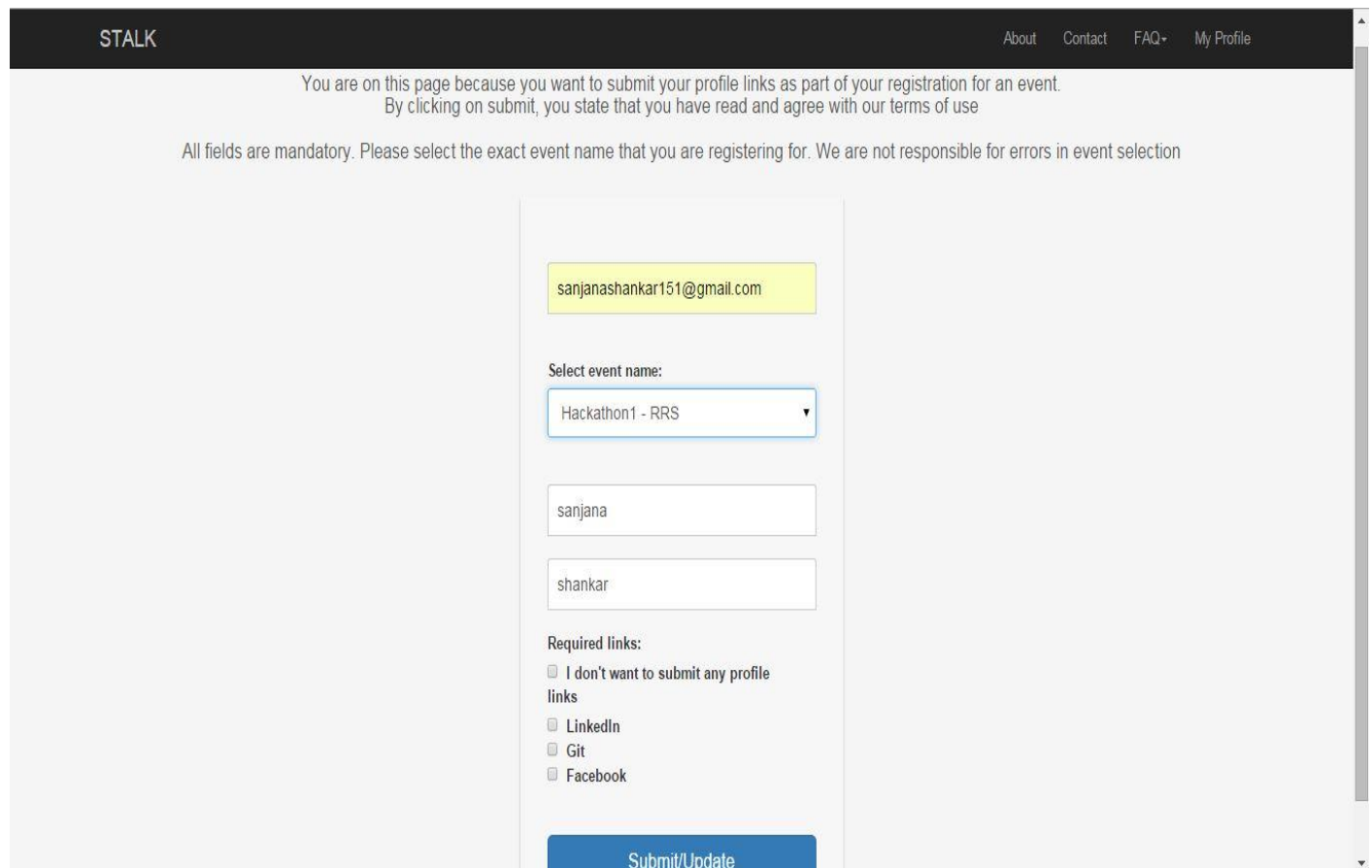
The registration form includes the following fields and elements:

- An "Email" input field.
- A "Select event name:" label followed by a dropdown menu.
- A "First name" input field.
- A "Last name" input field.
- A "Required links:" section with a checkbox labeled "I don't want to submit any profile links".
- A blue "Submit/Update" button at the bottom.

**Fig 10.6**

## Social DaTa Aggregator and Locator of Knowledge

To do so, the subject enters their email ID and selects an event to display the links required by the event.

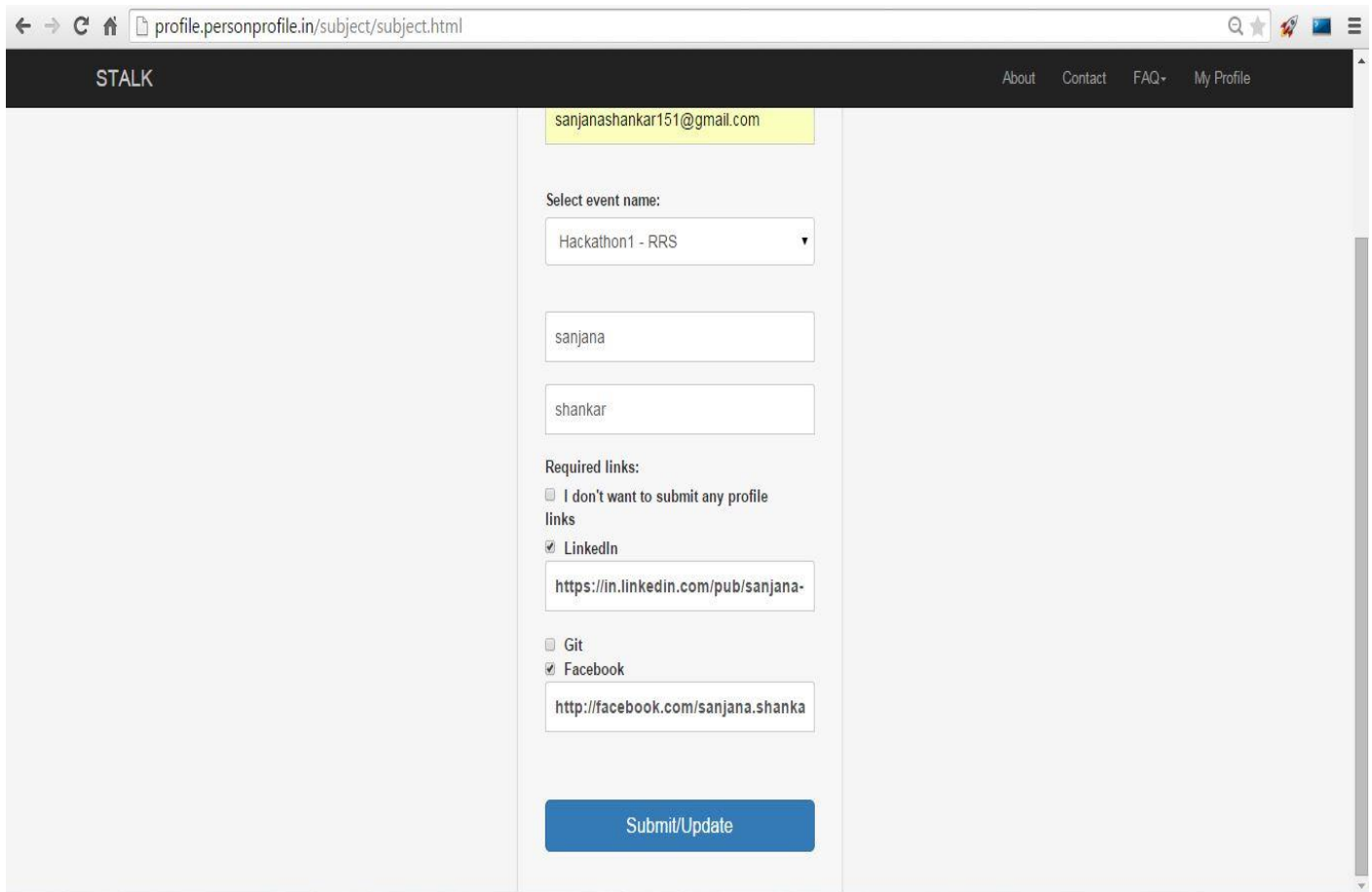


The screenshot shows a web application interface for 'STALK'. At the top, there is a navigation bar with links for 'About', 'Contact', 'FAQ+', and 'My Profile'. Below the navigation bar, a message states: 'You are on this page because you want to submit your profile links as part of your registration for an event. By clicking on submit, you state that you have read and agree with our terms of use'. A warning message follows: 'All fields are mandatory. Please select the exact event name that you are registering for. We are not responsible for errors in event selection'. The main form area contains several input fields: an email field with 'sanjanashankar151@gmail.com', a dropdown menu for 'Select event name:' with 'Hackathon1 - RRS' selected, a text field for 'sanjana', and another text field for 'shankar'. Below these fields, there is a section for 'Required links:' with three checkboxes: 'I don't want to submit any profile links', 'LinkedIn', 'Git', and 'Facebook'. At the bottom of the form is a blue button labeled 'Submit/Update'.

**Fig 10.7**

## Social DaTa Aggregator and Locator of Knowledge

The subject's details get automatically filled if present in the database already.

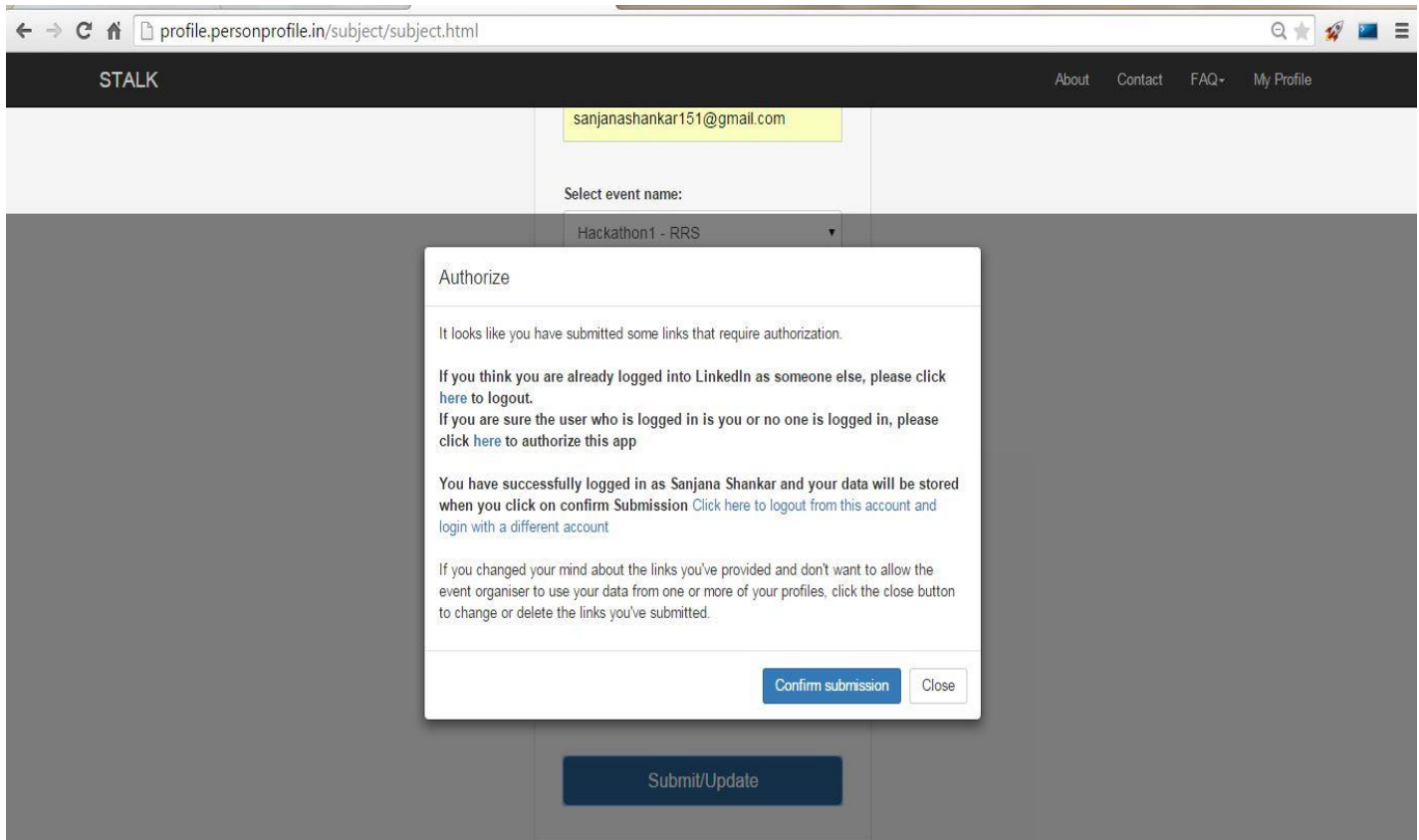


The screenshot shows a web browser window with the address bar displaying `profile.personprofile.in/subject/subject.html`. The page has a dark header with the word "STALK" on the left and navigation links "About", "Contact", "FAQ+", and "My Profile" on the right. The main content area is a form for editing a profile. At the top, the email address "sanjanashankar151@gmail.com" is displayed in a yellow box. Below this, there is a "Select event name:" dropdown menu currently showing "Hackathon1 - RRS". Underneath the dropdown are two text input fields: the first contains "sanjana" and the second contains "shankar". Further down, under the heading "Required links:", there are two checkboxes. The first checkbox, "I don't want to submit any profile links", is unchecked. The second checkbox, "LinkedIn", is checked, and it is followed by a text input field containing the URL "https://in.linkedin.com/pub/sanjana-". Below this, the "Git" checkbox is unchecked and the "Facebook" checkbox is checked, followed by a text input field containing the URL "http://facebook.com/sanjana.shanka". At the bottom of the form is a blue button labeled "Submit/Update".

**Fig 10.8**

## Social DaTa Aggregator and Locator of Knowledge

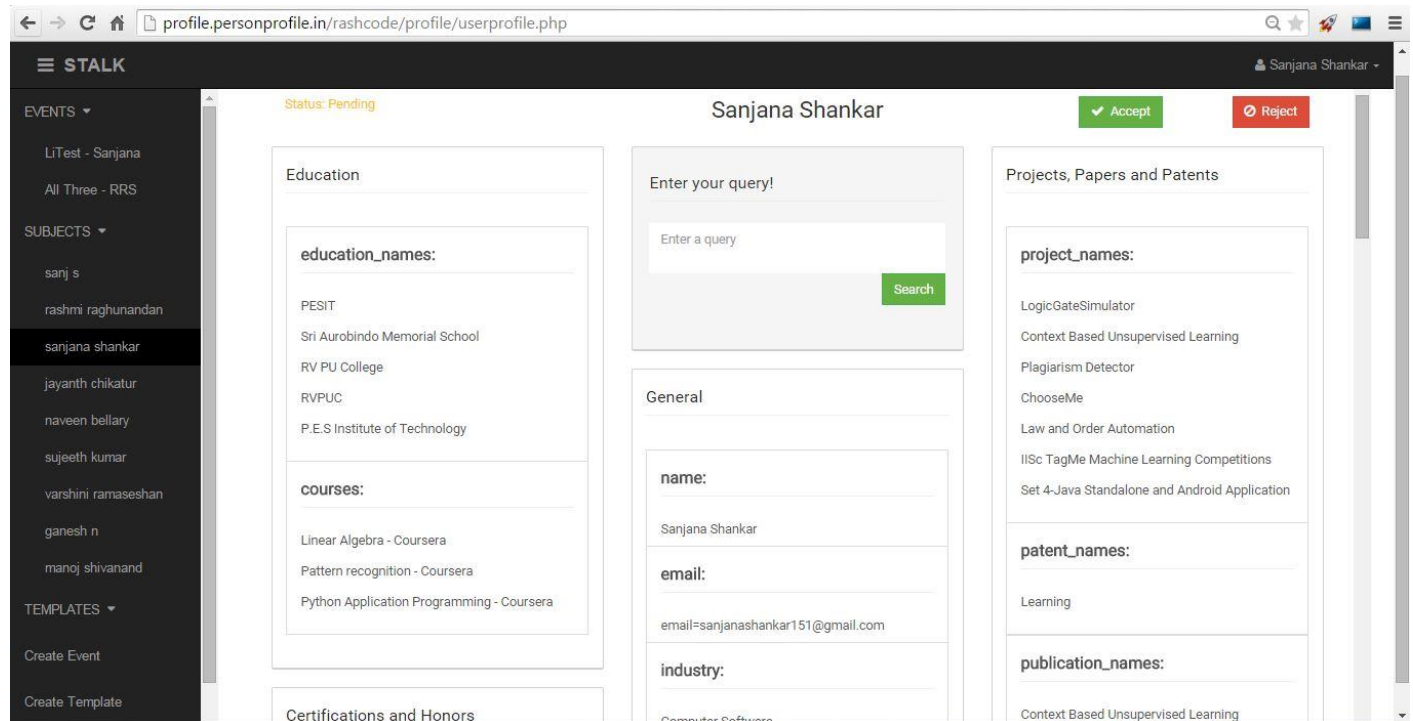
If authentication is needed, the modal window opens.



**Fig 10.9**

## Social DaTa Aggregator and Locator of Knowledge

Once authentication is completed and the subject has registered successfully, his record is processed. Once processing is complete, the user can view the consolidated profile.



The screenshot displays a web application interface for a user profile. The browser address bar shows the URL: `profile.personprofile.in/rashcode/profile/userprofile.php`. The user is logged in as Sanjana Shankar, with a status of **Pending**. The interface includes a sidebar with navigation options like **EVENTS**, **SUBJECTS**, and **TEMPLATES**. The main content area is divided into several sections:

- Education:** Lists education names (PESIT, Sri Aurobindo Memorial School, RV PU College, RVPUC, P.E.S Institute of Technology) and courses (Linear Algebra - Coursera, Pattern recognition - Coursera, Python Application Programming - Coursera).
- General:** Contains fields for name (Sanjana Shankar), email (email=sanjanashankar151@gmail.com), and industry (Computer Software).
- Projects, Papers and Patents:** Lists project names (LogicGateSimulator, Context Based Unsupervised Learning, Plagiarism Detector, ChooseMe, Law and Order Automation, IISc TagMe Machine Learning Competitions, Set 4-Java Standalone and Android Application) and patent names (Learning).

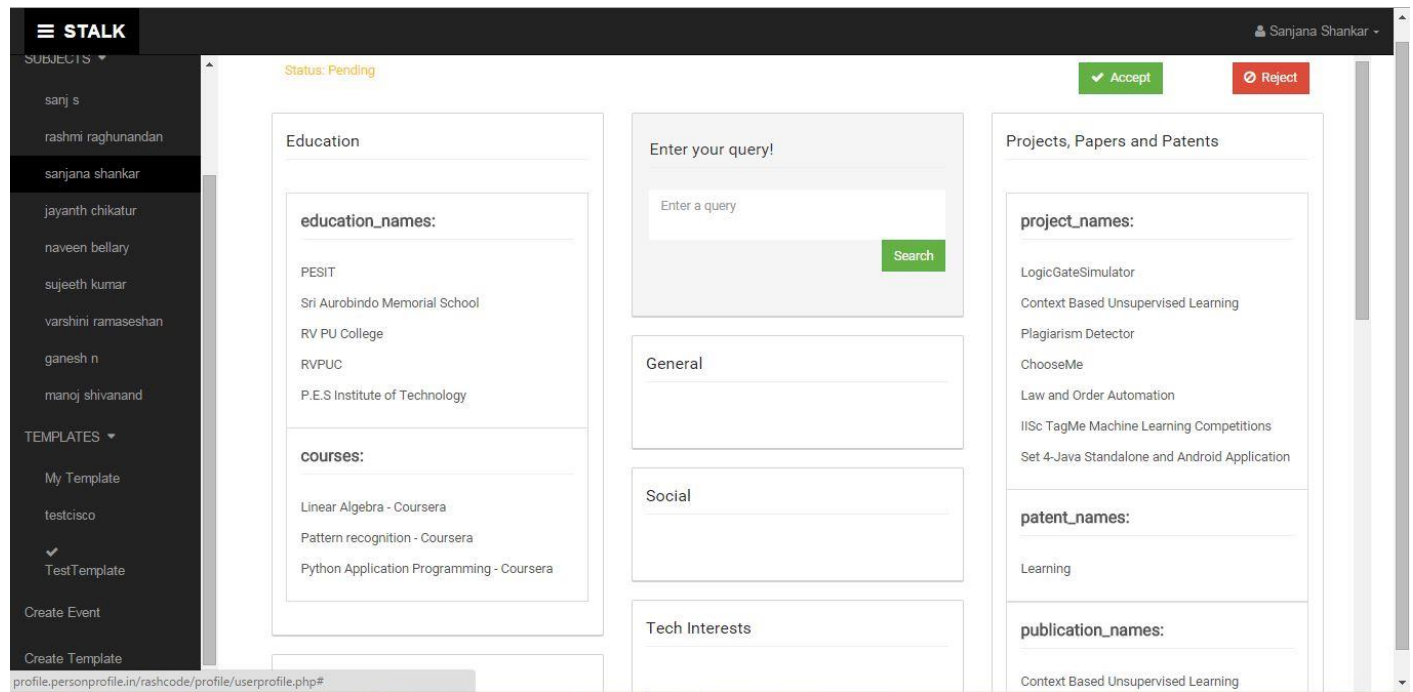
A search bar is available in the center, and buttons for **Accept** and **Reject** are located at the top right of the profile section.

**Fig 10.10**



## Social DaTa Aggregator and Locator of Knowledge

A template can also be applied.



The screenshot displays the 'STALK' section of the application for user Sanjana Shankar. The interface includes a sidebar with a list of subjects and templates. The main content area shows a 'Status: Pending' notification and several input fields for user information.

**STALK** Sanjana Shankar

Subjects: sanj s, rashmi raghunandan, **sanjana shankar**, jayanth chikatur, naveen bellary, sujeeth kumar, varshini ramaseshan, ganesh n, manoj shivanand

Templates: My Template, testcisco, **TestTemplate**, Create Event, Create Template

Status: Pending

**Education**

education\_names:

PESIT  
Sri Aurobindo Memorial School  
RV PU College  
RVPUC  
P.E.S Institute of Technology

**courses:**

Linear Algebra - Coursera  
Pattern recognition - Coursera  
Python Application Programming - Coursera

Enter your query!

Enter a query **Search**

**General**

**Social**

**Tech Interests**

**Projects, Papers and Patents**

project\_names:

LogicGateSimulator  
Context Based Unsupervised Learning  
Plagiarism Detector  
ChooseMe  
Law and Order Automation  
IISc TagMe Machine Learning Competitions  
Set 4-Java Standalone and Android Application

patent\_names:

Learning

publication\_names:

Context Based Unsupervised Learning

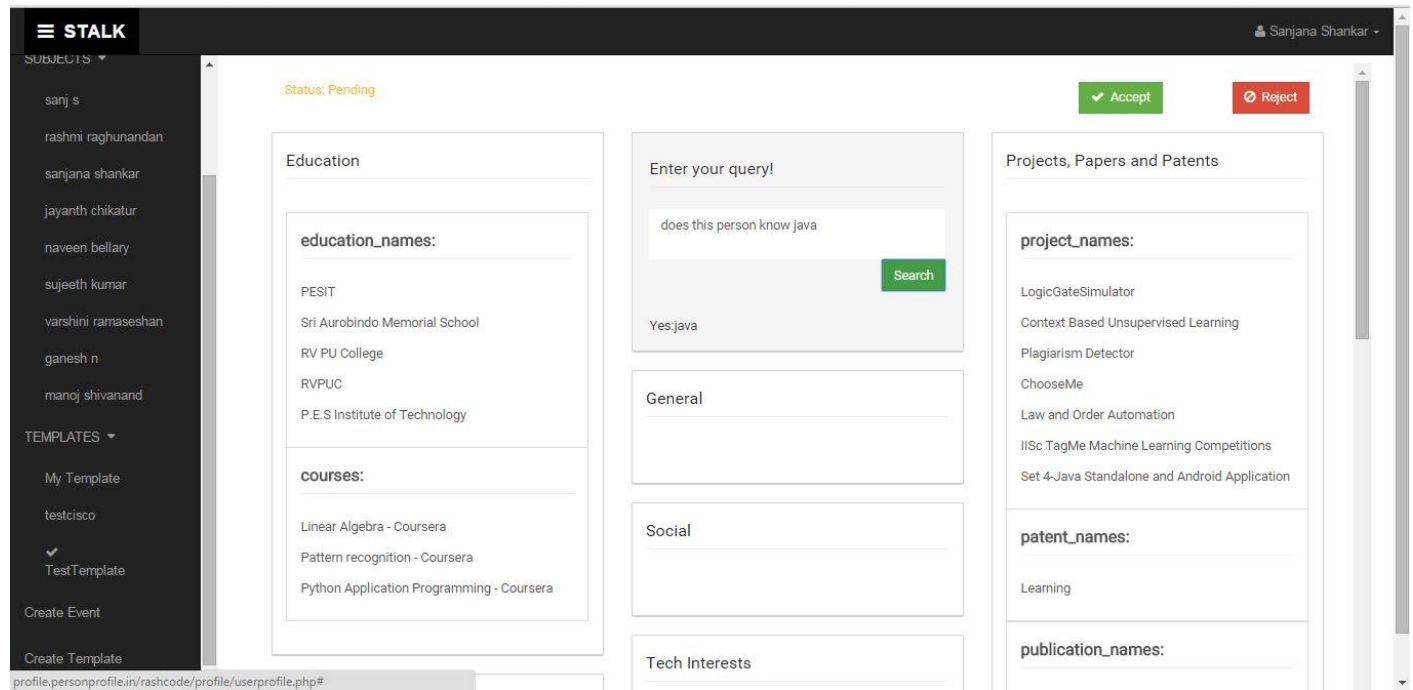
**Accept** **Reject**

profile.personprofile.in/rashcode/profile/userprofile.php#

**Fig 10.11**

## Social DaTa Aggregator and Locator of Knowledge

The user can query for information.

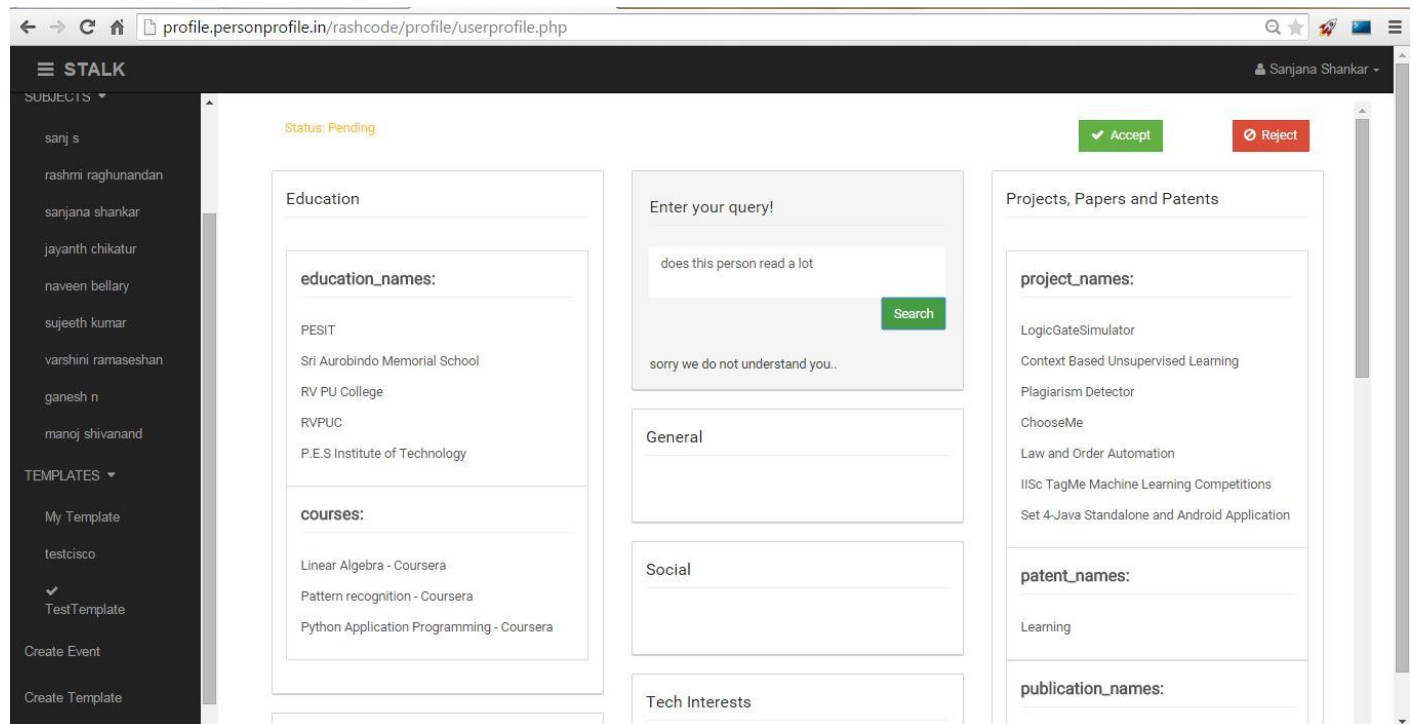


The screenshot displays a web application interface for a social data aggregator. On the left, a dark sidebar contains a 'STALK' menu with a list of users and a 'TEMPLATES' section. The main content area is divided into several sections: a top status bar indicating 'Status: Pending', an 'Education' section listing institutions like PESIT and Sri Aurobindo Memorial School, a 'courses' section listing Coursera courses, a search bar with the query 'does this person know java' and a 'Search' button, a 'General' section, a 'Social' section, a 'Tech Interests' section, and a 'Projects, Papers and Patents' section listing various projects and patents. At the top right, there are 'Accept' and 'Reject' buttons. The URL at the bottom is 'profile.personprofile.in/rashcode/profile/userprofile.php#'. The user's name 'Sanjana Shankar' is visible in the top right corner.

**Fig 10.12**

## Social DaTa Aggregator and Locator of Knowledge

Unfortunately, not all queries can be understood by the system.



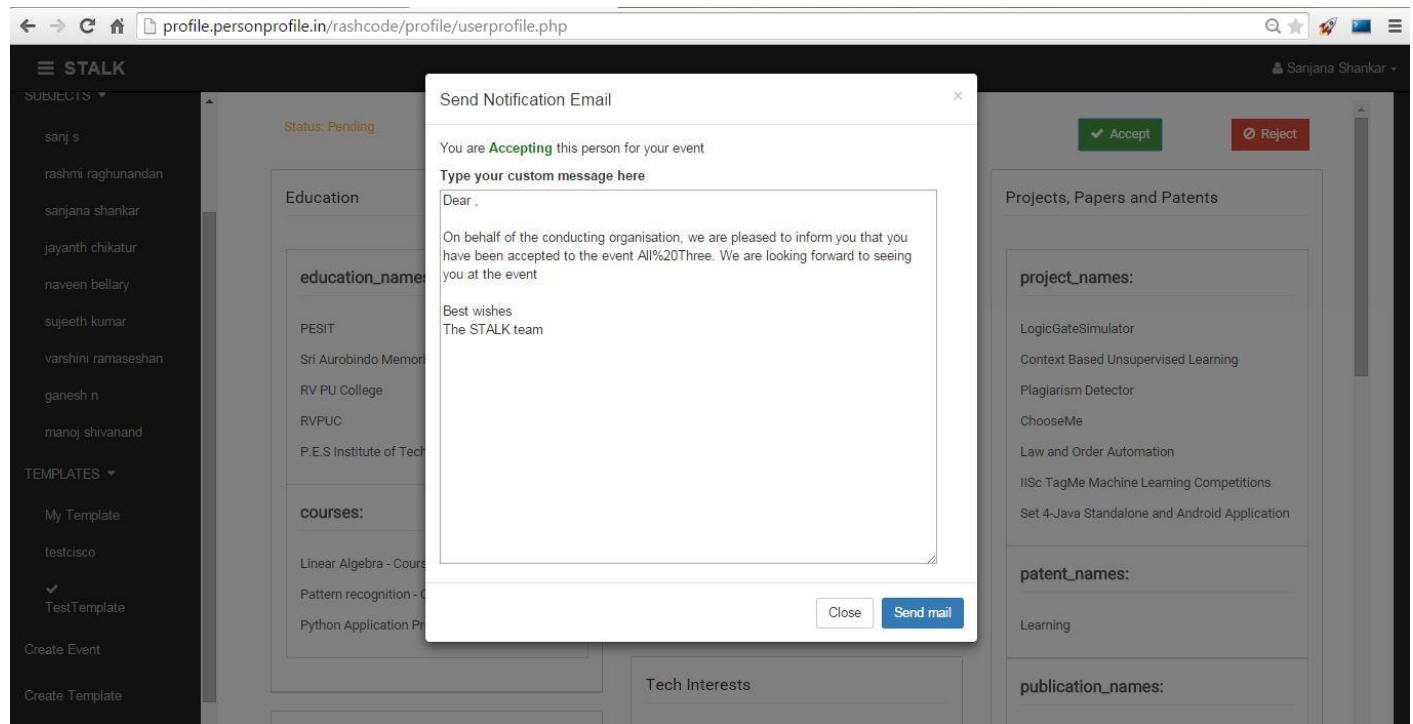
The screenshot shows a web application interface for a user profile. The browser address bar displays `profile.personprofile.in/rashcode/profile/userprofile.php`. The user is logged in as Sanjana Shankar. The interface is divided into several sections:

- Left Sidebar:** Contains a 'STALK' header, a 'SUBJECTS' dropdown menu with a list of names (sanj s, rashmi raghunandan, sanjana shankar, jayanth chikatur, naveen bellary, sujeeth kumar, varshini ramaseshan, ganesh n, manoj shivanand), and a 'TEMPLATES' dropdown menu with options like 'My Template', 'testcisco', 'TestTemplate', 'Create Event', and 'Create Template'.
- Main Content Area:**
  - Status:** Pending
  - Education:** A section with a list of education names: PESIT, Sri Aurobindo Memorial School, RV PU College, RVPUC, and P.E.S Institute of Technology.
  - Courses:** A section with a list of courses: Linear Algebra - Coursera, Pattern recognition - Coursera, and Python Application Programming - Coursera.
  - Search:** A search bar with the query 'does this person read a lot' and a 'Search' button. Below the search bar, a message says 'sorry we do not understand you..'. There are 'Accept' and 'Reject' buttons at the top right of the search area.
  - General:** A section with a text input field.
  - Social:** A section with a text input field.
  - Tech Interests:** A section with a text input field.
  - Projects, Papers and Patents:** A section with a list of project names: LogicGateSimulator, Context Based Unsupervised Learning, Plagiarism Detector, ChooseMe, Law and Order Automation, IISc TagMe Machine Learning Competitions, and Set 4-Java Standalone and Android Application.
  - patent\_names:** A section with a list of patent names: Learning.
  - publication\_names:** A section with a text input field.

**Fig 10.13**

## Social DaTa Aggregator and Locator of Knowledge

Finally, the user can choose to accept or reject the subject for his event.



**Fig 10.14**

## **11. CONCLUSION**

The product- Social daTa Aggregator and Locator of Knowledge can be used a proof of concept as well as a usable product. It is a first of its kind products that consolidates and provides information about a person from his various social media profiles. It provides information on the go with its query feature, similar to and an extension of the facebook graph search.

This project was an interesting and challenging one as it involved us to deep dive into different domains. We understood the nature of social networks, learnt the usage and importance of API's, experimented with different natural language processing algorithms.

We used growing technologies like Data Mining and MongoDB to solve a problem that is critical in today's internet era.

The usage of our product in the three use cases- shortlisting for hackathons, pre screening for job selections and market analysis of a product acceptance have shown how this product has diverse scope in solving some day to day problems.

## **12. FUTURE SCOPE**

Every product has room for improvement. Some areas where STALK can be improved are:

1. Adding more social media profiles for analysis
2. Allowing multiple users to host and manage 1 event
3. Offering suggestions when our query system fails to understand the query.
4. Improving and providing more features to the analysis of a profile.
5. Improved UI