Search systems using Apache Solr: Travel Destination Search

Team Name: badk
Masters' in Computer Science
University at Buffalo, The State University of New York
CSE 535 Information Retrieval (Fall 2013)

Anirudh Karwa
akarwa@buffalo.edu

Babu Prasad Dhandapani
babupras@buffalo.edu

Deeshen Shah
deeshend@buffalo.edu

Kalpesh Kagresha
kkagresh@buffalo.edu

ABSTRACT
The purpose of this project is to build a full search system using Apache Solr to solve a real life problem. The aim is to have teams apply the different concepts learnt in the class to some interesting problems while learning how to setup, configure and use Apache Solr.

General Terms
Query Processing, Indexing, Tokenization, Parsing, Data retrieval, Information Retrieval, Web Application

Keywords
Apache Solr, Apache Lucene, Google Maps, Ranking models, Wikivoyage

1. INTRODUCTION
The aim of this project is to suggest travel destinations to users based on their preferences. Thus, building an search system that takes input from user and giving results. We have limited our attention to English wikivoyage for this purpose. Each page has similar subsections describing activities to do, stats about climate, getting there etc.
We have indexed around 5000 different pages. We allow the users to specify multiple parameters, at least as ANDed conditions. We also provide a filter facility allowing users to apply / remove preferences or automatically discover new facets within their results to drill down on.
We have develop a web application where in user can specify his/her search parameters (which state, which city, what kind place he want to visit, etc.) and based on it we will provide the ways in which user can reach to its destination. Along with that we will show pictures of the destination and any reviews that are valid for the result. The architecture of our application is shown in figure 2. The subsequent sections explain about the various modules of our architecture.

2. DATABASE MODULE
The main goal of the database module is to take the English wikivoyage documents/dumps as input, parse, tokenize, index and then store them into Solr. The Figure 1 shows the flowchart will explain the module in detail.

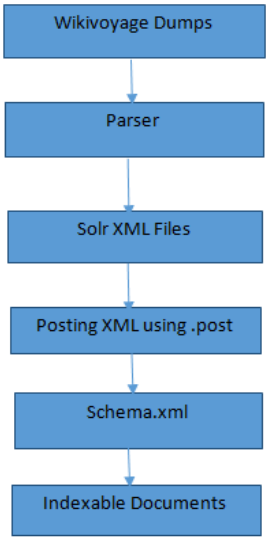


Figure 1:Data Flow

The custom Parser will first remove the wiki markup and then will add the xml markup to the wikivoyage content. This will convert the dump xml files into Solr XML files. The xml markup will be added as per the requirement to index the data.The Solr XML will be the input the Posting XML using .post. The Solr xml files are given as input to schema.xml. The structure of schema.xml and the Solr xml documents will be similar and predefined in the style the documents need to be indexed. Inbuilt analyzers and tokenizers provided by Solr will be implemented during the indexing time. The same analyzers and tokenizer rules will be applied in the query processing module. These analyzers and tokenizers will be defined in the schema.xml file.

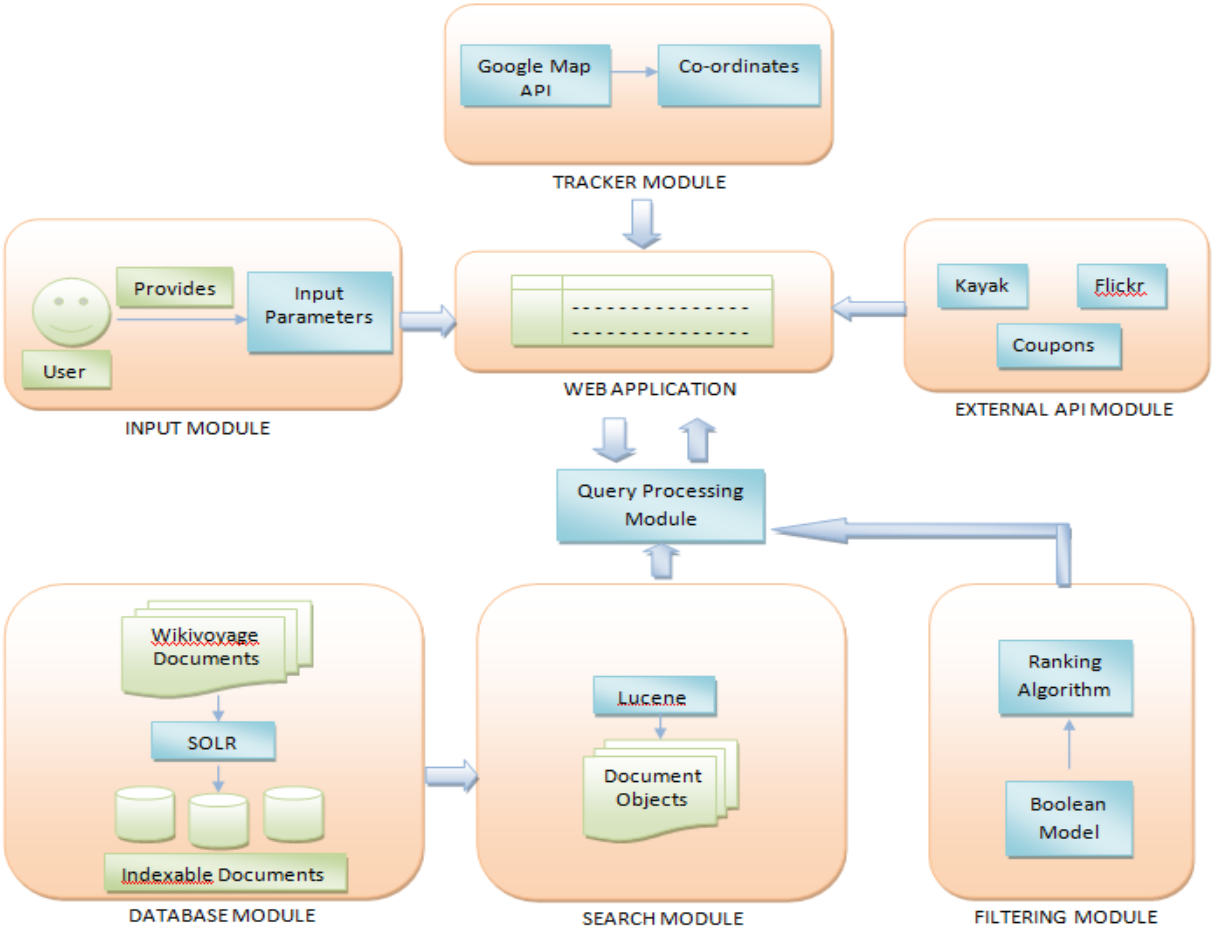


Figure 2: System Architecture

3. PARSER MODULE

For Travel Based Destination Search, we have used dump of approximately 49k Wikivoyage pages belonging to different cities across the globe. For testing purpose, only 5k pages of cities belonging to United States are extracted and parsed. Every wikivoyage page starts with a default section describing the city followed by some sections namely Districts, Understand, Get In, Get Around, See, Do, Learn, Buy, Eat, Drink, Sleep etc. These sections describe the city in terms of places, history, climate, transportation, activities and events. User might be interested to spend his/her vacation in one of the popular places/ doing activities irrespective of the destination. We have observed that all pages have similar sections and

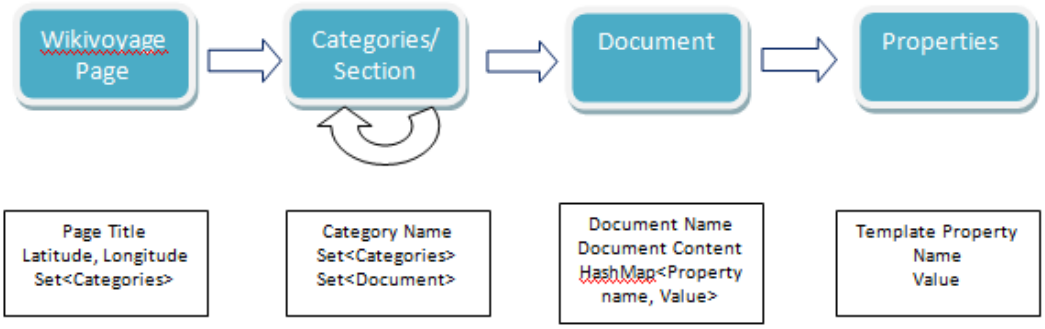


Figure 3:Parser Schema and Methodology

activities. This led us to group all pages/cities according to sections mentioned above i.e. section named "Eat" will include parsed data from all 5k pages. Further on every section consists of sub-section describing more about city and listing places to visit. Therefore every section consists of list of sections and documents.

The above figure 3 represents the schema design for parsing the wikivoyage pages. For parsing the wikivoyage page about the Buffalo city, we consider "Do" as a category/section having the subsection as "Festival and Events". This subsection consists of documents named "Buffalo Niagara Film Festival" and "Buffalo Pride Festival". These documents have various properties such as name content, address, contact, latitude and longitude. Another example is from "Eat" section consisting of "Local chains" as subsection and list of restaurants as documents. In order to parse this information, we check for section and template markup tags in the documents.

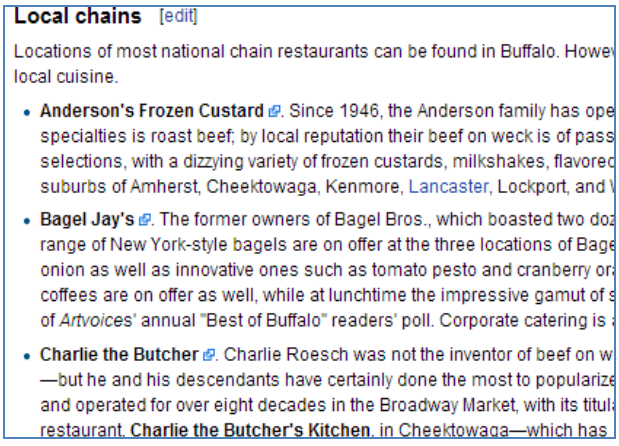

Do [edit]

Festivals and Events [edit]

Buffalo's calendar of annual festivals, parades and events is huge and growing. Eth role, though a diversity of events of all kinds is enjoyed by citizens. Naturally, the lion offerings in winter as well.

The festivals and events listed in this section take place at multiple venues city-

- **Buffalo Niagara Film Festival**. An international film festival for and by filmmake Jr., Lou Ferrigno, and Buffalo native William Fichtner, the Buffalo Niagara Film Fe independent feature-length and short films. In addition to film screenings, semir where festival sponsors can promote their wares. edit
- **Buffalo Pride Festival**. The gay rights movement emerged later in conservative Buffalo Pride Festival has been helping LGBT Buffalonians and their straight alli The Buffalo Pride Festival is multifaceted and multi-venue: it kicks off with a flag- footrace through the streets of downtown, picks up intensity in Allentown with the performances, and street activism, and culminates with the Pride Festival itself i each year with a beach party at **Woodlawn Beach State Park** in Hamburg. edit


Local chains [edit]

Locations of most national chain restaurants can be found in Buffalo. Howe local cuisine.

- **Anderson's Frozen Custard**. Since 1946, the Anderson family has ope specialties is roast beef; by local reputation their beef on weck is of pass selections, with a dizzying variety of frozen custards, milkshakes, flavored suburbs of Amherst, Cheektowaga, Kenmore, Lancaster, Lockport, and
- **Bagel Jay's**. The former owners of Bagel Bros., which boasted two doz range of New York-style bagels are on offer at the three locations of Bage onion as well as innovative ones such as tomato pesto and cranberry or coffees are on offer as well, while at lunchtime the impressive gamut of s of *Artvoices*' annual "Best of Buffalo" readers' poll. Corporate catering is
- **Charlie the Butcher**. Charlie Roesch was not the inventor of beef on w —but he and his descendants have certainly done the most to popularize and operated for over eight decades in the Broadway Market, with its titul restaurant, **Charlie the Butcher's Kitchen**, in Cheektowaga—which has

Figure 4: Snippet from Wikivoyage Page (Title: Buffalo)

4. SOLR CONFIGURATION DETAILS

Every Wikivoyage page contains all the activities or information regarding that particular title (town/city/state). We have divided the single wikivoyage page into multiple xml files each xml file being a prospective activity for vacation planning. For example, restaurant.xml file will contain the information of all the restaurants for all the wikivoyage pages. In restaurant.xml, each restaurant will have a unique id associated with it and will be indexed as a separate document. If we have two restaurants in city Akron, then we will have two different documents one for restaurant "Beaus Grille" and one for "Bricco".

<doc>
<field name="id">32308</field>
<field name="title">Akron</field>
<field name="main">eat</field>
<field name="section">mid-range</field>
<field name="name">Beaus Grille</field>
<field name="content">Banquet Rooms, Live Music, Bar, Seafood</field>
<field name="phone">+1-330 -865-5577</field>
<field name="address">3180 West Market Street</field>
<field name="url">http://www.beausgrille.com/</field>
<field name="weight">1</field>
<field name="clicks">0</field>
</doc>
<doc>
<field name="id">32309</field>
<field name="title">Akron</field>
<field name="main">eat</field>
<field name="section">mid-range</field>
<field name="name">Bricco</field>
<field name="content">Serves a reasonably priced Italian-based menu and specialized in pizzas and pastas ranging from $9-18. On Monday and Tuesday nights they sell wine at state minimum retail prices. There is also a relatively lively bar scene for weeknights.</field>
<field name="phone">+1-330 -475-1600</field>
<field name="fax">+1-(330)-475-1604</field>
<field name="address">1 West Exchange</field>
<field name="hours">Hours: Mon-Thu: 11 - Midnight, Fri: 11 - 01:00, Sat: 16 - 01:00, Closed most Sundays</field>
<field name="url">http://www.briccoakron.com/</field>
<field name="weight">1</field>
<field name="clicks">0</field>
</doc>

So using this strategy we finally had 188 xml files and these files had overall 105488 documents to be indexed.

4.1 Important fields (field types) and their uses.

1. **ID** (string): This is the unique document identifier.
2. **Title** (lowercase): This is the wikivoyage page title.
3. **State** (lowercase): Indicates that the title belongs to which state in the country.
4. **Country** (lowercase): Stores the country name for the title.
5. **Main** (text_general): Contains all the activities for the document.
6. **Section** (text_general): Contain the activity for the document.
7. **Sub-section** (text_general): Indicates within which sub-section the document belongs.
8. **Content** (text_ws): Contains all the details about a particular place/activity.
9. **Name** (lowercase): Indicates the name of the place.
10. **Weights** (float): It is used for the custom ranking purpose.
11. **Clicks** (int): Used for the custom ranking purpose. Indicates how many hits the page has received.
12. **Store** (location): Contains the location coordinates (latitude and longitude) for the place.

4.2 Various Copy Fields used:

1. **Searchfield** (text_general): It copies the data from main, section, sub-section and sub-sub-section fields. A particular activity can be present in any of the four fields. So to search the pages within a city (level 2), we will use this field.

2. **Autosuggestactivity** (text_general): This field is explicitly used for the auto suggest functionality for the activity text box. It currently copies data from main field only. We have implemented it as copy field so if required in near future, we can easily add other fields for auto suggest.

3. **Autosuggeststate** (lowercase): This field is explicitly used for the auto suggest functionality for the city/state text box. It copies data from title and state fields.

4. **Displaycontent** (text_general): This field is used for to store detailed level information for a particular place. This information will be displayed when the user will click on a particular place (level 3). For example, if the user clicks on a place “National Museum”, then all the details for the museum like its address, timings, etc. will be displayed.

4.3 Dynamic Fields:

1. ***extra**(text_general): Any fields which ends with extra will be copied here. This data will generally be displayed using the displaycontent field.

Note:*Dynamic fields are more useful when we have to index data during run time. As we are not adding data dynamically, it is not much useful from the current project implementation point of view.*

4.4 Various Field Types:

1. String: Standard class “solr.StrField" is used.

2. Text_general: Standard class “solr.TextField" is used.
   Tokenizer class used: solr.StandardTokenizerFactory
   Filter class used: solr.StopFilterFactory and solr.LowerCaseFilterFactory

3. Int: Standard class “solr.TrieIntField” is used.

4. Float: Standard class “solr.TrieFloatField” is used.

5. Location: Standard class="solr.LatLonType” is used. This is explicitly used for geospatial search.

6. Lowercase: Standard class “class="solr.TextField” is used
   Tokenizer class used: solr.KeywordTokenizerFactory
   Filter class used: solr.LowerCaseFilterFactory

   This is used for explicitly when we want to store the value with lower case only and without applying any other tokenizer rule. When we store title as “New York”, then “New” and “York” should not be stored as individual tokens. Also if user is querying with lowercase as new york, we should be able to fetch the result.

7. Text_ws:Standard class “solr.TextField" is used.
   Tokenizer class used: solr.StandardTokenizerFactory
   Filter class used: solr.StopFilterFactory, solr.LowerCaseFilterFactory and solr.PorterStemFilterFactory

4.5 Different handlers and components used:

Custom Handlers:

1. **Suggeststate**: This handler is explicitly used for auto suggesting keywords for the city/state text box on the user interface. It fetches value from the autosuggeststate copy field.

2. **Suggestactivity**: This handler is explicitly used for auto suggesting keywords for the activity text box on the user interface. It fetches value from the autosuggestactivity copy field.

3. **Spellstate**: This handler is explicitly used for spell correction of keywords for the city/state text box on the user interface. It fetches value from the autosuggeststate copy field.

4. **Spellactivity**: This handler is explicitly used for spell correction of keywords for the activity text box on the user interface. It fetches value from the autosuggestactivity copy field.

`	Search Component Class	Search Component Name	Handler Class
Suggestactivity	solr.SpellCheckComponent	suggestactivity	org.apache.solr.handler.component.SearchHandler
Suggeststate	solr.SpellCheckComponent	suggeststate	org.apache.solr.handler.component.SearchHandler
Spellactivity	solr.SpellCheckComponent	spellcheckactivity	solr.SearchHandler
Spellstate	solr.SpellCheckComponent	spellcheckstate	solr.SearchHandler

Table1: Custom Handlers

4.6 Standard Handlers Used: Select, Update

5. Query Processing Module

When the user types in any preference or the state information, the user interface invokes the Query Processing module which connects to solr and retrieves the required information. Query Processing involves the following sub modules.

1. The input parameter is first validated through Spell Checker Module which will return the true or suggested results.
2. Next if the update weight flag is set, then weight which exists in the index for the items that are currently selected is incremented by a standard factor. Two different weights along with time and ID of the selected item are maintained for global relevance ranking and user preference.
    a. Global weight for the currently selected item is updated in the index of the specific field that is searched for the result.
    b. At the same time local user specific weight is updated in the local user cache which is identified with the user id. (In our case we have considered IP Address as the user ID)
    c. Also the time during which the item is viewed is stored in the local user cache for displaying most recently viewed items.
3. URL is constructed dynamically through the parameters fetched from the user input such as city name, preference name or state name and depending upon the page where the query processing is invoked. Also facets are added for categorizing the Solr response .
4. Query is executed and the response is parsed to a JSON parser to retrieve the details that are expected from the user interface and corresponding ID, weight, etc.
5. And if the grouping is needed by the user interface then data is grouped according to the category specified and returned to the UI.

Flow Diagram:

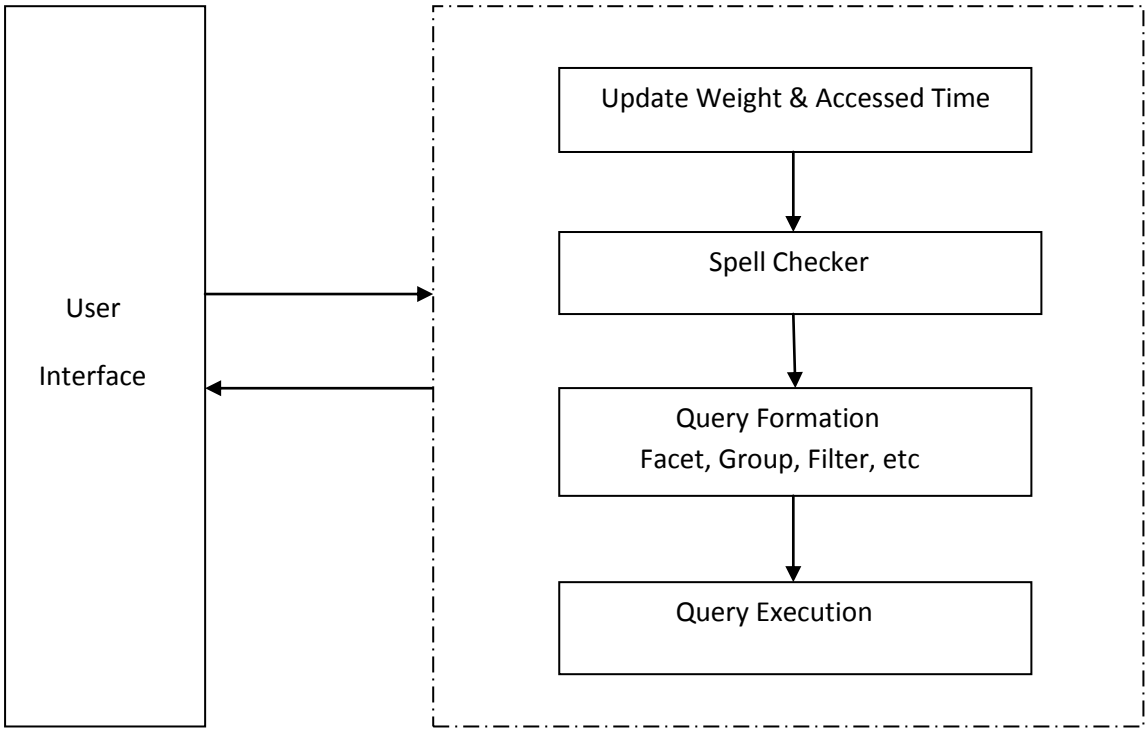


Figure 5: QP Flow

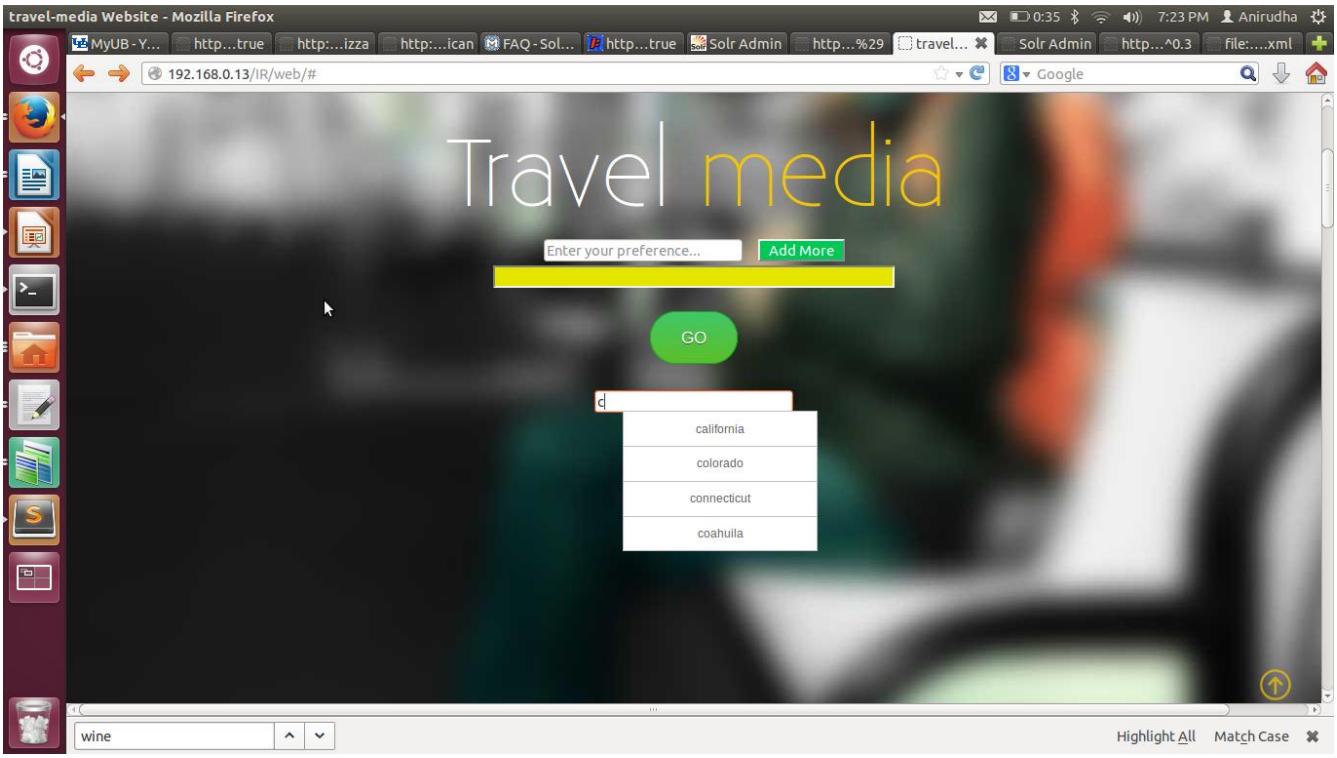
6 FEATURES

6.1 Auto Suggesstion

- Autosuggestion functionality makes the system more intelligent as it tries to guess the keywords the user is searching for.
- These keywords are guessed from the indexes of field autosuggeststate and autosuggestactivity for state textbox and activity text box respectively.
- We are showing the best 5 matching results by making the suggestion count to 5.
- We have enabled the onlyMorePopular parameter as true as the goal is to show the most popular travel locations/activities first.

We have defined two custom handlers for the auto-suggest functionality. More implementation details of this feature are included in the configuration section.

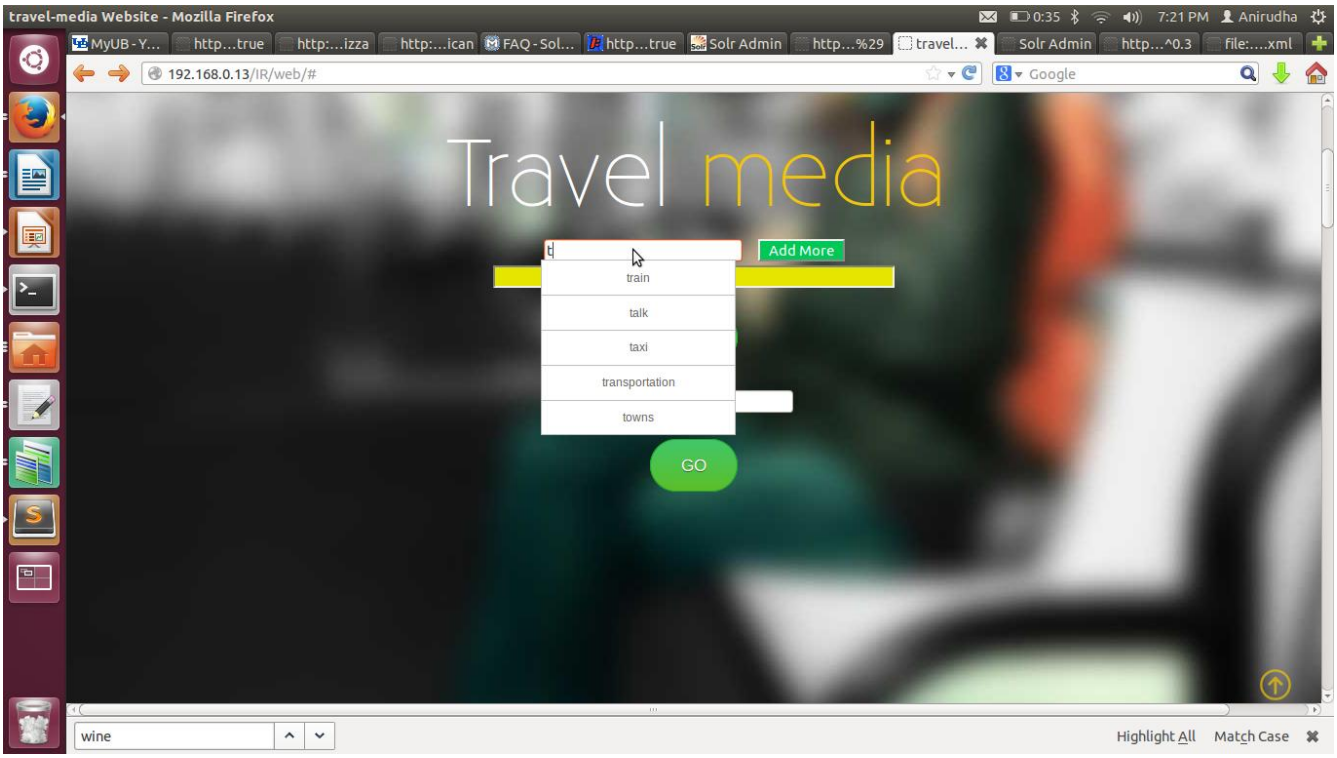
UI example for state text box:



When the user enters word “c” in the state text box, the system suggests california, colarado, connecticut and coahuila as the possible keywords the user is looking for.

UI example for activity text box:

When the user enters word “t” in the activity text box, the system suggests train, talk, taxi, transportation and towns as the possible keywords the user is looking for.



Spell Correction:

This feature checks for the misspelled keywords entered by the user. If user enters

- Correct spelled keyword: The system would fetch the results for the entered keyword.
- Misspelled keyword: The system would try to guess the correctly spelled word the user is looking for would display result for that keyword.
- Random word: Suggest the user to refine its search by providing a message. (Random word is a word that is not present in the index).

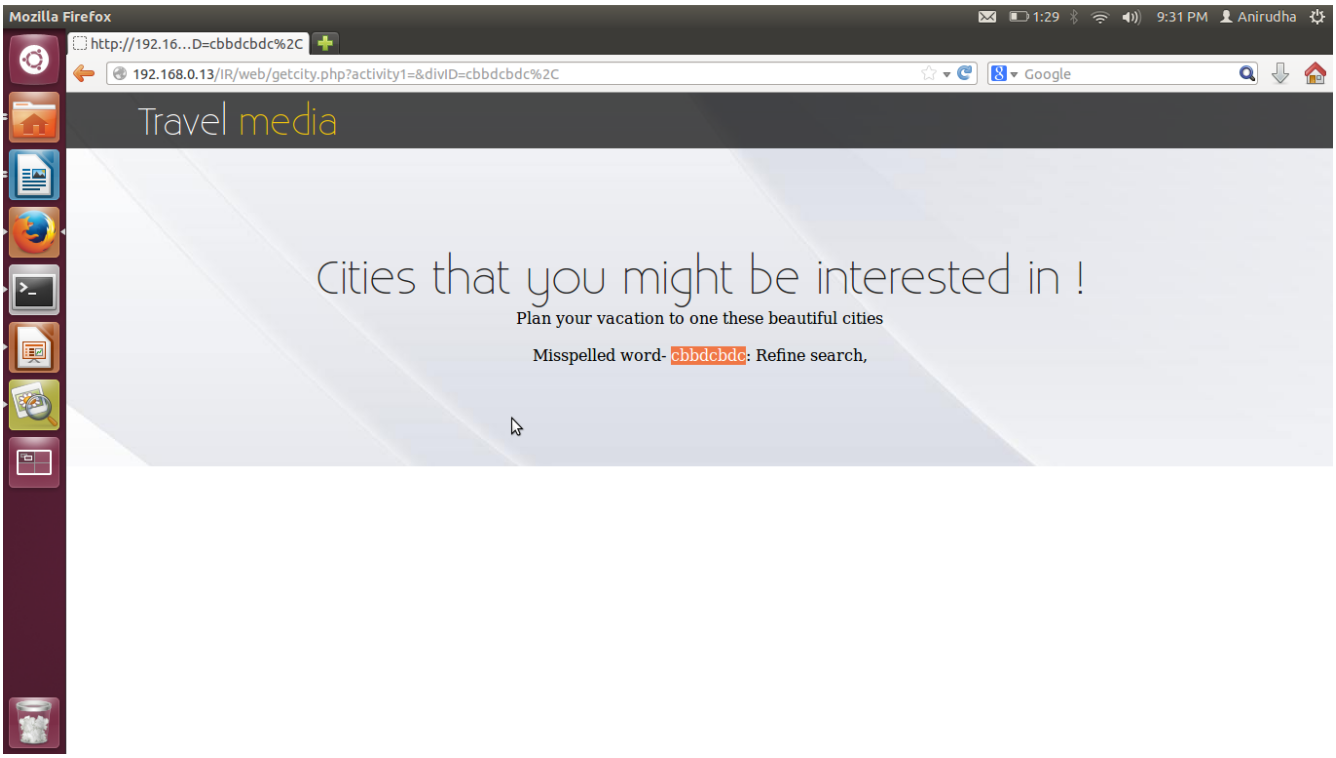
We have defined two custom handlers for the auto suggest functionality. More implementation details of this feature is included in the configuration section.

UI example for misspelled word:

When the user enters word “hikinh” in the activity text box, the system suggests hiking as the possible keyword and display results for it.

UI example for random word:

When the user enters word “cbbdcbdc” in the activity text box, the system suggests refining the search.



6.2 Drill down approach

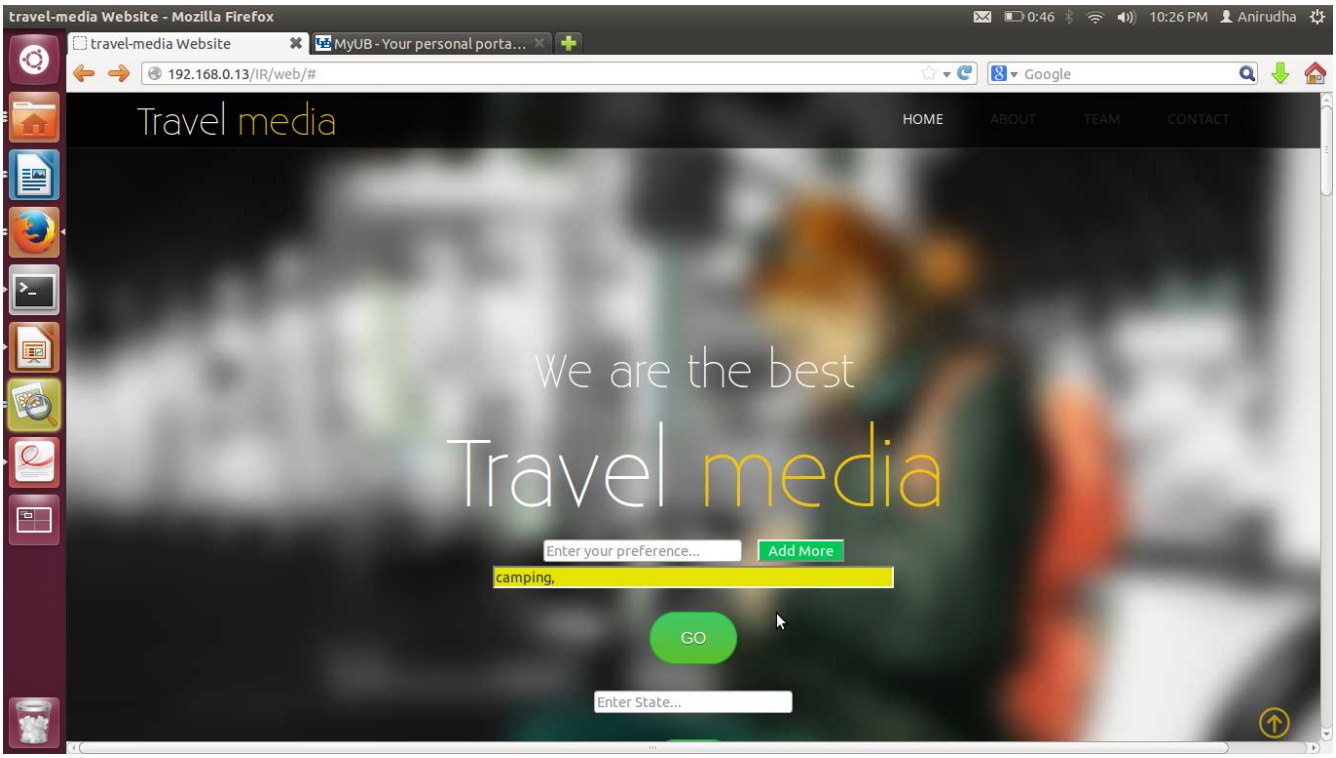
If the user in not sure where he/she wants to travel, then drill down approach is an effective way to help user narrow down his/her search.

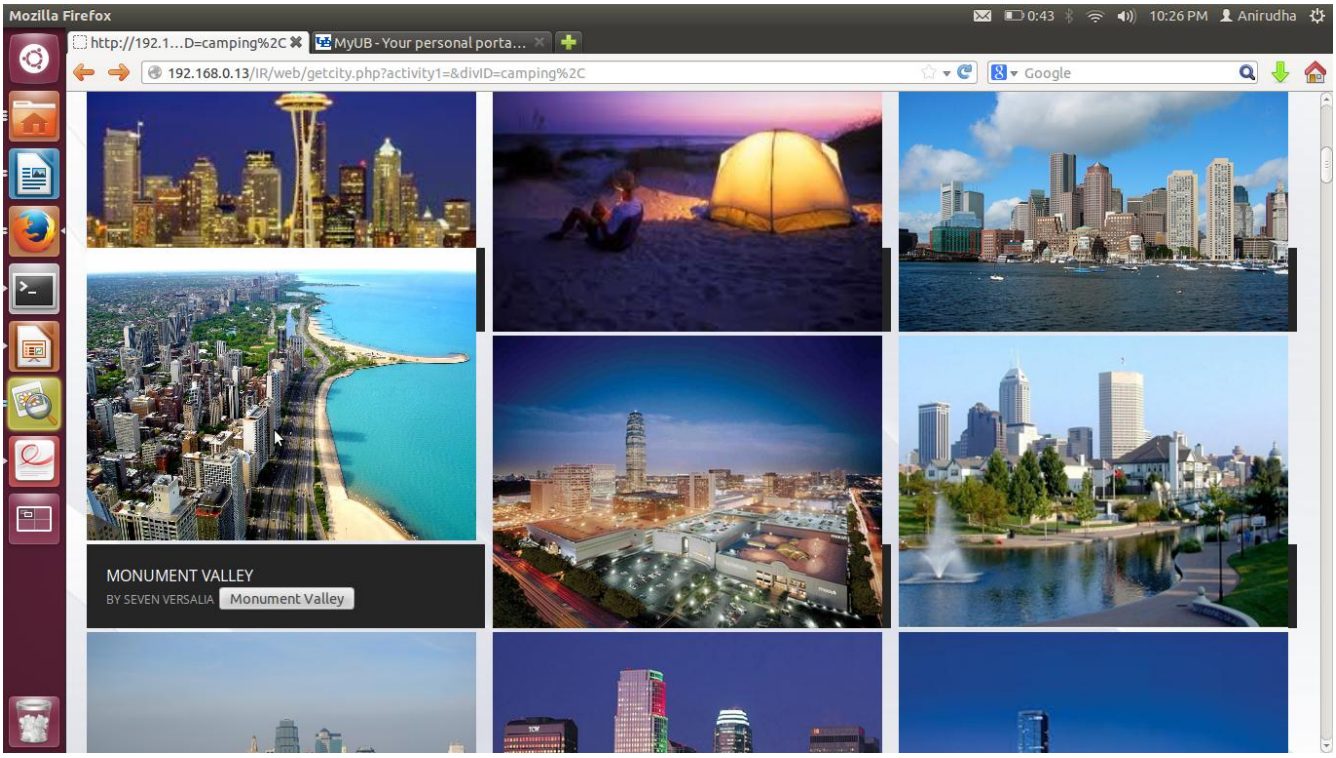
Appraoch 1:

1. Set of activities-->List of cities (Level 1)
2. Select a particular city from List of cities-->Display all the results for the selected city(Level 2)
3. Select a particular result/place → Display details or information about the selected place(Level 3)

Example:

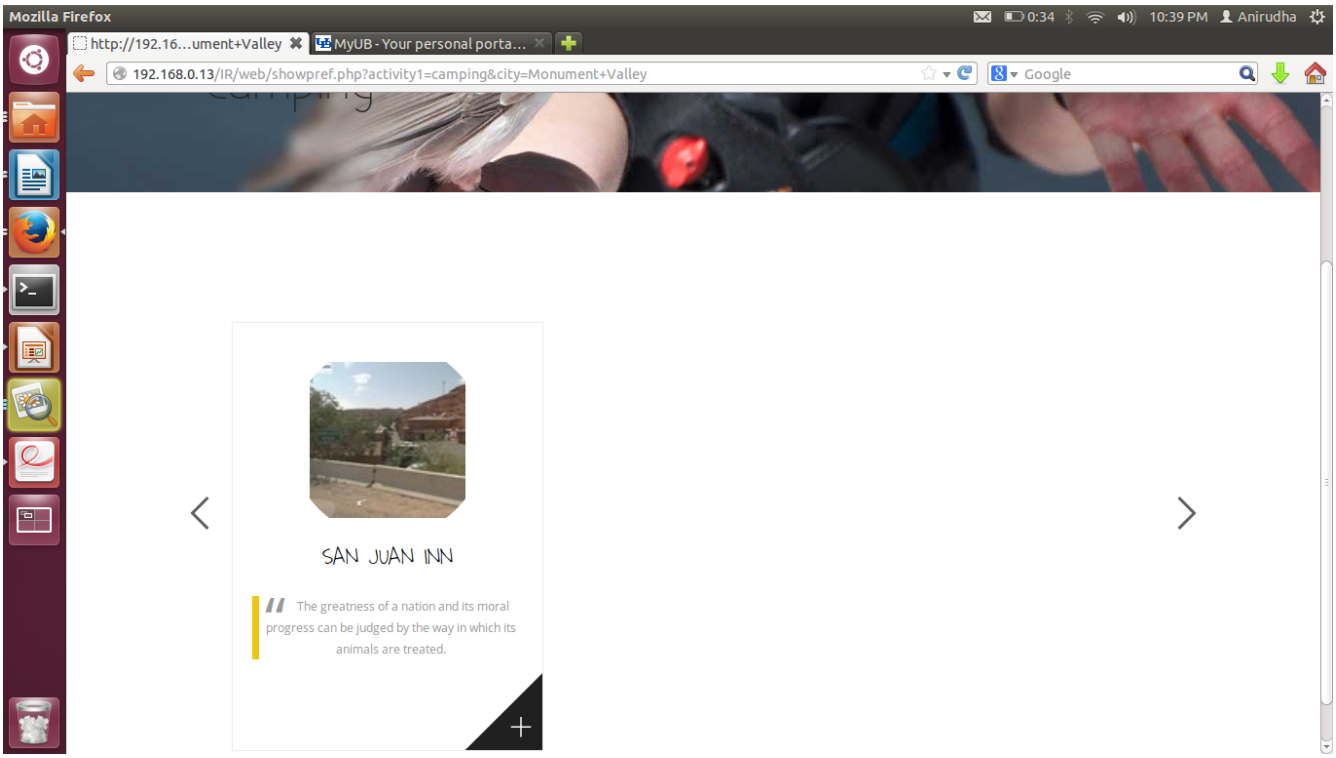
1. Input Level1: Set of activities (camping)



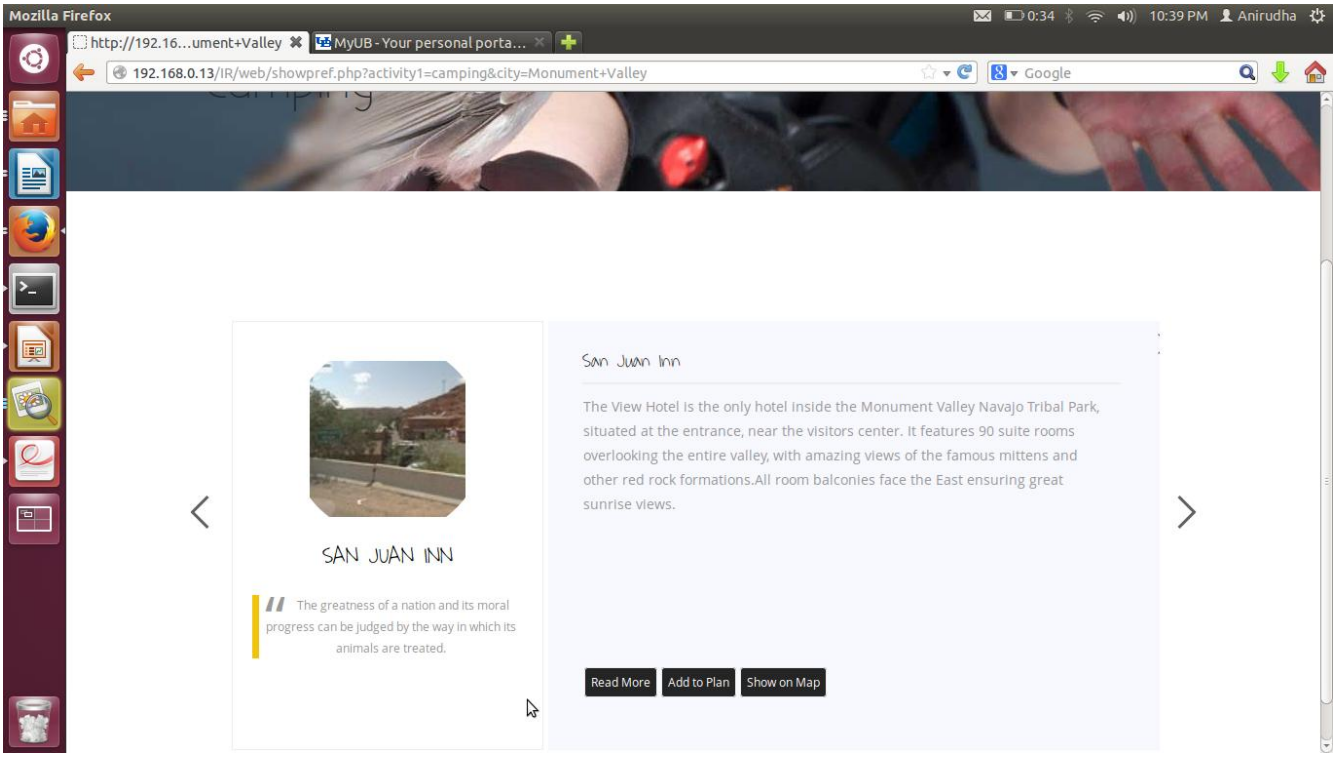


2. Level1: Various cities displayed cities displayed. (Select Monument Valley.)

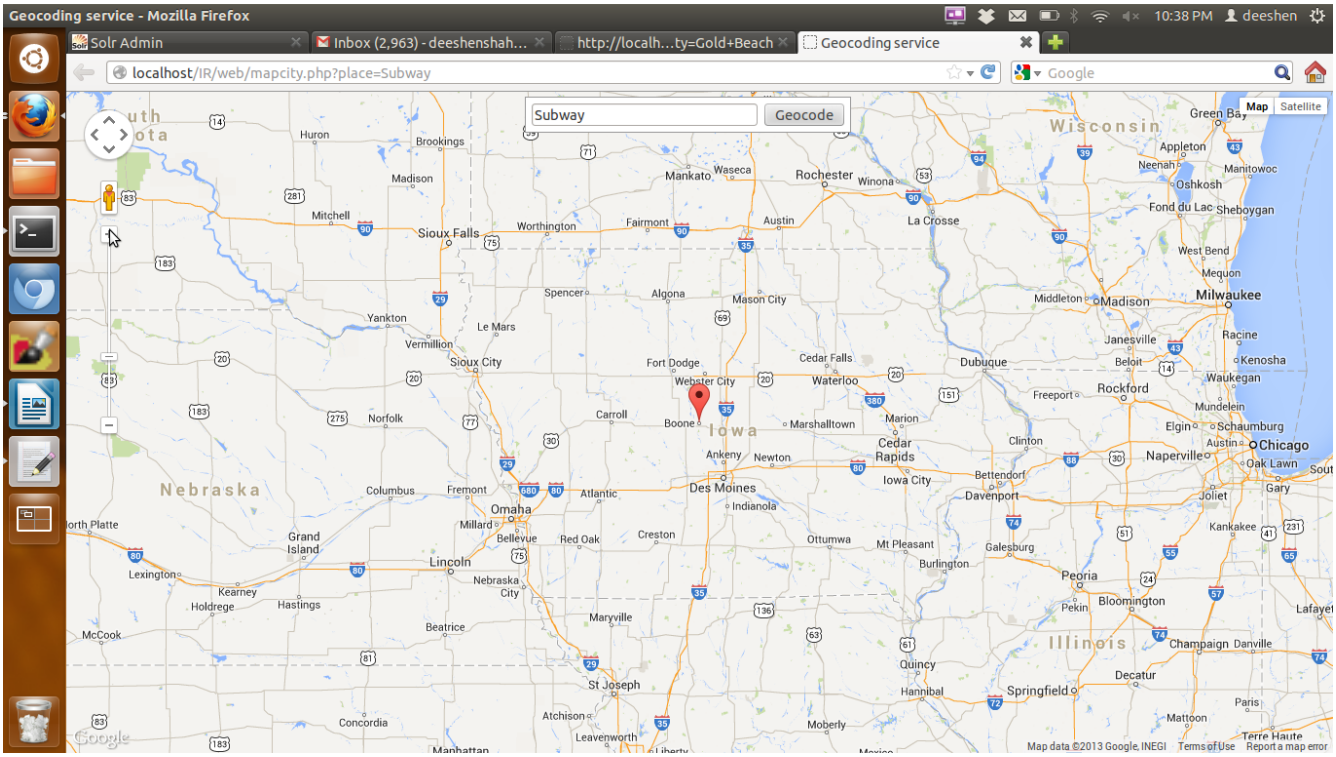
3. Level2: Monument Valley city selected and various results for camping activity displayed.



4. Level3: San Jaun Inn place selected and detail information for it is displayed.



Level3: Detail information about San Juann Inn displayed.



Approach2:

1. Select a particular state-->List all the cities and the activities possible within that state (Level1)
2. Select a particular activity--> Cities will be filtered and results will display only the city supporting the activity (Level2)
3. Select a particular city -->Display all the results for the selected city and activity(Level 3)
4. Select a particular result/place → Display details or information about the selected place(Level 4)

6.3 User Personalized Results/ Ranking

The ranking of results depends on two factors i.e. user preference and global weight of the document. In order to handle these two scenarios we have implemented a methodology to take care of both local (user searched) and global weights on click events. At an initial stage, all documents are given equal weightage. For every click, we update (increase) the global weight by 1 and add document in the solr to add user preference. This document is uniquely identified by combination of IP address and doc-ID.

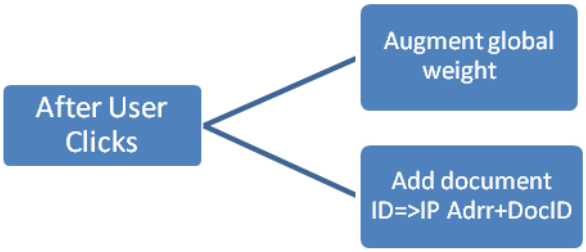


Fig: Weight Update Rule

Fig: Ranking of results

Thereafter if user clicks same result, we update the weight of newly added document by 1. Results are then ranked by the combination of both local and global weights i.e. **Combined weight = Local^0.9 + Global^0.1.** This helps to give more preference results to user query.

6.4 GEO Spatial Sorted Search

This feature in our project helps the user to identify the results of similar category in nearby proximity. For e.g. user queries for restaurants and clicks on a specific one, he/she might be interested to know the nearby results. To implement this feature, we extract the latitude, longitude and category of restaurant from the Wikivoage page template. We then execute the following SolrQuery

SolrQuery

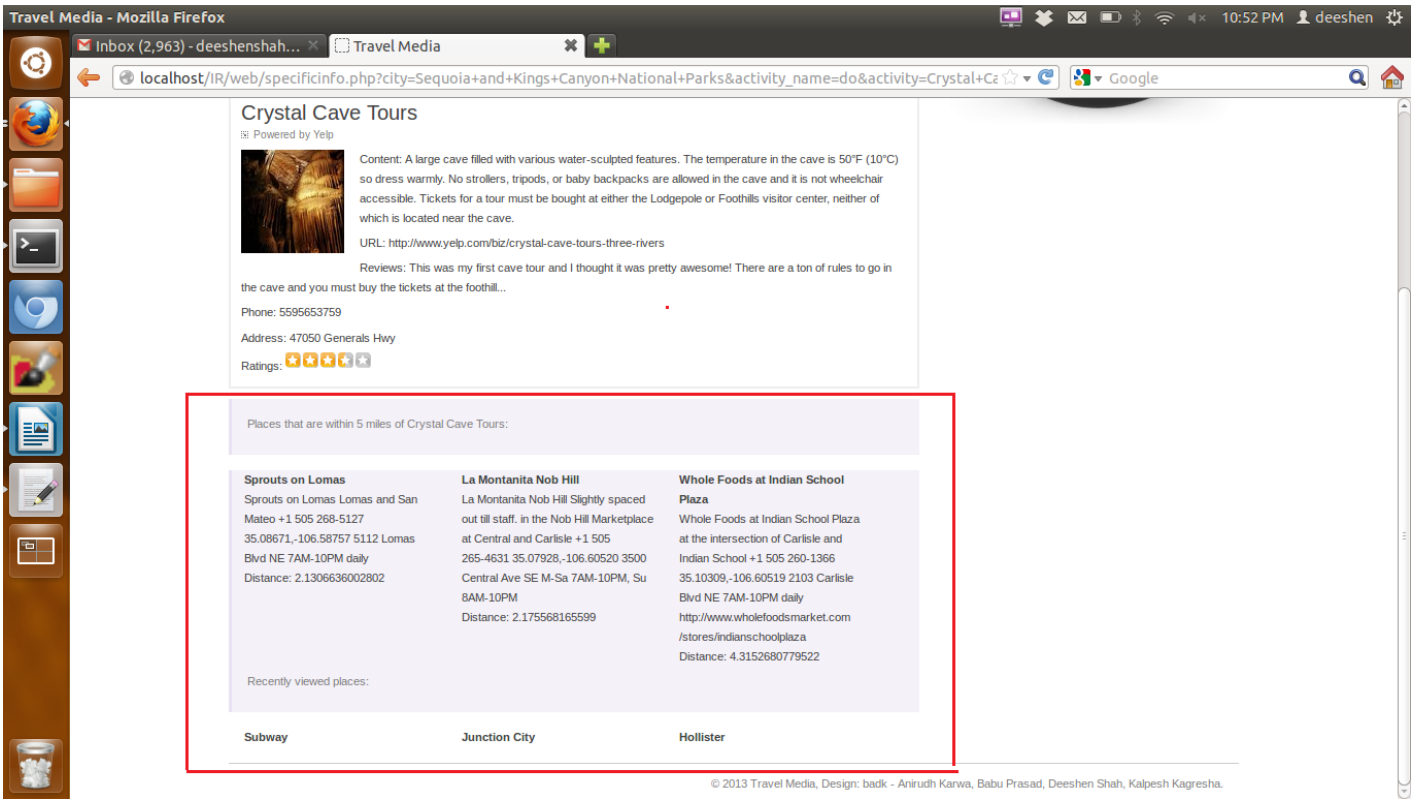
http://localhost:8983/solr/collection1/select?q=title:AlbuquerqueANDsearchfield:category&fq={!bbox}&pt=sfield:store&d=10&fl=id,name,displaycontent,closedist:geodist()&sort=geodist () asc & rows=5

Where
Query q = Search for specific city and category
FilterQuery fq

- Bounded-box(bbox) filter to encompass all points of interest with approximate distance
- Spatial field: store (class = solr.LatLonType)
- Distance of nearby proximity

Fields = Id, name, display content and distance of document
Sort = Sort the results in ascending order of distance
Rows: No of results required to display

Screenshots: Figure Geo Spatial Sorted Search



6.5 Most Popular results

Popular results are those which are clicked by maximum no. of users with regards to given category and location/city/state. These results are obtained by sorting the global weight of all documents. User can view these results after selection of a particular document.

SolrQuery

http://localhost:8983/solr/collection1/select?q=title:cityORstate:stateNameANDsearchfield:category & fl=id, name, displaycontent & sort=weight desc & rows=10

Where
Weight: field of document

The above query inputs title, state and category and returns the results sorted by the global weight of document.

6.6 Recently Viewed results

While exploring the complete details of interested places, a separate list of places that are recently viewed by the user is shown in the User interface. As the timestamp information is stored in the local cache, recently viewed places result is achieved by retrieving the search results sorted by most recent accessed time for every place that is viewed by the user

http://192.168.0.19:8983/solr/travelsearch/select?q=state%3A%22texas%22&fl=id,time,title,_version_&rows=100&sort=time+desc&wt=json&indent=true&facet=true&facet.field=searchfield

Where
time: timestamp of document viewed

The above query inputs title, state and category and returns the results sorted by the global weight of document.

6.7 External Tracking API

- **Google Place API for fetching Place Images**

We have used Google Place and Yelp API for showing pictures and coupons that are relevant to the user's travel. First we have used Google Places API and Panoramio API for showing the pictures in Google Map. The Google Place Widget API is a JavaScript library that provides easy-to-use graphical UI elements and search capabilities so that you can show photos on your web site.
From section 7 we know that we will get the coordinates (latitudes and longitudes) and we can send these coordinates to the Google Place API in order to get the images.
For eg: Giving the query shown below will give us the response as shown in Table 2 [5].

https://maps.googleapis.com/maps/api/place/details/json?reference=<big long key for place>&sensor=true&key=AIzaSyAiFpFd85eMtfbvmVNEYuNds5TEF9FjIPI


```
{
  "html_attributions": [],
  "result": {
    "address_components": [
      {
        "long_name": "48",
        "short_name": "48",
        "types": [ "street_number"]
      },
      {
        "long_name": "Pirrama Road",
        "short_name": "Pirrama Road",
        "types": [ "route"]
      },
      {
        "long_name": "Pyrmont",
        "short_name": "Pyrmont",
        "types": [ "locality", "political"]
      },
      {
        "long_name": "NSW",
        "short_name": "NSW",
        "types": [ "administrative_area_level_1",
"political"]
      },
      {
        "long_name": "AU",
        "short_name": "AU",
        "types": [ "country", "political"]
      },
      {
        "long_name": "2009",
        "short_name": "2009",
        "types": [ "postal_code"]
      }
    ],
    "formatted_address": "5/48 Pirrama Road, Pyrmont
NSW, Australia",
    "formatted_phone_number": "(02) 9374 4000",
    "geometry": {
      "location": {
        "lat": -33.8669710,
        "lng": 151.1958750
      }
    },
    "icon":
"http://maps.gstatic.com/mapfiles/place_api/icons/gen
eric_business-71.png",
    "id":
```

```
 "4f89212bf76dde31f092cfc14d7506555d85b5c7",
    "international_phone_number": "+61 2 9374 4000",
    "name": "Google Sydney",
    "rating": 4.60,
    "reference":
 "CnRlAAAAAfV6JIqSzL8Cf4VnXn0EaI1d5k3IPhdk
 Eonq0MxiUbQFFSVuptVbXbNH4mrevb0bc7G8yWq
 TUv76i4KTuO_Wf3OrRHjCJJwzQ0mNLjbYGSVqy
 2eqyrgOUkl6S_sJfTbHzWZYrfPy7KZaet0mM5S6thI
 QJYuy5v_JD--ZxXEJLWTQRRoU5UaciXBBo89K-
 bce18Ii9RsEIws",
    "types": [ "store", "establishment"],
    "url":
 "http://maps.google.com/maps/place?cid=1028111959
 6374313554",
    "vicinity": "5/48 Pirrama Road, Pyrmont",
    "website": "http://www.google.com.au/"
  },
  "status": "OK"
 }
```

- **Google Map API for showing results on Map**

This module is used for showing the location to user about how to reach to his/her destination. For this purpose, we have decided to use Google Map API for showing the directions to the user.

API = Application programming interface. An API is a specification used by software components to communicate with each other. Here our web application will communicate with the Google service in order to show in the map. It will give us the set of locations. This set of locations will be used to communicate with the Google Map API. We will use Google Geocoding API for converting the locations into Latitude and Longitude. Geocoding is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers or position the map [1]. Now after we get the latitude and longitude we pass these values to Google Map API to show the location on the Map. A simple code snippet is shown in Table 1. for showing location on the Google Map.

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyDY0kkJiTPVd2U7aTOAwhc9ySH6oHxOIYM&sensor=false">
</script>
<script>
function initialize()
{
var mapProp = {
  center:new google.maps.LatLng(51.508742,-0.120850),
  zoom:5,
  mapTypeId:google.maps.MapTypeId.ROADMAP
  };
var map=new google.maps.Map(document.getElementById("googleMap")
  ,mapProp);
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
</head>
<body>
<div id="googleMap"
style="width:500px;height:380px;"></div>
</body>
</html>
```

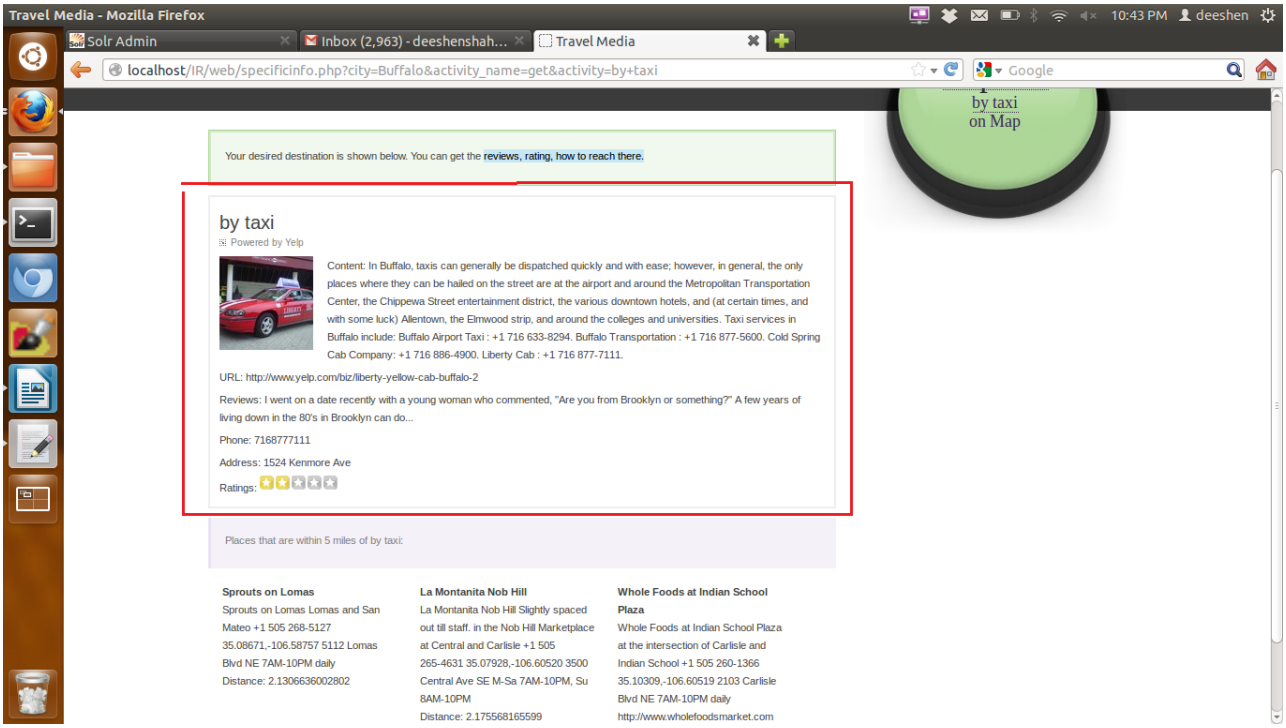
- **Yelp API for fetching Places Reviews, Content, Ratings, Images**

We are using Yelp API for fetching ratings, reviews, images of a specific place.

We are using the following URL for fetching the details of particular place ar particular locations: http://api.yelp.com/v2/search?term=placenaem&location=location

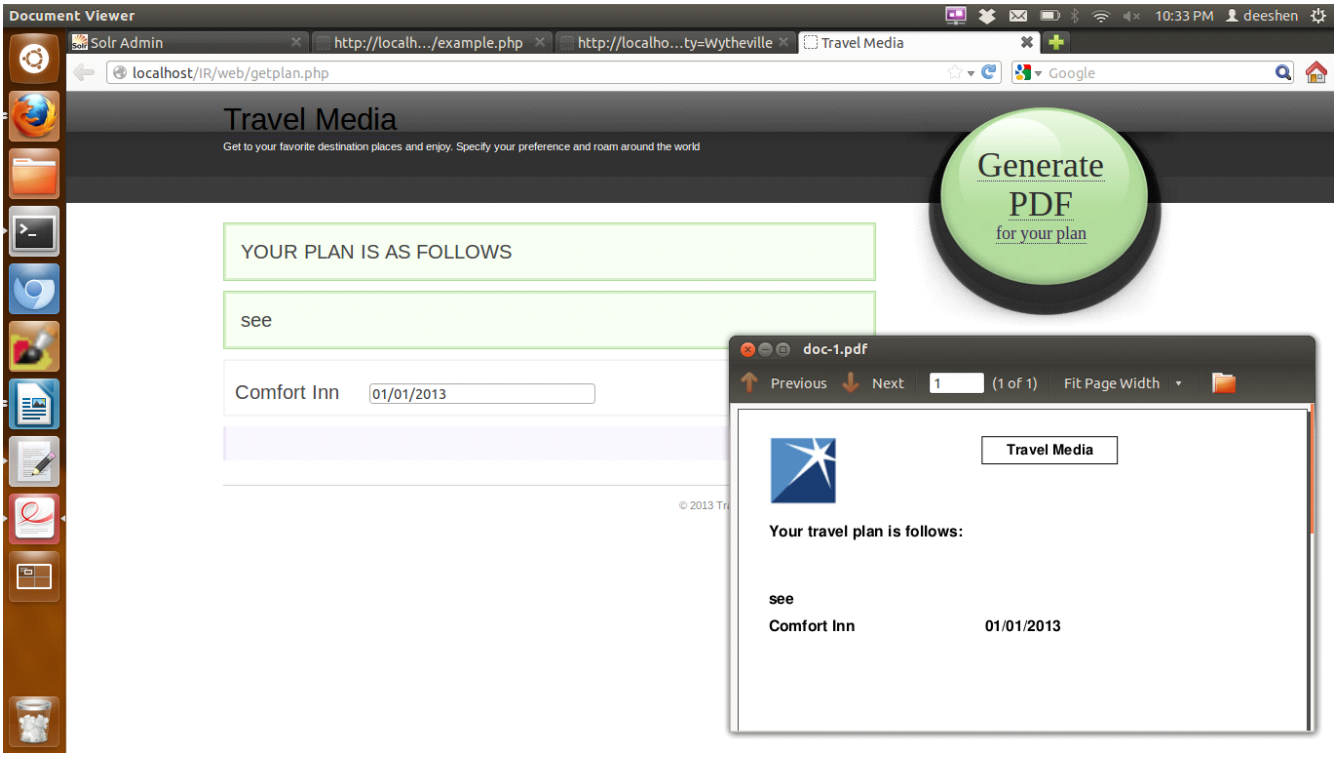
Following is the response for the query above

"region": {"span": {"latitude_delta": 0.0, "longitude_delta": 0.0}, "center": {"latitude": 43.083731700000001, "longitude": -79.082836}}, "total": 2, "businesses": [{"is_claimed": false, "rating": 2.5, "mobile_url": "http://m.yelp.ca/biz/best-western-fallsview-niagara-falls", "rating_img_url": "http://s3-media4.ak.yelpcdn.com/assets/2/www/img/c7fb9aff59f9/ico/stars/v1/stars_2_half.png", "review_count": 19, "name": "Best Western Fallsview", "snippet_image_url": "http://s3-media1.ak.yelpcdn.com/photo/jmi-YJkvM4AX84W96dacdg/ms.jpg", "rating_img_url_small": "http://s3-media4.ak.yelpcdn.com/assets/2/www/img/8e8633e5f8f0/ico/stars/v1/stars_small_2_half.png", "url": "http://www.yelp.ca/biz/best-western-fallsview-niagara-falls", "phone": "9053560551", "snippet_text": "we stayed at the hotel for 3 nights and had the breakfast brunch for 8.99 on two mornings. the buffet was really quite good. compared to others we've eaten...", "image_url": "http://s3-media3.ak.yelpcdn.com/bphoto/wBho7U5_QGfQzS-RKorZJg/ms.jpg", "categories": [["Hotels", "hotels"]], "display_phone": "+1-905-356-0551", "rating_img_url_large": "http://s3-media2.ak.yelpcdn.com/assets/2/www/img/d63e3add9901/ico/stars/v1/stars_large_2_half.png", "id": "best-western-fallsview-niagara-falls", "is_closed": false, "location": {"city": "Niagara Falls", "display_address": ["6289 Fallsview Boulevard", "Niagara Falls, ON L2G 3V7", "Canada"], "postal_code": "L2G 3V7", "country_code": "CA", "address": ["6289 Fallsview Boulevard"], "state_code": "ON"}}]}
http://s3-media3.ak.yelpcdn.com/bphoto/wBho7U5_QGfQzS-RKorZJg/ms.jpg

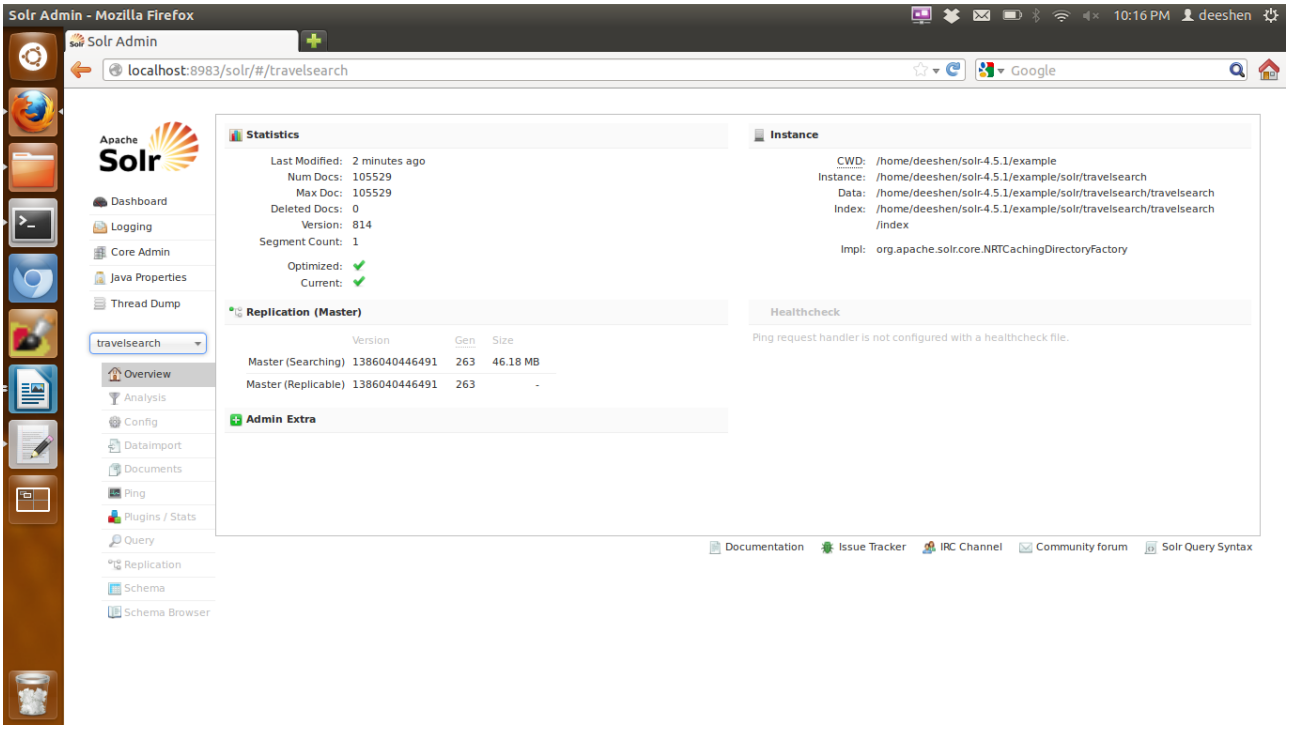


6.8 Create Plan for your travel: Generate pdf report for your travel plan

One of the major feature of our application is to provide user the option to create his plan/itinerary for travel. Our Application provides the opportunity to browse through various travel destination places and also provide user the option to add to plan. User can add as many places as he want. After selecting all the travel places he can generate a plan for his travel wherein we are creating a pdf for the user which he can download directly from the Web Application.

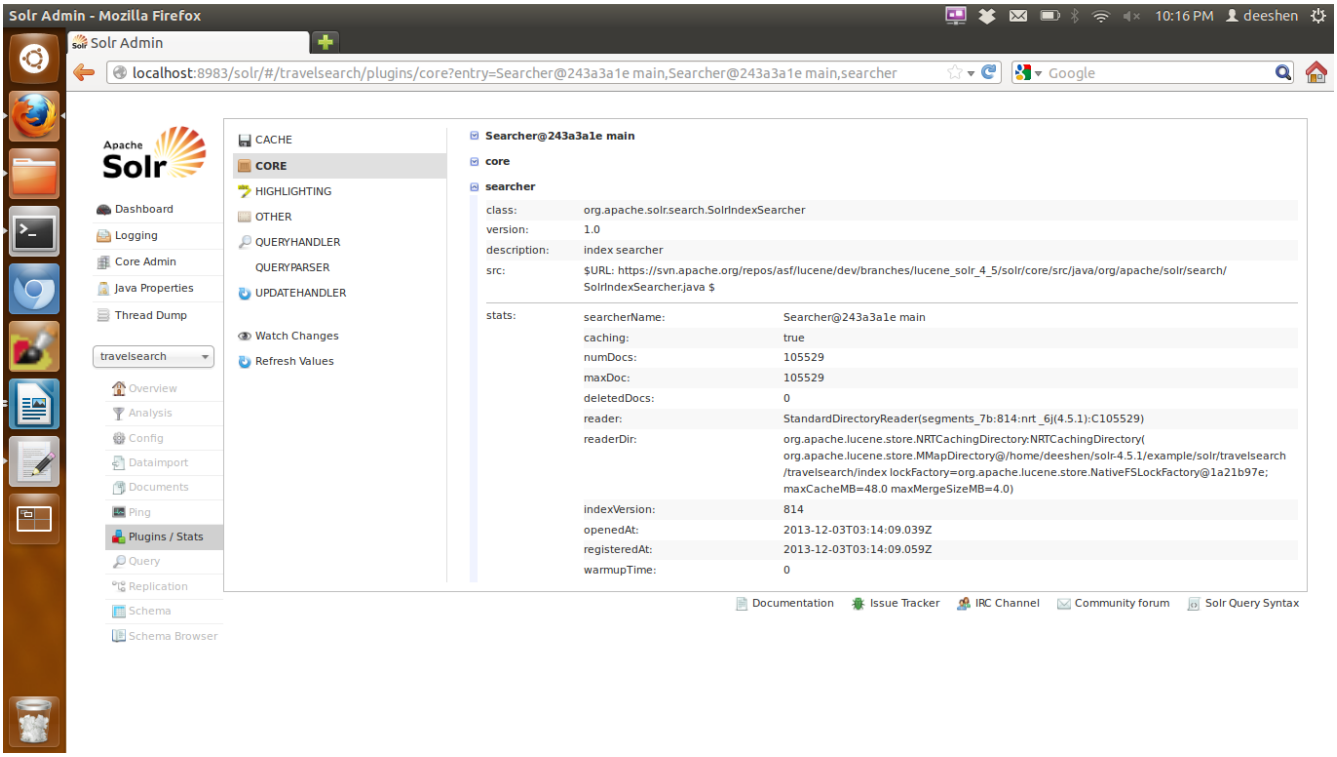


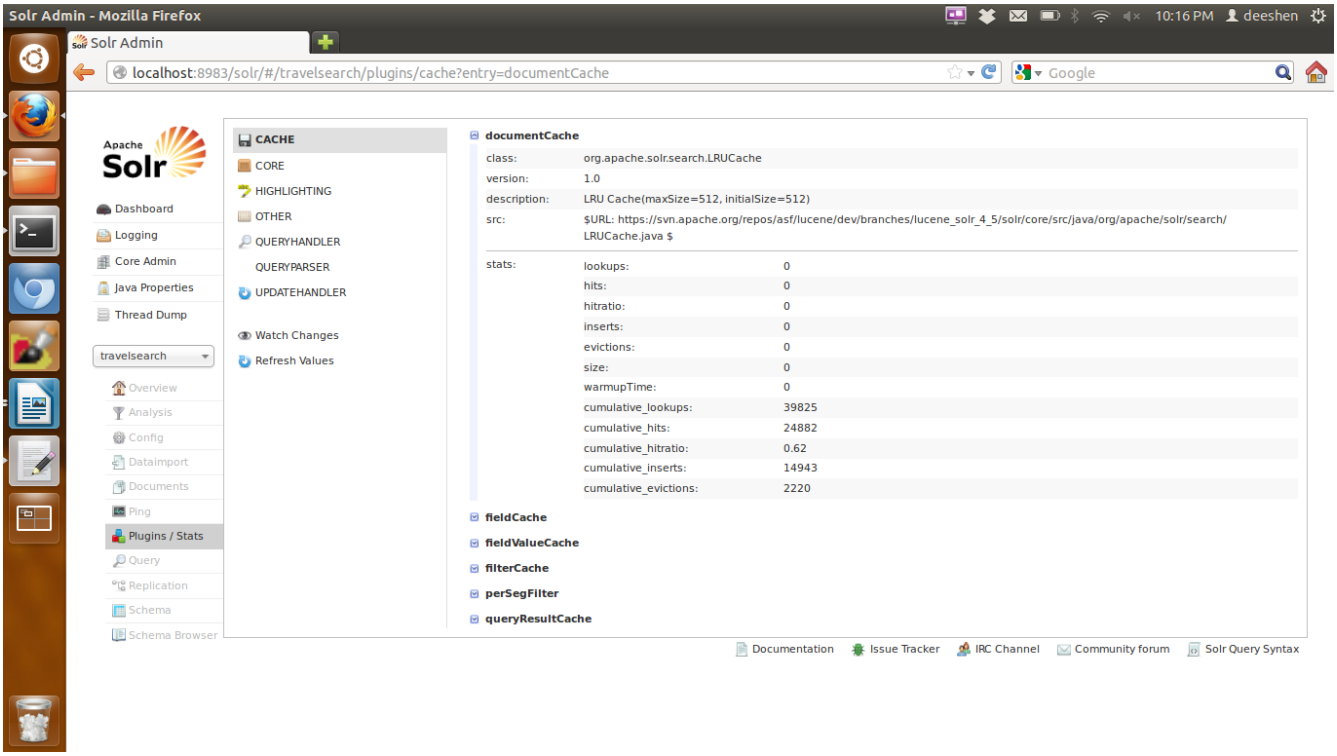
7 Solr Statistics



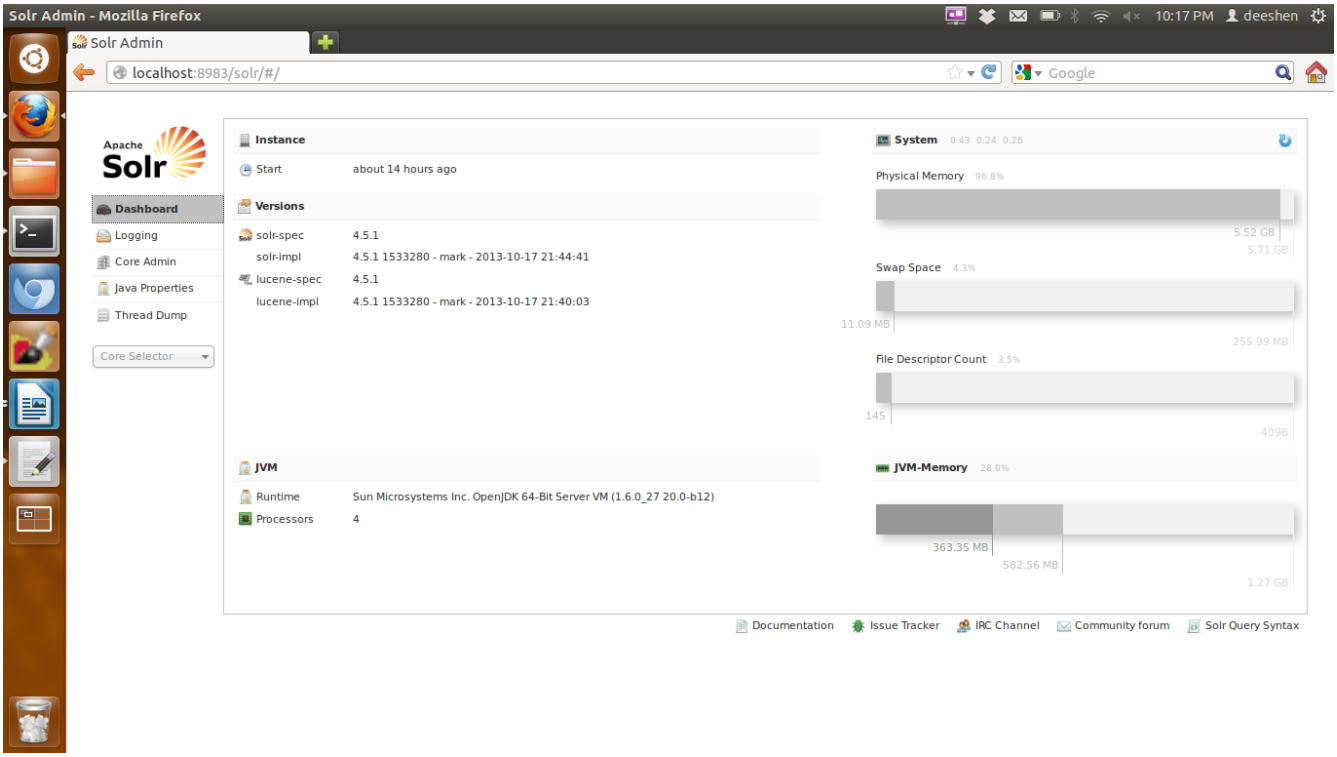
1. Solr home page:

2. Solr Document Cache.

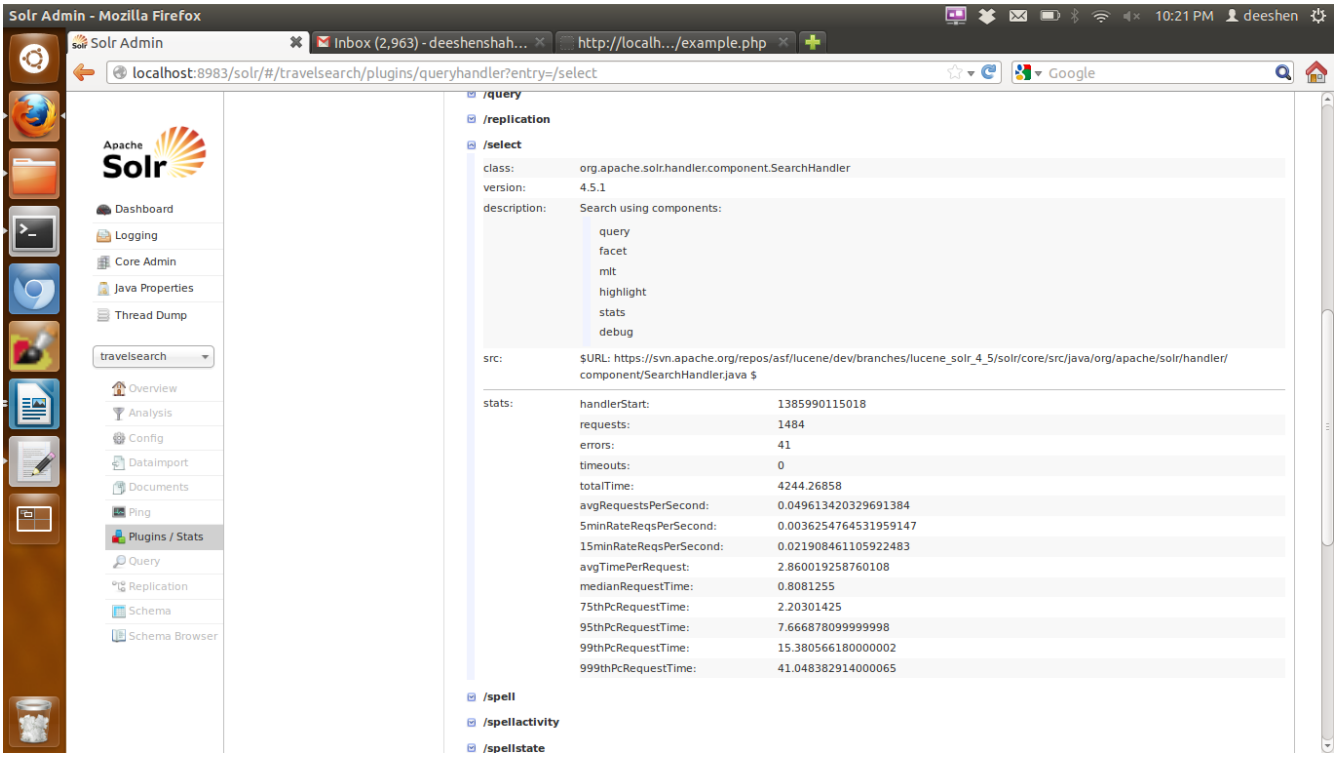




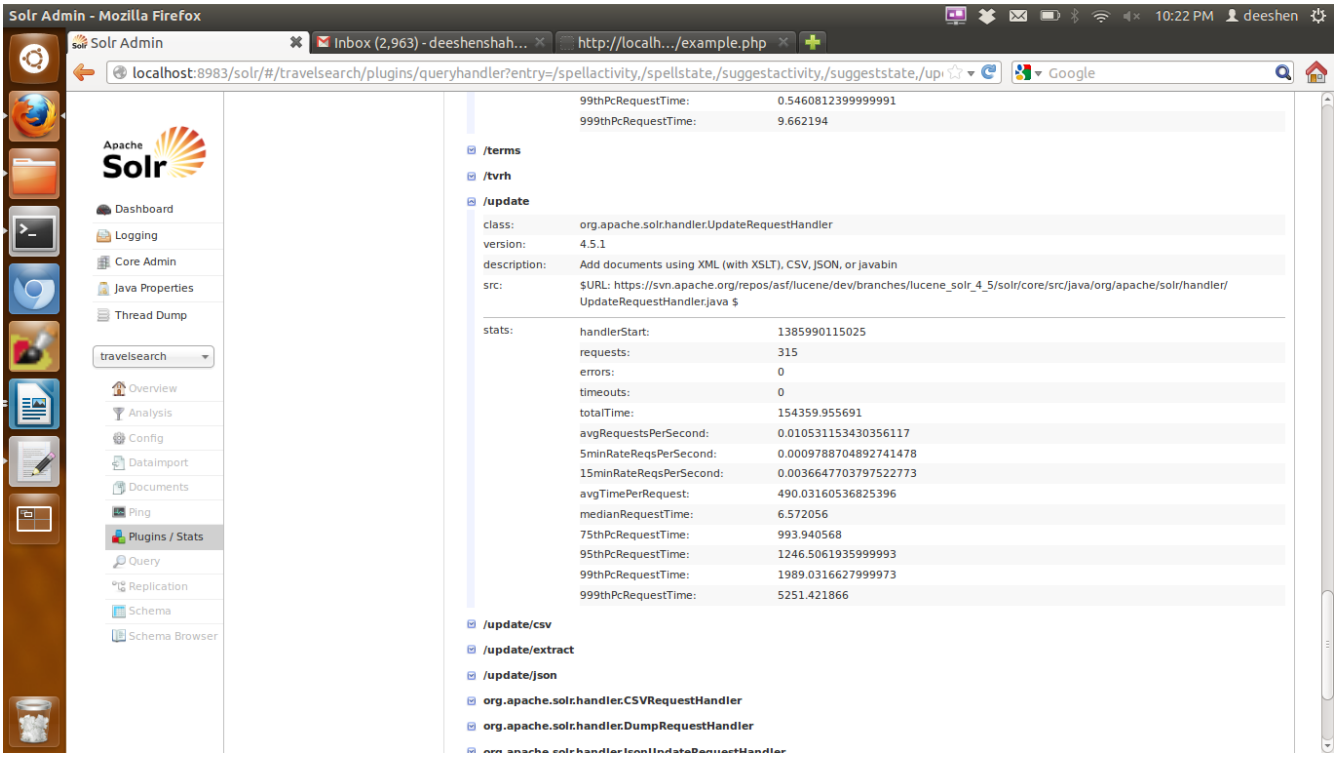
3. Solr DashBoard



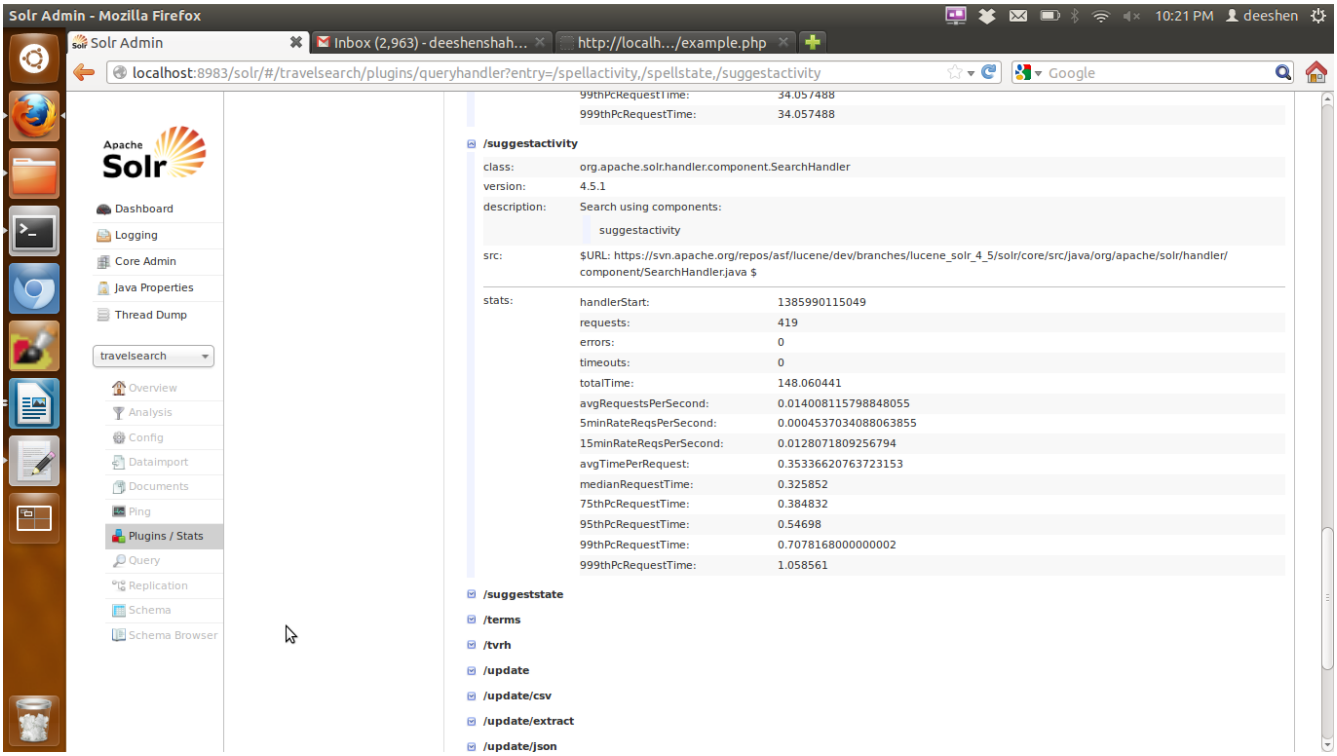
4. Select Handler.



5. Update Handler



6.Suggestactivity Handler


Solr Admin - Mozilla Firefox
10:21 PM deeshen
Solr Admin
Inbox (2,963) - deeshenshah...
http://localh.../example.php
localhost:8983/solr/#/travelsearch/plugins/queryhandler?entry=/spellactivity,/spellstate,/suggestactivity
Google
Apache Solr
Dashboard
Logging
Core Admin
Java Properties
Thread Dump
travelsearch
Overview
Analysis
Config
Dataimport
Documents
Ping
Plugins / Stats
Query
Replication
Schema
Schema Browser
99thPcRequestTime: 34.057488
999thPcRequestTime: 34.057488
/suggestactivity
class: org.apache.solr.handler.component.SearchHandler
version: 4.5.1
description: Search using components:
suggestactivity
src: $URL: https://svn.apache.org/repos/asf/lucene/dev/branches/lucene_solr_4_5/solr/core/src/java/org/apache/solr/handler/component/SearchHandler.java $
stats:
handlerStart: 1385990115049
requests: 419
errors: 0
timeouts: 0
totalTime: 148.060441
avgRequestsPerSecond: 0.014008115798848055
5minRateReqsPerSecond: 0.0004537034088063855
15minRateReqsPerSecond: 0.0128071809256794
avgTimePerRequest: 0.3533662076372315
medianRequestTime: 0.325852
75thPcRequestTime: 0.384832
95thPcRequestTime: 0.54698
99thPcRequestTime: 0.7078168000000002
999thPcRequestTime: 1.058561
/suggeststate
/terms
/tvrh
/update
/update/csv
/update/extract
/update/json

8 WORK DISTRIBUTION

Following is the work distribution for our work.

MODULE NAME	TEAM MEMBER
Database Module	Anirudh, Kalpesh
Search Module	Babu, Deeshen
Input Module, Parser	Kalpesh
Tracker Module	Deeshen,Babu
External API Module	Deeshen,Anirudh
Web Application Module	Deeshen
Query Processing Module	Babu

9 CONCLUSION

At the end of this project we will have a web application that will help user to apply multi-filter option to discover his/her travel destinations.

10 REFERENCES

[1] https://developers.google.com/maps/documentation/geocoding/?csw=1

[2] http://www.w3schools.com/googleAPI/google_maps_basic.asp

[3] http://www.sitepoint.com/find-a-route-using-the-geolocation-and-the-google-maps-api/

[4] http://www.panoramio.com/api/widget/api.html

[5] http://stackoverflow.com/questions/8462253/how-to-get-a-picture-of-a-place-from-google-maps-or-places-api

[6] http://www.yelp.com/developers/getting_started

[7]http://www.geonames.org/

[8]http://wiki.apache.org/solr/

[9]http://dumps.wikimedia.org/enwikivoyage/20131115/