iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** XI **Month of publication:** November 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue XI November 2017- Available at www.ijraset.com

Application of Semantic Web on Massive open online Course (MOOC) Using Linked Data [Oneduhub]

Anjali Dwivedi[1], Varsha Bhosale[2]
[1]*M.E Student, Computer Engineering, S.L.R.T.C.E, Mumbai University*
[2]*Vice-Principle/Associate Professor, Computer Engineering, V.I.T , Mumbai University*

Abstract: ***The term Linked Data is used for connecting and publishing structured data on the Web. Over a past few years these practice have been adopted by increasing number of data providers, leading to the creation of a data space containing billions of assertions- the Web of Data. Linked Data is an emerging trend used by the top companies for promoting their own means of marking up semantic data, connecting and publishing data on Web. Despite the increasing generality of Linked Data, there are a limited number of applications that take advantage of its capability, particularly in the domain of online/open education on web. In this project we are using Semantic web technologies to create a semantic data model for online/open educational data, specifically about Massive Open Online Courses (MOOCs), connecting and publishing this data as linked data on the Web. Data from various open/online courses educational providers is collected, integrated and published as Linked Data. We present a web portal that utilizes the data to discover and compare open courseware.***

Keywords: ***Linked data; Semantic web; Mooc; online education; RDF; Ontology***

I. INTRODUCTION

Linked Data include using the Web to create typed links between data from different sources [1]. Source data may vary in size, location, subject-matter and how compatibly it is structured. Typed links generated from this data create define properties explicitly and uniformity in an effort to make data machine readable form. This is opening up opportunities for applications that were previously impractical such as Berners-Lee's intelligent agents [2].Linked Data is relatively not evaluated in the domain of education. Although there are several data models for structuring educational data as well as repositories adopting these models, Linked Data-driven educational applications are far and few between. As a result, enterprise such as the Linked up Challenge have surfaced to encourage innovative applications focused on open educational data [3].Massive Open Online Courses or MOOCs are online courses accessible to anyone on the web. Hundreds of institutions have joined in an effort to make education more accessible by teaming up with MOOC providers such as Alison, udacity, edX and Courser. Delivering course content through videos and lectures as well as traditional materials such as problem sets and reading, MOOCs encourage interactivity between student and professors around the world by way of graded assessment and discussion forums. Courser, a leading MOOC provider, offers a RESTful API [6] for most information associated with their course catalogue. This includes properties such as a course's instructor, title and syllabus details. Although Coursera's course catalogue data is easily accessible as JSON, there is no option to retrieve and use it in a Linked Data format such as the Resource Description Framework (RDF). Moreover, there is no Linked Data available for MOOCs or an ontology that denotes properties distinct to MOOC.

II. PROPOSED MODEL

Let us finally discuss the proposed framework figure 1. There is various consideration level of proposed framework thus these are being proposed in context with certain already existing framework not only proposed but also implemented. On the basis of framework proposed in [12], the framework has been proposed that explains the layers into better modules and in a sequence. It clearly depicts that online course s (MOOC) documents will be extracted and gathered using scrapy tool, once the data is gathered it is available is different format so ontology tool will create a graph of collected data into one common format understandable by the developer. RDF is used to convert the graph into xml format so the web application /developer and user can easily understand the documents and it is then uploaded on our web portal and then the document is available for end user to use. It also runs the recommendation engine and stores data into MySQL database

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue XI November 2017- Available at www.ijraset.com

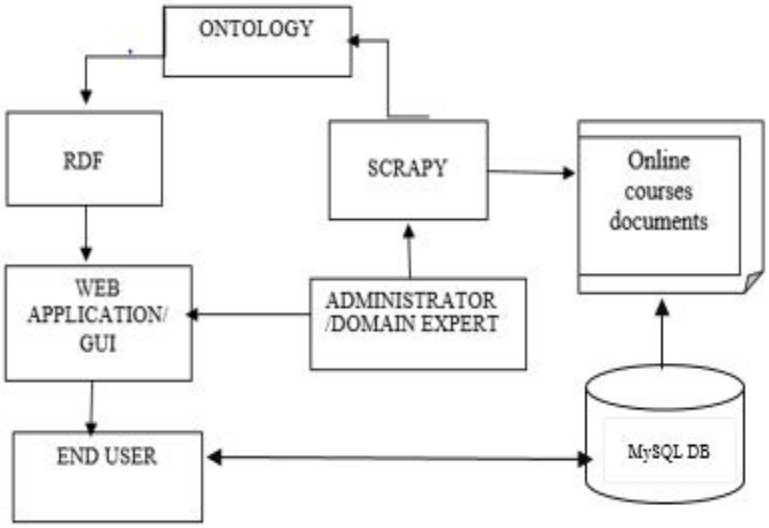


Fig. 1 the Proposed Framework

III. BACKGROUND

A. *Simple Protocol and RDF Query Language*

RDF Query Language or simple protocol or SPARQL is an RDF query language that allows users to modify and retrieve data stored in RDF format [7]. SPARQL is used as our application's query language in order to retrieve data to populate web pages with course information.

B. *Resource Description Framework*

The most notable model for Linked Data is the Resource Description Framework (RDF), which encodes data as predicate, subject, predicate and object triples [7]. The object and subject are both Uniform Resource Identifiers (URIs), while the predicate inform how the subject and object are related to each other, using a URI. For the purposes of this paper, Linked Data is presented as XML/RDF, XML syntax for RDF, which is also used in the development of our application

C. *Linked Data Principles*

Tim Berners-Lee established a collection of rules for business knowledge on the net in order that this knowledge will becomes a part of a worldwide area during which each resources square measure connected. the foundations square measure as follows:

1) Use URIs as names for things
2) Als0 Use protocol URIs in order that individuals will rummage around for those names
3) Once somebody appearance for a URI, offer helpful data (RDF, SPARQL).
4) Invariably embrace links to different URIs, in order that they will discover a lot of things.

These principles by Tim offer basis information contributed to a connected knowledge during which a several variety of datasets from completely different fields in international house of human information area unit interconnected. Our project aims to figure on these principles

D. *Building Ontologies*

Ontology is common vocabulary for researcher who needs to share information in a domain. We develop ontology to share common understanding of structure of information among software agent and people to separate domain knowledge from the operational knowledge and to analyse domain knowledge.

E. *Scrapy*

Scrapy is written in python it's an open source web crawling framework. It is originally designed for web scraping, it can also be used to extract data using APIs or as a general purpose web crawler. Now a days it is maintained by Scraping hub Ltd., a web scraping development and services company. Scrapy project structure is built around 'spiders', which are self-contained crawlers which are given a set of instructions. Scrapy provides a shell (web crawling) which can be used by developers to test their assumptions on a site's behaviour.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue XI November 2017- Available at www.ijraset.com

IV. ON-EDUHUB

On-EduHub is a web application which combines multiple online courses from different mooc provider as Linked Data and utilizes that data to search and compare online courseware. This section of the paper summaries our approach including selecting our providers, data generation, modelling and the development of our web application.

A. MOOC Providers

Coursera is one amongst the biggest MOOC supplier within the world with 641 courses from 108 establishments and seven.1 million as of Gregorian calendar month 2014 [14]. These courses span twenty five classes together with four subcategories of engineering. All course details area unit recoverable by communications protocol GET technique exploitation Coursera's reposeful prospectus API and came back in JSON.

edX is another premier MOOC supplier with over a pair of.5 million users and over two hundred courses as of June 2014 [15]. edX courses area unit distributed among twenty nine classes, several of that overlap with Coursera's. edX doesn't offer associate degree API for accessing their prospectus, however, as of June 2013, edX's entire platform is ASCII text file.

Alison is associate degree e-learning supplier and academy. Its main objective is to alter individuals to achieve basic academic data and work skills. Contrary to different the bulk of Alison's learner's area unit situated within the developing world with the quickest growing variety of users in Bharat. Alison is creating the net education supplier one amongst the most important MOOCs outside of the United States of America

B. Data Model

The data model of our website is organized in hierarchy of types associated with set of properties as we are taking course detail from different mooc websites so it is possible that their properties may be different from each other so to reduce the complexity, we have define some properties common to different mooc provider in our website like course name, course id, provider ,session , category , time taken to complete the course , all these data properties to be used as model for data generation. The final outcome is an ontology in which classes are defined using web ontology language.

C. Data Generation

Courser's catalogue of courses is reachable through JSON and RESTful API as the data exchange format. Using Requests, a Python HTTP library, we retrieved a full list of courses, universities, categories, instructors and course sessions in JSON using the GET method. edX and Alison do not have an API therefore it became necessary to use a scraper to obtain course data. We are using Scrapy, web crawling framework and open-source screen scraping for Python to retrieve properties from edX and Udacity course pages with XPath selectors. Screen uniformity with the data we retrieve from Coursera. After collecting the data from multiple online courses, we create Linked Data from JSON using open source Semantic Web framework(Apache Jena) for Java. First, we import the ontology model we created in Portege to retrieve the types and properties to be assigned. We use three methods i.e. read json, map property and rdf graph , one for each course provider, to read JSON specific to each provider (using Google Json), map property names from JSON to our ontology's properties, and write each property to the RDF graph. The RDF is output in a flat file, serialized as RDF/XML.As recommended by Berners-Lee in his Linked Data Principles, each property is assigned a URI. HTTP URIs are used so these resources can be looked up. Because the majority of our data points to URIs within the same RDF, hash URIs are used to identify the local resources. Further details on the naming schemes of these URIs are available in the implementation part of this paper, where our RDF data for Coursera, edX, and Alison is discussed in details.

D. Web Application

Our web application is created using different technologies. We have used Python based micro-MVC framework called Flask for the web application. Frontend is coded using HTML and Bootstrap so that the application is responsive. We have also used a few external dependencies like SQLAlchemy to have an ORM wrapper on top of MySQL Database. Our primary source is RDF xml for the courses content but the recommendation data is stored in the MySQL database. We have routes like /courses, /login and /recommendations to get the relevant data.

Most important file in the MVC code is app.py. It consists of all the routes and business logic to process the XML file and dump data on to the screen. HTML is a set of templates supported using Jinja and Bootstrap. We use RDFLib to parse and process the XML too. It helps us read the N3Triples and parse into Python objects. These python objects are converted to relevant data types. Front-end filters have ran using jQuery especially the Dictionary filter and Course Comparison modal.

```
filename = main_file_name

courses = []
university_list = {}
unique_categories = {}

with open('unvi-ranking.csv', 'rb') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader: # read a row as {column1: value1, column2: value2,...}
        if row['unv-name'] not in university_list:
            university_list[row['unv-name']] = row['unv-rank']
graph = Graph()
graph.parse(filename, format='xml')

count = 0
index_count = 0
for course in graph.subjects(RDF.type, FOAF.Basic):
    course_info = {}

    index_count = index_count + 1

    course_info['id'] = index_count

    for url in graph.objects(course, FOAF.url):
        course_info['url'] = url
    #print course_info['url']
    for title in graph.objects(course, FOAF.title):
        course_info['title'] = title
```

Fig. 2 MVC code in app.py

V. IMPLEMENTATION

Our application is categorized into three sub-projects. (i) Sub-project crawls data in the JSON format from the data sources. (ii) Sub-project creates an RDF xml file consolidating data from the JSON documents. (iii) Sub-project is a Python Flask-based Web application that takes feed from the RDF file and shows the data to the user. It also runs the recommendation engine and stores user information in a MySQL database.

A. Web Crawling

We have taken data from 4 courses websites - Coursera, Udacity, EdX and Alison. All of them are courses platform dealing with different domains across the site categorized into the subjects. Udacity is the only platform that deals with Computer Science. Coursera provides a Rest API access for fetching the data available on the website, whereas the other websites there is no API available. So unfortunately we had to scrap data using different scraping technologies. We have used Urllib, Scrapy and Beautiful Soup, Python libraries for crawling the data.

Scrapy and Beautifulsoup are Python libraries that read HTML from the given URL and parse HTML to pull-out the data based on HTML ids and classes. This section details the process of writing a Scrapy crawler for edX in Python. After starting a new Scrapy project in the terminal, we define the Item or container that will be loaded with scraped data. This is done by creating a scrapy. Item class and defining its attributes as scrapy. These attributes, which are ultimately output as JSON, are named slightly different than our RDF properties. As a consequence, the naming scheme is mapped to the types defined in our ontology later in our Jena methods, which convert the collected JSON into RDF/XML. Sample Field Class is added as shown in the Fig 3.

```
class EdxItem(scrapy.Item):
    name = Field()
    about = Field()
    instructor = Field()
    school = Field()
    courseCode = Field()
    startDate = Field()
    url = Field()
    length = Field()
    effort = Field()
    prereqs = Field()
    video = Field()
    image = Field()
    category = Field()
    source = Field()
    certificates = Field()
```

Fig. 3 Sample Field Class

The site is crawled creating a Crawl Spider based on Scrapy. It starts with a base project URL, scrapy project name, and allowed domains. We create a set of start_urls and it starts processing. Scrapy has a designated parse method where the first set of parsing is done. We create a Selector class and feed the response to it. With Xpath queries supported on Selector, we pull out relevant data. If the scrapping requires further detailing or nested of crawling we the use the following syntax

request = scrapy. Request (course_url,callback=self.parse_catalog)

Where course_url is the URL to be crawled, and parse_catalog is the callback function. We return the request as an array to the function and it turn calls the scrapping methods internal to Scrapy. We use the following command to generate the JSON file.

Scrapy crawl edx -o edx. json. Coursera data is fetched using the Rest API. Sample Python code using Requests library is as attached below Fig. 5.

```
class EdXSpider(CrawlSpider):
    name = "edx"
    allowed_domains = ["edx.org"]
    start_urls = []

    file = ['Computer Science', 'Architecture']
    #file = open('category_map', 'r')
    #
    for line in file:
        #url = "https://www.edx.org/course-list/allschools/" + line.strip() + "/allcourses"
        url = base_url + "/course/?subject=" + line.strip()
        start_urls.append(url)


    def parse(self, response):
        #filename = base_path +  response.url.split("/")[-2]
        #open(filename, write_mode).write(response.body)
        sel = Selector(response)
        sites = sel.xpath('//a[@class="course-link"]/@href').extract()
        requests = []

        for site in sites:
            course_url = base_url + site
            item = EdxItem()
            item['url'] = course_url
            item['source'] = site_name
            parsed = urlparse.urlparse(response.request.url)
            item['category'] = urlparse.parse_qs(parsed.query)['subject']
            request = scrapy.Request(course_url,callback=self.parse_catalog)
            request.meta['item'] = item
            requests.append(request)
        return requests
```

Fig. 4 A sample code to crawl the data.

```
url = "https://api.coursera.org/api/courses.v1?limit=10&fields=id,slug,name,instructorIds,partnerIds,phot

instructor_url = "https://api.coursera.org/api/instructors.v1"
partner_url = "https://api.coursera.org/api/partners.v1"

base_url = "https://www.coursera.org/learn/"

course_data = []

response = requests.get(url)
data = json.loads(response.text)
for course in data['elements']:
    try:
        course_info = {}
        course_info['url'] = base_url + course['slug']
        course_info['category'] = course['domainTypes'][0]['domainId']
        #print course_info['url']
        course_info['courseCode'] = course['slug']
        course_info['name'] = course['name']
        instructor_ids = course['instructorIds']
        course_instructors = []

        for instr_id in instructor_ids:
            ins_url = instructor_url + "/" + str(instr_id)
            instr_response = requests.get(ins_url)
            ins_data = json.loads(instr_response.text)

            for instructor in ins_data['elements']:
                course_instructors.append(instructor['fullName'])

        course_info['instructor'] = course_instructors

        partner_ids = course['partnerIds']
        course_info['school'] = ""
        for partner_id in partner_ids:
            school_url = partner_url + "/" + partner_id
```

Fig. 5 Sample Python code using Requests library.

B. RDF Generation

We have a separate script that takes all the JSON files and creates one RDF.xml file to be fed into the Web Application. RDFLib is a Python library for working with RDF, for storing information on Graphs. RDFLib provides a RDF API, a Graph is a python collection of RDF Subject, Predicate and Object Triples. The triples consist of two components they are URIs (resources) or Literals (values), URIs are grouped together by namespace, common namespaces are included in RDFLib. The library contains serializes and parsers for N3, RDF/XML, N-Quads, NTriples, N-Quads, TriX, Turtle, RDF and Micro data. The library produce a Graph interface which can be supported by any one of a number of Store implementations. This RDFLib package consist of store

implementations for in persistent storage and memory storage on top of the Berkeley database. A SPARQL implementation is included in RDF. We have chosen to store the data in NTriples and export the data into a single .xml file. The script runs all the data from all the json files stored in the data/ directory. We run the file as python rdf-gen.py

C. WEB Application

Our web application is created using different technologies. We have used Python based micro-MVC framework called Flask for the web application. Frontend is coded using HTML and Bootstrap so that the application is responsive. We have also used a few external dependencies like SQLAlchemy to have an ORM wrapper on top of MySQL Database. Our primary source is RDF xml for the courses content but the recommendation data is stored in the MySQL database. We have routes like /courses, /login and /recommendations to get the relevant data.

Most important file in the MVC code is app.py. It consists of all the routes and business logic to process the XML file and dump data on to the screen. HTML is a set of templates supported using Jinja and Bootstrap. We use RDFLib to parse and process the XML too. It helps us read the N3Triples and parse into Python objects. These python objects are converted to relevant data types. Front-end filters have ran using jQuery especially the Dictionary filter and Course Comparison modal.

D. Output

1) Home Page: As shown in the figure below, Home Page of this web Application On - EduHub has a very simple and User Friendly User Interface. The Homepage has a welcome Message on the centre with the Web Application Title on the Upper Left Corner along with the Logo. The Page has some self Explanatory tabs that links to different pages of the Application. The Different Tabs (Links) are in a sequence like Home, Recommendation, Dictionary, Courses, Login/Signup. The Home tab will be the default Tab on opening the page. Recommendation Page offers some recommended courses based on the users visited course. If the user is not logged in it redirects the user to the Login /Sign up Page where he needs to enter his basic credentials like Email Address with a password. The Other Tab named Dictionary provides all the courses links based on the Alphabet selected from the upper part. This Works as a filter for courses on users choice and preference. The Bottom of the Home page has Information about Partner web application from where the data is collected about the respective Courses.

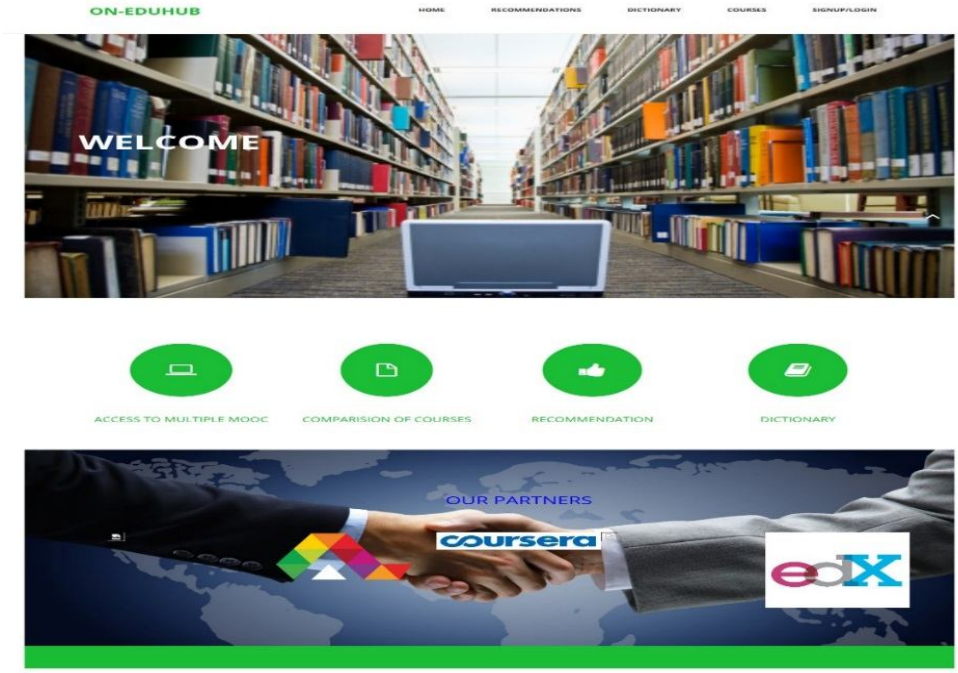


Fig. 6 Screenshot of Home Page

2) Course Page: It is headed by a search bar which includes all and name of categories of courses present in course page .it is headed by a navbar which contains links to course "Providers", "Subjects". "Providers" drops down into a list of Alison, Coursera and edX, and links when hovered over. Each of these pages is a paginated list of courses from their respective provider. "Subjects" leads to a table of links to our categories. Clicking on one of these retrieves a paginated list of courses in that category. The "Course

Spotlight" is an image carousel which features courses that might be popular with average users. Currently it is a RDF such as how many users have enrolled or completed a specific course. This will depend on the future availability of MOOC statistics via provider API or otherwise. The course images in this carousel are retrieved from mooc. The IDs for these images are taken from our RDF which collects links to introductory videos. Where an ID is not provided, a placeholder logo for the provider, such as the Coursera logo in the rightmost course, is shown. Hovering over a course prompts links to the On-EduHub course details page and the introductory video. The course title, provider, start date, as well as the beginning of the course summary are provided below the course images.

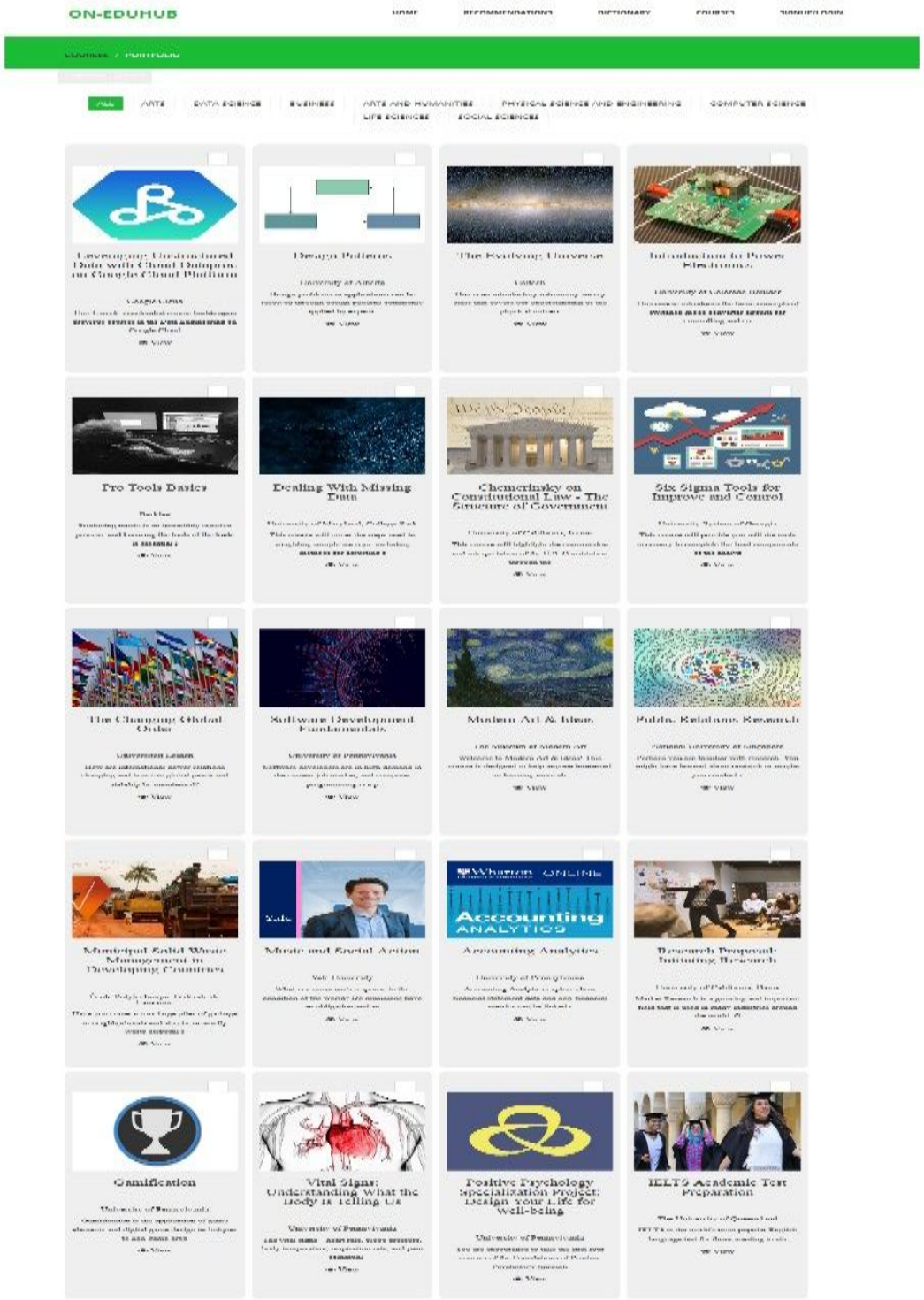


Fig. 7 Screenshot of Course Page.

3) Comparison: In Fig. 8, the table for the comparison of courses is shown. Information pertaining to each course is displayed as an "accordion" in which clicking on a property opens the field for each course being compared allowing for simple detail-by- detail comparison. Courses are once again filterable by course provider and hovering over the course image brings up a link to its corresponding OnEduHub "course details" page as well as a link to enroll in the course.

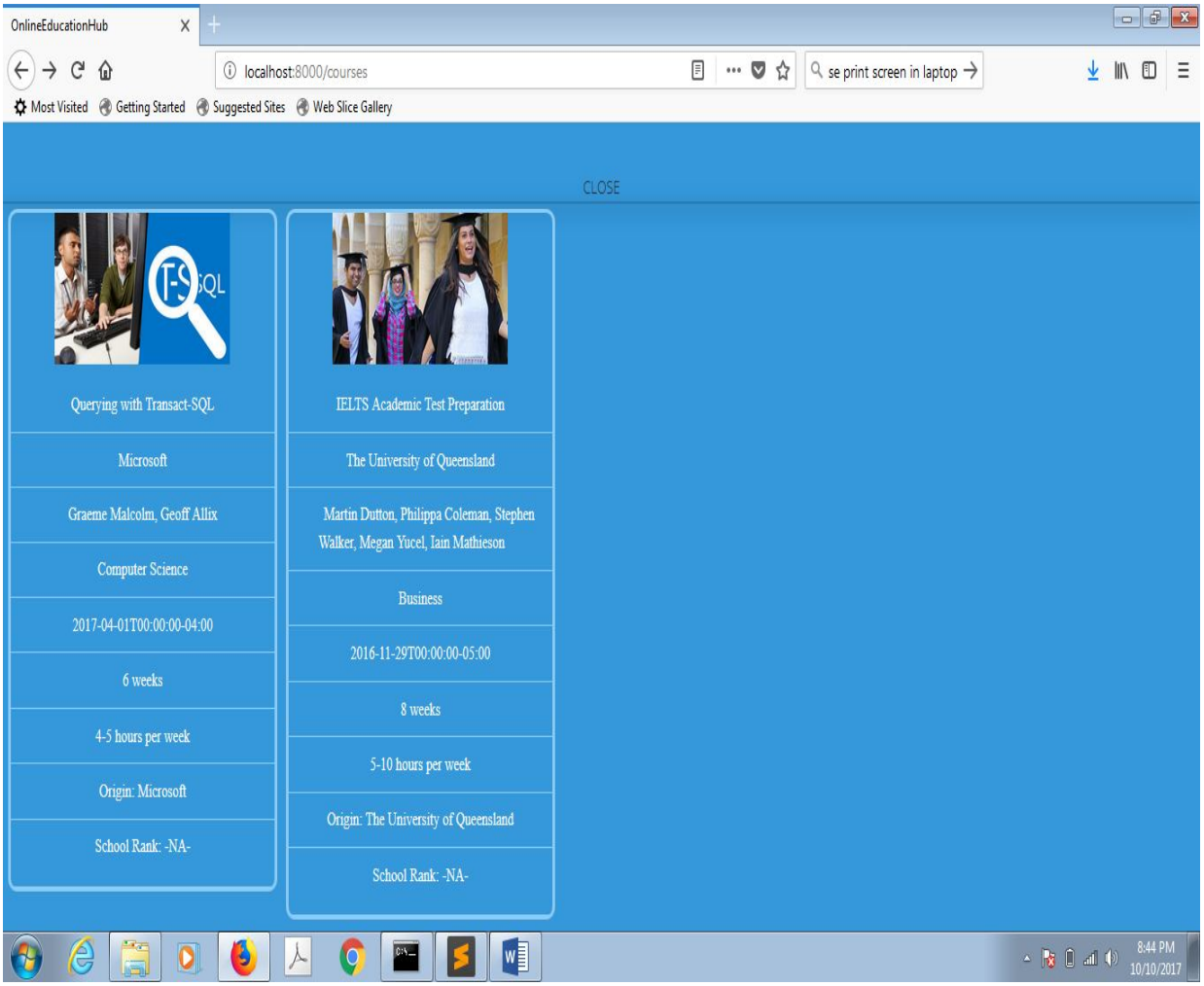


Fig. 8 Screenshot of Comparison Table

5) Recommendation: Recommendation tab will be displaying a page where the current user logged in will be shown some links of recommended courses based on the courses viewed by the current user. These recommended courses will be selected on the basis of a list of common courses surfed by the user and other past users. The list will be sorted on the basis of most visited courses by the common users in descending order. The Page will have some tabs on the top to display only the specific stream courses form the recommendation list. The User is searched for reference of the current logged in user from the session based on the attribute value "User-id" which is unique for every user.

1) "SELECT page_url FROM page_visits WHERE user_id =:user_id1" visited by the current logged in user.

2) "SELECT page_url, COUNT (*) as page Counts FROM page_visits WHERE user_id IN (SELECT DISTINCT user_id FROM page_visits WHERE page_url IN (SELECT page_url FROM page_visits WHERE user_id = :user_id1) AND user_id != :user_id2) AND page_url NOT IN (SELECT page_url FROM page_visits WHERE user_id = :user_id3) GROUP BY page_url ORDER BY page Counts DESC "

Example:

If the User A is logged in currently and visited courses 1, 2, 4 and 5. User B had visited courses 1,3,4,7. User C had visited courses 2,3,4,5,9,11. User D had visited course 5, 7.User A will be recommended courses: 3, 7,9,11.

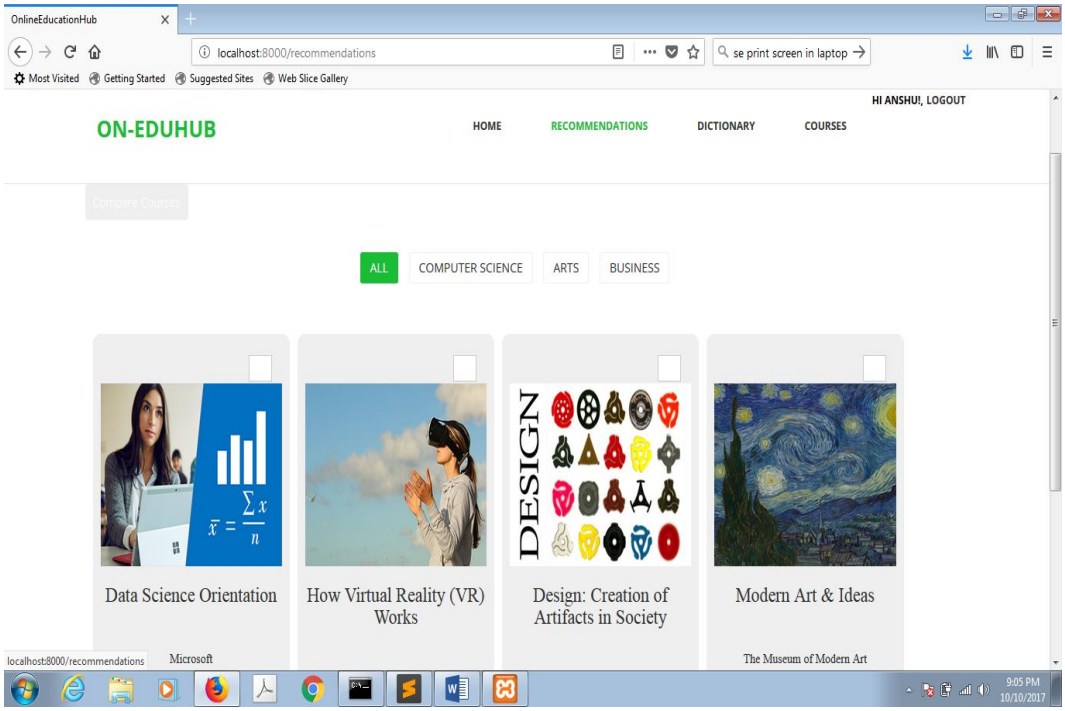


Fig. 9 Screenshot of recommendation

6) *Dictionary:* This Page contains all the courses Grouped according to the initial Alphabet of the course name. Top of the page has all the alphabets which are selectable. The Alphabet Selected will make all the courses filtered accordingly. All the courses contains view tab that links to the course page on the respective website. If an alphabet is selected and the list does not have any course name starting with it, the page will have no results. Data is collected from the XML File i.e. the RDF Structure as raw data. This Raw data is converted into a graph by XML Parser. On selecting an Alphabet from the List, the Graph makes it easier to retrieve courses starting from the alphabet

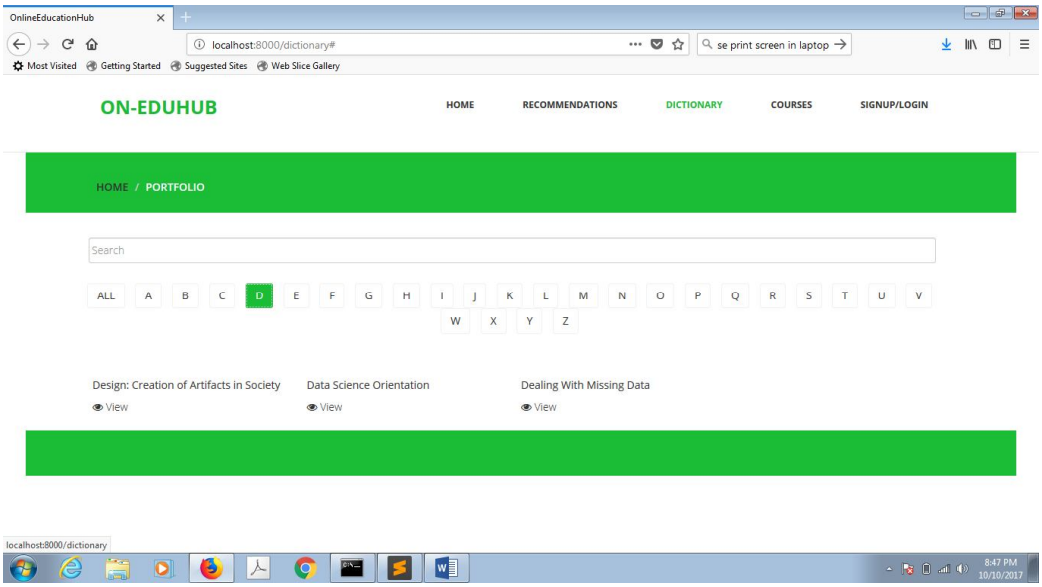


Fig.10 Screenshot of Dictionary

7) *Signup/Login:* We have created this Signup/Login page to keep track of student's activity i.e. which course they are studying or which course is most preferable, we are storing all this data in MYSQL database so that this data will help us to keep record of student and their preferred courses , this data helps us for recommending courses to users.

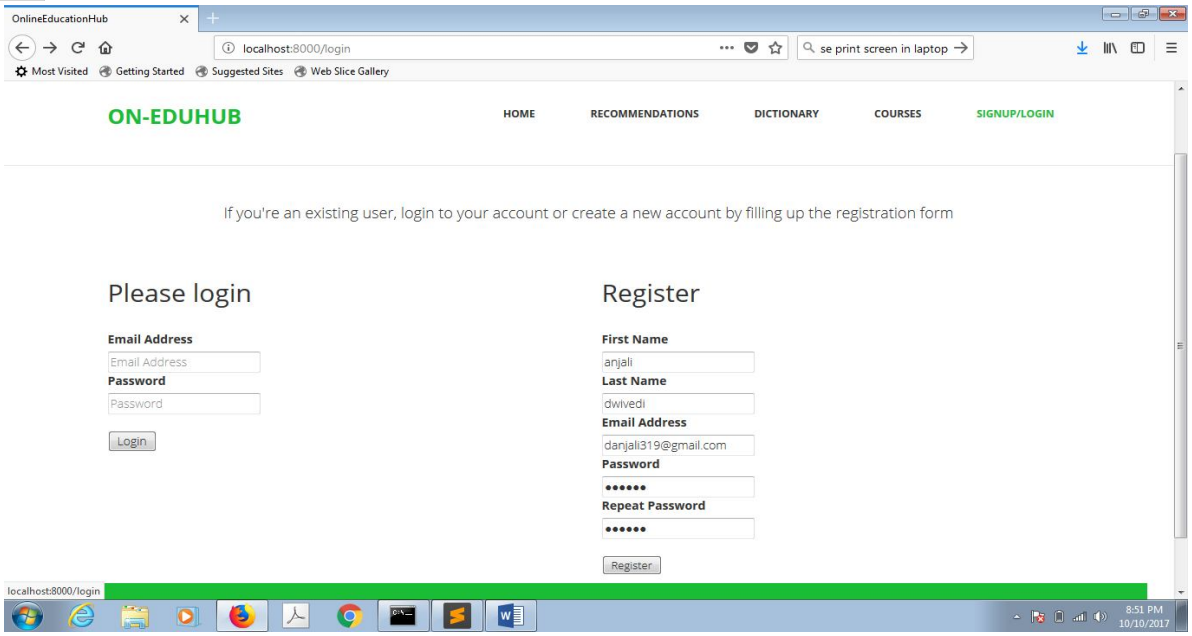


Fig 11 Screenshot of Signup/Login page

VI. SIMULATION ENVIRONMENT

A computer with minimum configuration of Windows 7 with 4 GB RAM and software like SPARQL, XML, APACHE TOMCAT SERVER, XAAMP server, SCRAPPY TOOL, JSON needs to be installed on the computer for the project work.

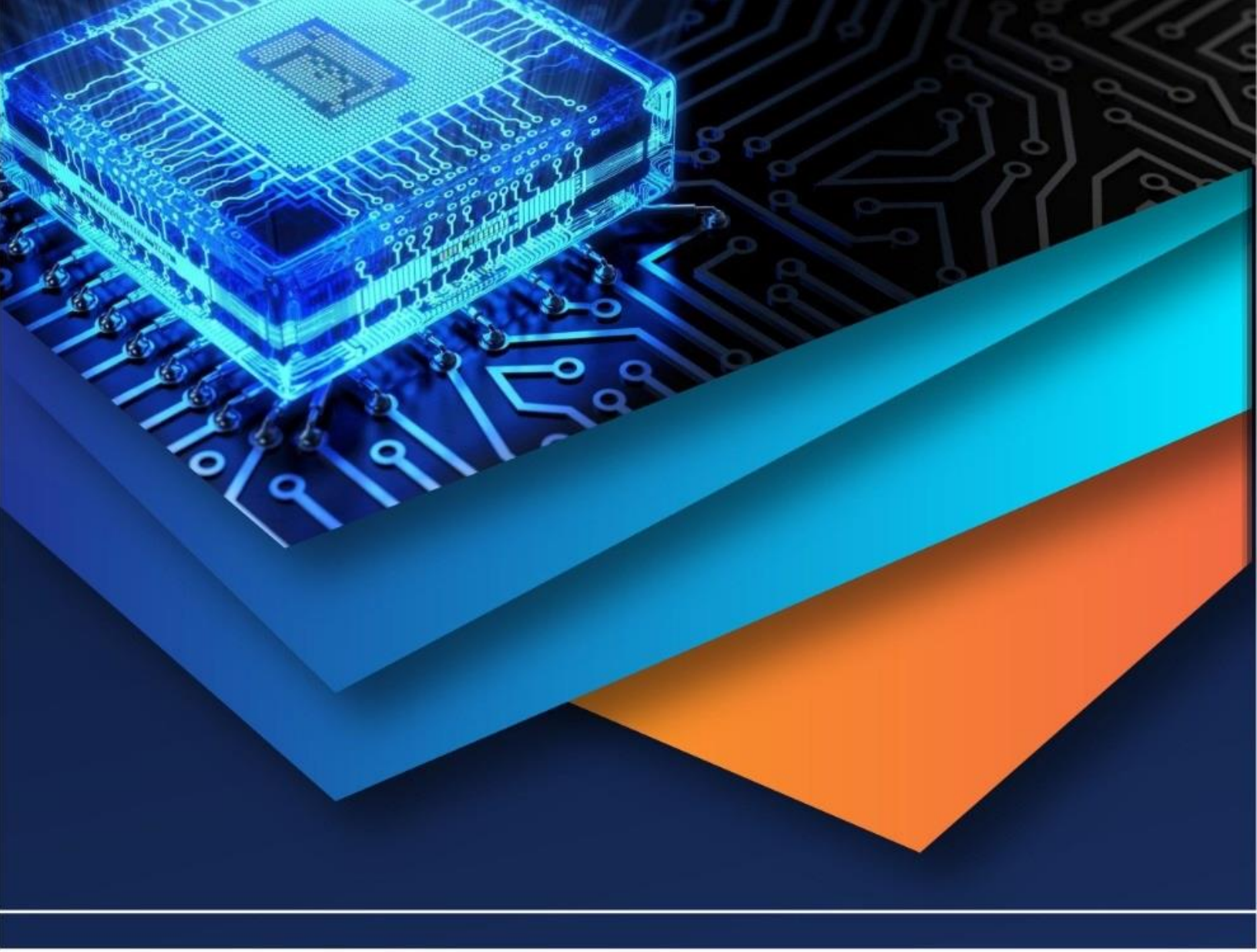
VII. CONCLUSION/FUTURE WORK

We have presented a Linked Data vocabulary, which incorporates domain knowledge i.e. types and properties found in online courses. The ontology allows for assignment of data properties relevant to MOOCs as well as object properties which link MOOC categories, sessions, and instructor and course details together. Also presented is our approach to collecting and generating Linked Data from three MOOC providers: Alison, Coursera, edX, and Udacity. Using Coursera's API and two Scrapy crawlers for edX and Alison, we collect MOOC data in JSON and convert it to RDF with Apache Jena. We describe implementation of OnEduHub, a web application which utilizes the Linked MOOC Data to allow users to search/discover, compare and recommend online courses. We have used Python based micro-MVC framework called Flask for the web application. We have also used a few external dependencies like SQLAlchemy to have an ORM wrapper on top of MySQL Database. Our primary source is RDF xml for the courses content but the recommendation data is stored in the MySQL database. We have routes like /courses, /login and /recommendations to get the relevant data. The functionality as well as the look and feel of the application are highlighted with output screenshots. Our future work will focus on: automating website updates, incorporating demographic data, reviews, developing an item pipeline for our crawlers, summaries for more robust data and search and natural language processing of syllabi.

REFERENCES

[1] Linked Data - The Story So Far Christian Bizer, FreieUniversität Berlin, Germany Tom Heath, Talis Information Ltd, United Kingdom Tim Berners-Lee, Massachusetts Institute of Technology, USA. Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS). http://linkeddata.org/docs/ijswis-special-issue (2011)
[2] ScientificAmerican:TheSemanticWebhttp://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 May 17, 2001
[3] Role of Ontology in Semantic Web DESIDOC Journal of Library & Information Technology, Vol. 31, No. 2, March 2011, pp. 116-120
[4] Massive Open Online Courses Tharindu Rekha Liyanagunawardena School of Systems Engineering, University of Reading, Whiteknights, Reading, RG66AY, UK;Academic Editor: Satoshi P. Watanabe Received: 13 November 2014 Published: 28 January 2015
[5] Pappano, L. (2012). The Year of the MOOe. The New York Times, 2(12),2012. Chicago
[6] Coursera Catalog API. (n.d.). - Cow'sera Technology. Retrieved July 22, 2017, from https:lltech.coursera.org/app-platform/catalog/
[7] Klyne, G., & Carroll, J. J. (2006). Resource description framework (RDF): Concepts and abstract syntax.
[8] X. H. Zhang and G. S. Ying, “Ontology-based semantic retrieval system,” in Proc. International Conference on Wireless Communication, Networking and Mobile Computing, pp. 1-4, October 12-14, 2008
[9] W. D. Fang, L. Zhang, Y. X. Wang, and S. B. Dong, “Towards a semantic search engine based on ontologies,” in Proc. of the Fourth International Conference on Machine Learning and Cybernetics, Guang Zhou, vol. 3, pp. 1913-1918, 18-21 August 2009

[10] An Enhanced Page Rank Algorithm over Domain , Preetibala Deshmukh ,Vikram Garg International Journal of Computer Applications (0975 – 8887) Volume 139 – No.1, April 2016
[11] https://www.csee.umbc.edu/courses/771/papers/ieeeIntelligentSystems/ontologyLearning for Semantic web, Alexander Maedche and Steffen Staab, University of Karlsruhe
[12] X. H. Zhang and G. S. Ying, "Ontology-based semantic retrieval system," in Proc. International Conference on Wireless Communication, Networking and Mobile Computing, pp. 1-4, October 12-14, 2008.
[13] https://github.com/RDFLib/rdflib
[14] proceedings of the 20 IS IEEE 9th International Conference on Semantic Computing (IEEE ICSC 20 IS) MOOCLink: Building and Utilizing Linked Data from Massive Open Online Courses Sebastian Kagemann Indiana University School of Informatics and ComputingBloomington, IN 47405, USAsakagema@indana.edu Srividya BansalArizona State UniversitySchool of Computing, Informatics, and Decision SystemsEngineering, Mesa AZ 85212, USAsrividya.bansal@asu.edu



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY