[Xinyang Zhang:9743946876 Yuxin Liu:2943825078]
[11/30/2018]

Final Report
A Cloud-based E-Books Metadata Extraction & Search Web App

1 Project Idea

In our project, we develop a web app which users could search eBooks on it by keywords and selecting checkboxes (faceted search). After searching, a collection of metadata and download link of results will be shown on the web page.
There are two components of this project. The first one is a program project to extract metadata from PDF documents and automatically upload metadata to Firebase.
The second one is a cloud-based search web app which is for searching E-books by keywords and facets. In addition, the searched eBooks can be downloaded via the download link shown within the results.
The main purpose of this project is to make students especially the college student finds the textbooks freely and more conveniently. The textbooks come from the www.bookboon.com which cover more than 8 majors, including Accounting, Data Analysis, Engineering, Economics, Languages, Natural Sciences, Statistics and Mathematics, Strategy and Management etc.
This project was designed to extract metadata from PDF documents like title, edition, publication date, author, press, ISBN, and total page number.
The PyPDF2 library was used for extracting metadata from PDF in a utility program. After extraction, the metadata were automatically uploaded to Firebase by Firebase REST API.
As for the programming languages and software libraries, Python, PyPDF2, requests (Firebase REST API), regex were used for extraction and uploading part. Python, JavaScript, jQuery, Flask were used for the web app development and front-end interaction.

2 Description of Documents and Metadata

The documents of our project are eBooks (PDF Files). The E-Books were downloaded from www.bookboon.com which provides free eBooks and textbooks. It covers more than 8 majors of textbooks. We download 5 majors of those eBooks (101 eBooks) to support our web app.
As Fig2.1, our books contain covers, titles and contents. However, only some of the metadata we extract are from the info of PDF Files which could be extracted directly from PyPDF2 package. The rest of metadata should be extracted from the content of each book.

Fig. 2. 1

In the first two to three page contains the information of the books which has edition, ISBN, author, creation year. So, we program another function to extract the rest metadata.

Budgeting: Planning for Success – Budgeting and Decision Making
1st edition

ISBN 978-87-7681-574-5

Fig.2.2

3 Metadata Extraction and Uploading

There will be three parts of the whole implement procedure. There are metadata extraction, inverted index creation, metadata and index uploading.

3.1 Metadata Extraction

As we mentioned before, we made two functions to extract the metadata. The first one is used to extract metadata from the info of books by PyPDF2 package. An example of info is shown in Fig 3.1.1. We could get title, authors, page numbers from info.

More Info:

Last opened:	Nov 28, 2018 at 4:22 PM
Title:	Working Capital and Debtor Management: Exercises
Authors:	Robert Alan Hill
Version:	1.7
Pages:	87
Resolution:	595×842
Security:	Password Encrypted
Content Creator:	Bookboon.com Ltd.
Encoding software:	Adobe PDF Library 10.0.1; modified by SetaPDF-FormFiller Component v2.28.0.1223 ©Setasign 2005-2018 (www.setasign.com)

Fig.3.1.1

A screenshot of python program is shown in Fig 3.1.2. We define a get_metadata function to extract metadata by PdfFileReader function of PyPDF2 package. And we create a dictionary to store all the metadata.

```python
from PyPDF2 import PdfFileReader
import regex as re

#define get_metadata function
def get_metadata(name):

    #use PdfFileReader function to read a pdf file
    pdf = PdfFileReader(name)

    #decrypt pdf file
    if pdf.isEncrypted:
        pdf.decrypt('')

    #get the info content of pdf file
    info = pdf.getDocumentInfo()

    #get the page number,author and title
    number_of_pages = pdf.getNumPages()
    author_lower = info.author.lower()
    title_lower = info.title.lower()

    #initial a dictionary to store metadata
    dic_meta = {}
    dic_meta.update({'Author':author_lower})
    dic_meta.update({'Title':title_lower})
    dic_meta.update({'Page_Number':number_of_pages})

    return dic_meta
```

```python
edition_dic = {}
#define get extra metadata function
def get_edition1(i):
    #use PdfFileReader function to get
    #the content of pdf file
    pdf = PdfFileReader(i)

    #decrypt pdf file
    if pdf.isEncrypted:
        pdf.decrypt('')

    list_me = []

    #get the content of page 2 and 3
    for j in (1,2):
        #get the content as txt
        pageObj = pdf.getPage(j)
        text = pageObj.extractText()

        #use regular expression to match creation year,
        #edition and ISBN
        year = re.search(r'© \d\d\d\d', text)
        if year :
            edition_dic.update({'Creation_Year':year.group()[2:6]})
        edition = re.search(r'\d*\w\w edition', text)
        if edition:
            edition_dic.update({'Edition':edition.group()})
        match = re.search(r'ISBN [\d*-]*\d*', text)
        if match :
            edition_dic.update({'ISBN':match.group()[5:]})
    return edition_dic
```

Fig.3.1.2

[Xinyang Zhang:9743946876 Yuxin Liu:2943825078]
[11/30/2018]
For the rest of the metadata, we made another function called get_edition1 which is shown in Fig 3.1.2. In this function, we first get the content of page 2 and 3, and use regular expression to match creation year, edition and ISBN.
Finally, we get nearly all the metadata of a eBook.

3.2 Inverted Index Creation

After collecting all the metadata, we create inverted index of author, title, year, ISBN and category. To imply this, we create unique id for each book from 0 to 100.
To better prepare for the search function, we split title and author into words and store id list under different words. Inverted index also be created for ISBN, year and category which are shown in Fig 3.2.1.

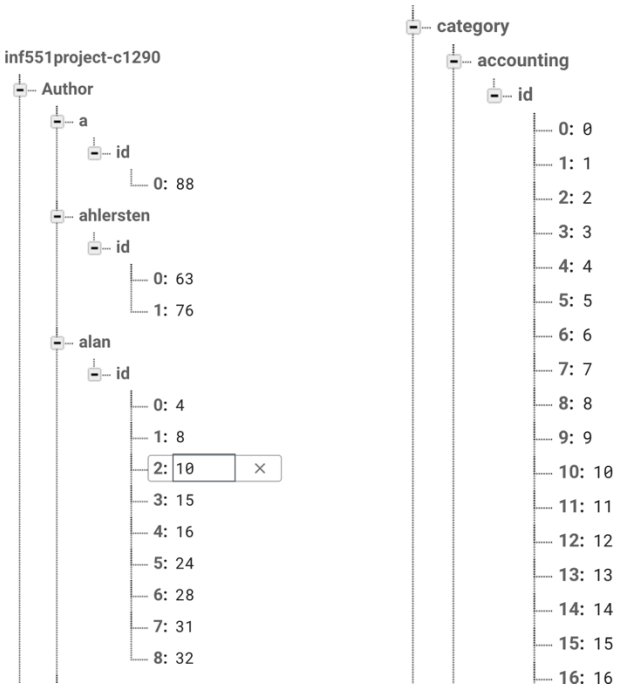


Fig.3.2.1

3.3 Metadata Uploading

When we upload pdf files to the Firebase Storage, we got the attributes which are the download link of each books. We store them along with id and metadata. Here is the screenshot of all the metadata which stored under “book” in Fig 3.3.1.

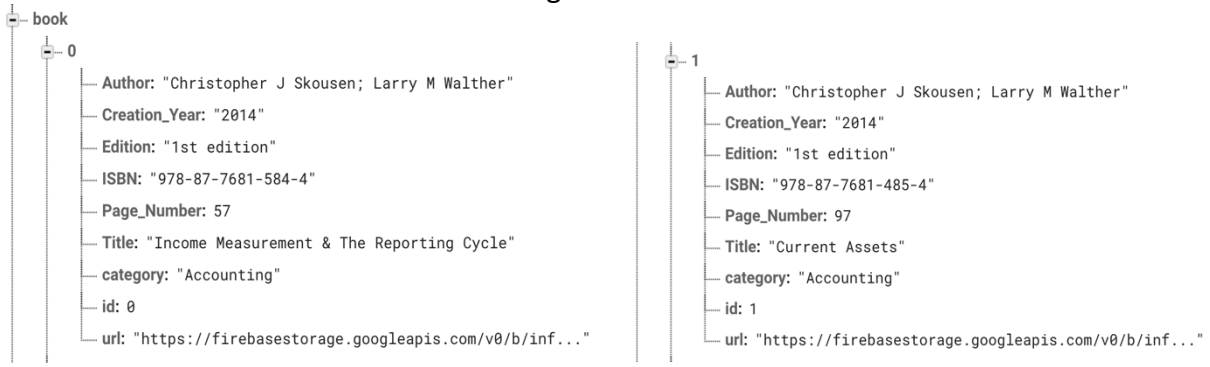


Fig.3.3.1

4 UI Design

For the User Interface Design, we use the html to design the website on the DreamWeaver. To simplify the search operation, we combine two search function at the same location which is the keyword search and the CheckBox search. The main body of our website is the result of the search which shows as the table like format. When user moves the mouse on the result which would change the color to highlight the result. What's more, to show the result more intuitively, we present the number of the result which the user search on the top of the result. Fig.4 shows the navigation part of our website and the result part of our website. The Categories part is a filter which based on the result from the keyword search, which means that when users search the keyword above, based the result the categories part's content would be changed. We choose 3 kinds of metadata and show the top 3 of each metadata which based on the result the keyword results return.

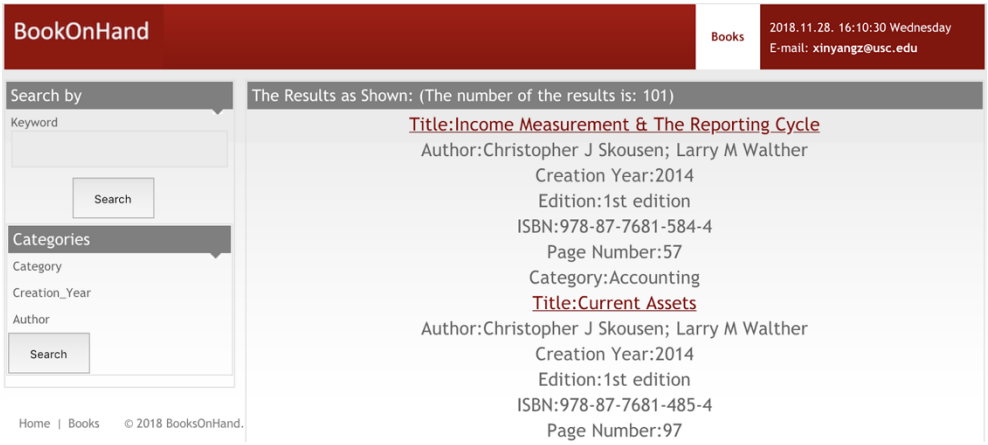


Fig.4

5 Search Function

5.1 Keyword Search Function Description

For the keyword search, we allow users to type many words which separate by spaces which just like you search in google, only need to type words without any symbol or any accurate information, just type what you know is enough. For example, if you want to find a ebook, but you merely know some 'david' wrote. In our website, you can just type it into and search the book you want to find which shown in Fig.5.1.

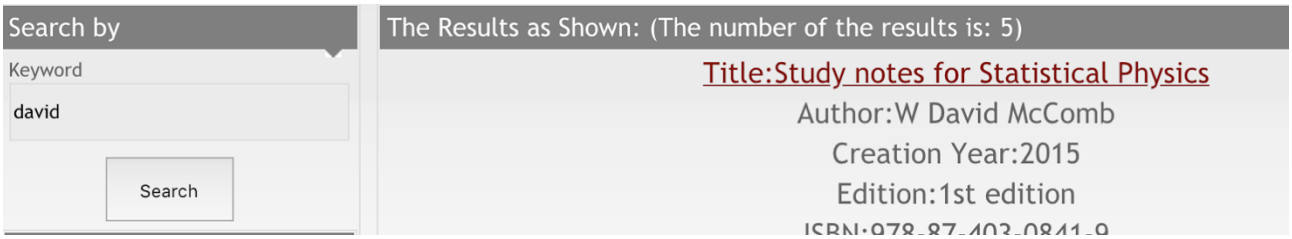


Fig.5.1

5.2 Keyword Search Function Implementation

We search all the results which stored in the firebase, we make the index for each metadata category which means the word you type may belong to any category. Thus, every word from the blank the user just type would transfer form front stage to the end-stage, and become the url link to firebase, and the get the index of each book. Finally, we get the list of the index and then find specific information in the 'book' which store the whole metadata.

[Xinyang Zhang:9743946876 Yuxin Liu:2943825078]
[11/30/2018]

```
if keyword != None and keyword!='':
    count3 =1
    keyword_input = keyword.lower()
    keyword_input_split = keyword_input.split(' ')
    # print(keyword_input_split)
    mov_keysli = keyword_input.replace('-','')
    count_split_keyword = len(keyword_input_split)
    # print(count_split_keyword)
    idlist_split_keyword = []
    for i in keyword_input_split:
        caturl = link[:-9]+'category/'+ i +'.json'
        titleurl = link[:-9]+'Title/'+ i+'.json'
        authorurl = link[:-9]+'Author/'+ i + '.json'
        yearurl = link[:-9]+'year/'+i + '.json'
        ISBNurl = link[:-9]+'ISBN/'+mov_keysli + '.json'
```

```
for i in return_list:
    resulturl = link[:-5]+'/'+str(i)+'.json'
    temp1 = requests.get(resulturl).json()
    result_pdf.append(temp1)
```

Fig.5.2

5.3 Sidebar Search Function Description

After you make the fuzzy query, keyword search, you may want to the result become more accurate. You can use the checkbox search to filter the result you just get. When you type david, you can get 5 books. Then, you can choose the checkbox to filter the result which you just get. For example, you can click the 2014 to return the books which are all from 2014 and David which shown in the Fig.5.3.

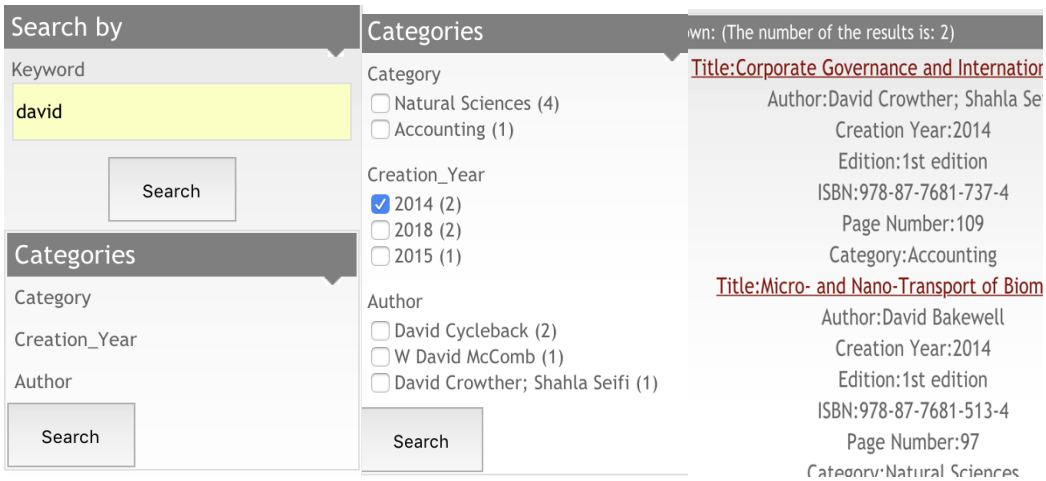


Fig.5.3

5.4 Sidebar Search Function Implementation

Each sidebar's information is based on the keyword search's results. We list top 3 of each category to do the filter and for every keyword search, the checkbox content and the number of the checkbox content would be updated. We implement the method like below Fig.5.4.

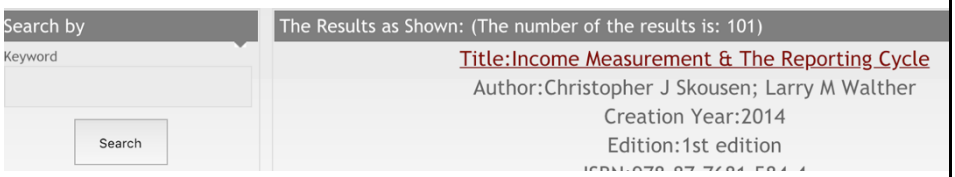
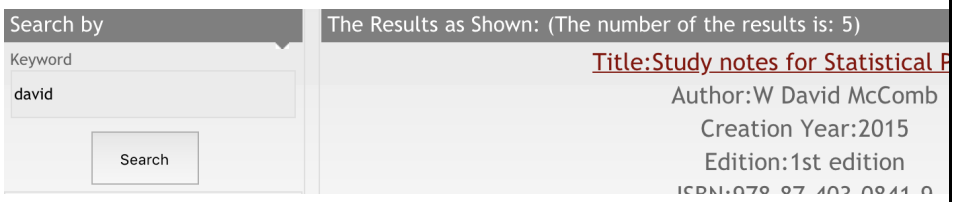
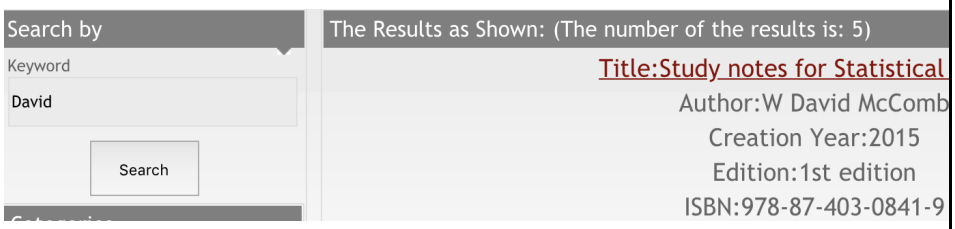
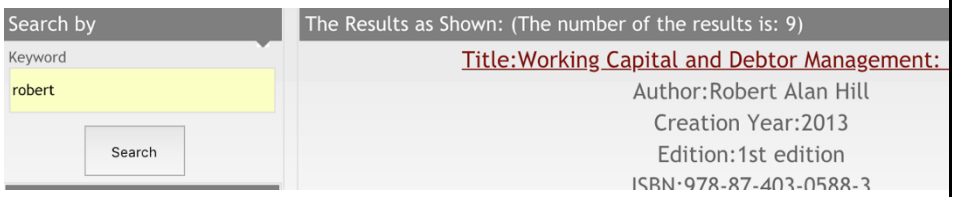
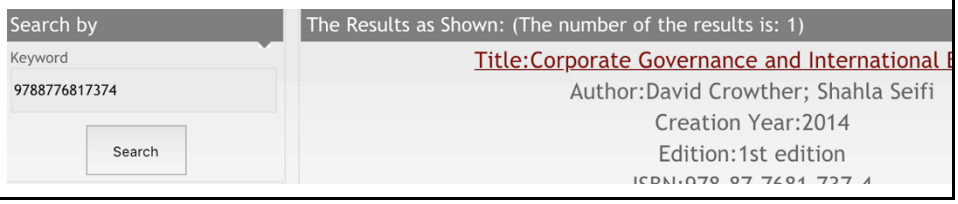
```
$('.rlt-key').remove()
var author_sum_list = metadata.new_author_sum_list
for (i in author_sum_list){
    var author = author_sum_list[i][0]
    var cnt = author_sum_list[i][1]

    var item_author = $("<li class='rlt-key'></li>")
    var item1 = $("<div><input type = 'checkbox' clas
    // var item1 = $("<type = "checkbox">"+author+' '

    item_author.append(item1)
    $('.resultkey3').append(item_author)
}
```

```
$('.check_value').each(function(i,e){
    var isChecked = $(this).prop('checked');
    if (isChecked == true){
        var sibtext = $(this).parents('.rlt-key').
        var name = $(this).parents('.box-content')
        if(name=="category"){
            cdata.push(sibtext)
        }
        if(name=="creation_year"){
            ydata.push(sibtext)
        }
        if(name=='author'){
            adata.push(sibtext)
        }
```

Fig.5.4

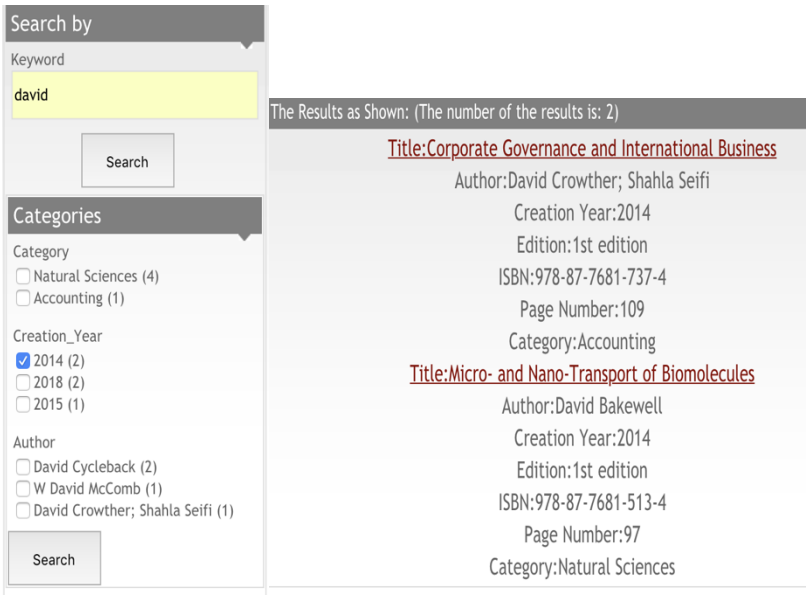
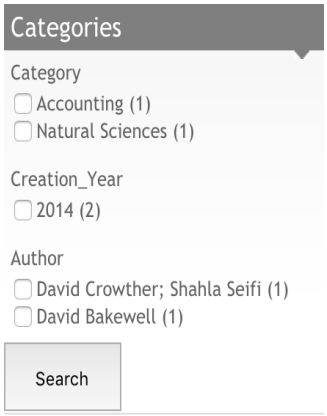
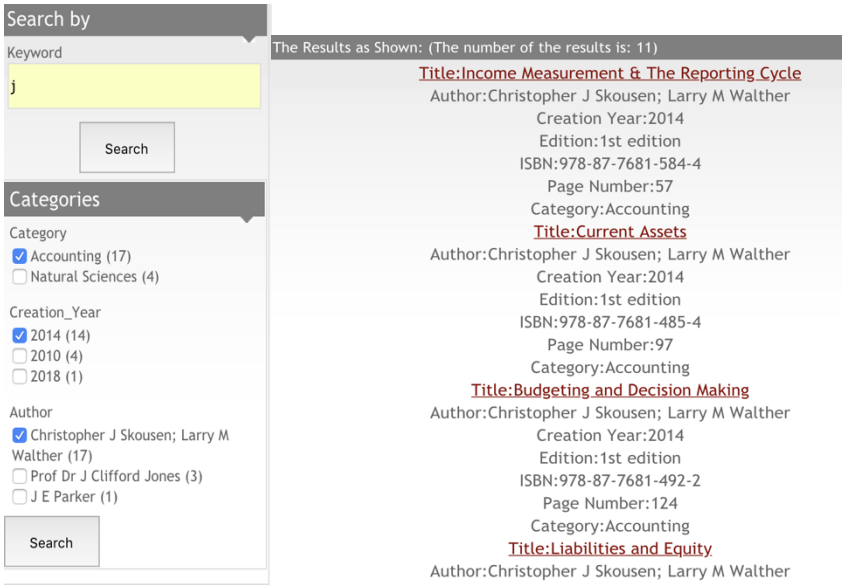
6 System Test

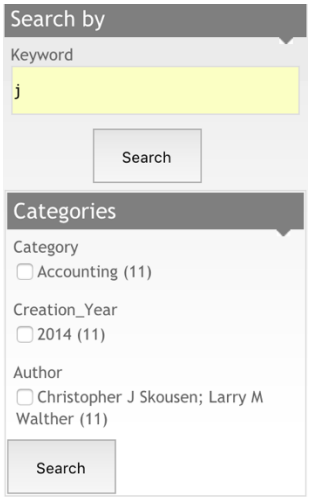
6.1 Example-Keyword

Keyword/Input	Output	Screenshot
(empty blank)	101	
david	5	
David	5	
david 2014	2	
robert	9	
978-87-7681-737-4	1	
9788776817374	1	

6.2 Example-Keyword+CheckBox

[Xinyang Zhang:9743946876 Yuxin Liu:2943825078]
[11/30/2018]

Keyword+CheckBox	Output	Screenshot
david + 2014	2	 
j+(Accounting+2014 +Christopher J Skousen; Larry M Walther)	11	

7 Group Formation

	Description of Work
Xinyang Zhang USC ID:9743946876	Webpage design and development, search function development, front end interaction
Yuxin Liu USC ID:2943825078	Metadata extraction and uploading program, search function development