

Project: Digital Library Search for Electronic Thesis and Dissertations
with a discussion forum.

Student Name: Sridivya Majeti

Student UIN : 01184044

1. Milestone Accomplishments:

My website is about Digital Library Search for Electronic Thesis and Dissertations. Using this website, we can search and retrieve information from remote databases of digitalized books. Users can sign up and login to search for the ETD's and Authenticated users can also download the pdfs documents and can add to their favorite lists. I have done this project using PHP, Html languages and Laravel Framework. I have also used Bootstrap, JavaScript, ajax, and CSS. So far, I have covered all the four milestone specifications. Below is the overview of my accomplished specifications:

Table 1: Overview of status for all Milestone specifications.

Fulfilled	S.No	Milestone	Description
Yes	1	1	The search page of my website is at landing page
Yes	2	1	Search Button next to the Search engine
Yes	3	1	Users can able to register new accounts using their Email address
Yes	4	1	Password will be encrypted before storing in database
Yes	5	1	Users cannot register duplicate accounts using same email address
Yes	6	1	Users should be able to login using their registered Email address and password
Yes	7	1	Users can be able to reset their passwords if they forget it
Yes	8	1	Users should be able to change their passwords if they want.

Yes	9	1	The website should have a homepage for each user, where they can view their profiles, change passwords, and update information.

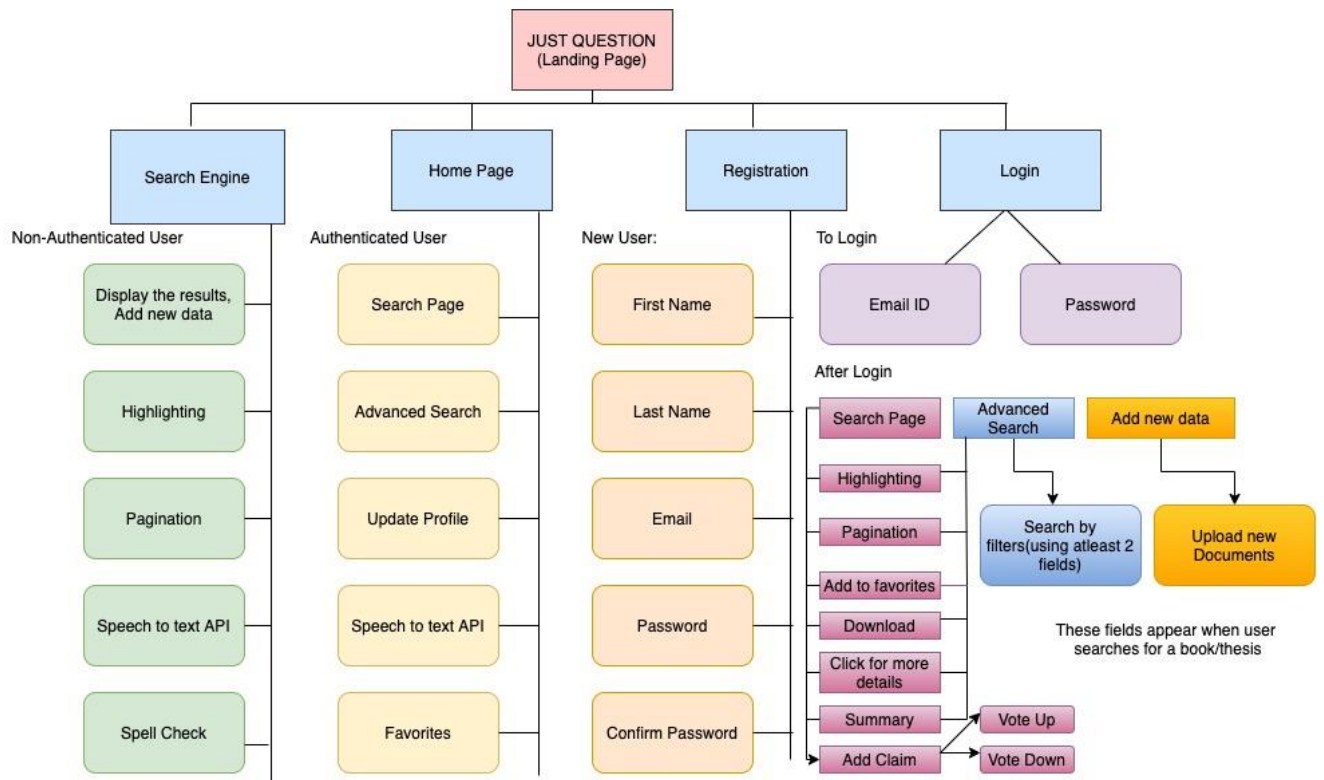
Yes	10	2	Users should be able to get a confirmation email to verify their email addresses for registration or Password reset
Partially Done	11	2	The website has an “Advanced Search” function so users can search by specifying values for at least two fields simultaneously.
Yes	12	2	The website should index at least 2000 “documents” (a document can be an ETD).
Yes	13	2	Users can query the search engine without logging in.
Yes	14	2	The search engine accepts a text query in the search box and return results on the search engine result page (SERP).
Yes	15	2	Each item in SERP should links to a page for a document.
Yes	16	2	The search engine should display the number of returned items on top of search results.
Yes	17	2	The search engine should display the number of returned items on SERP.
Yes	18	2	The SERP should contain a search box on the top.
Yes	19	3	The search engine should show the actual keywords after filtering on the SERP.

Yes	20	3	The search engine can prevent XSS vulnerability by removing tags existing in the query.
Yes	21	3	Users should be able to insert a new entry and search engine will index it.
Yes	22	3	The search engine can return paginated results.
Yes	23	3	The search engine should have a summary page for each thesis, the summary page should display at least the following 8 fields, when available: title, author, university, department, year, advisor, academic field (e.g., Computer Science), and abstract. Users can click each item on SERP and go to this summary page for each thesis.

Yes	24	3	User can switch back to the search results from the summary page.
Yes	25	3	In the summary page, below the metadata, there is a discussion board, where users can input a claim extracted from the thesis and add comments.
Yes	26	3	The search engine can highlight the matched terms on SERP.
Yes	27	4	Users can save items in search result to their profiles and Users can delete items from their favorite list.
Yes	28	4	Users have to login first to save search history to their profiles
Yes	29	4	Items in the favorite lists should be an ID followed by the title, year, and author linked to the summary page of the document.
Yes	30	4	Logged in users can “vote” a claim discussion
Yes	31	4	The search engine implements speech-to-text
Yes	32	4	The search engine implements spell check.
Yes	33	4	There is a button from which users can download the PDF from local storage from the summary page or from the SERP. For ETDs with multiple PDFs, the best solution is to zip them for download. But the basic requirement is to download one of them.

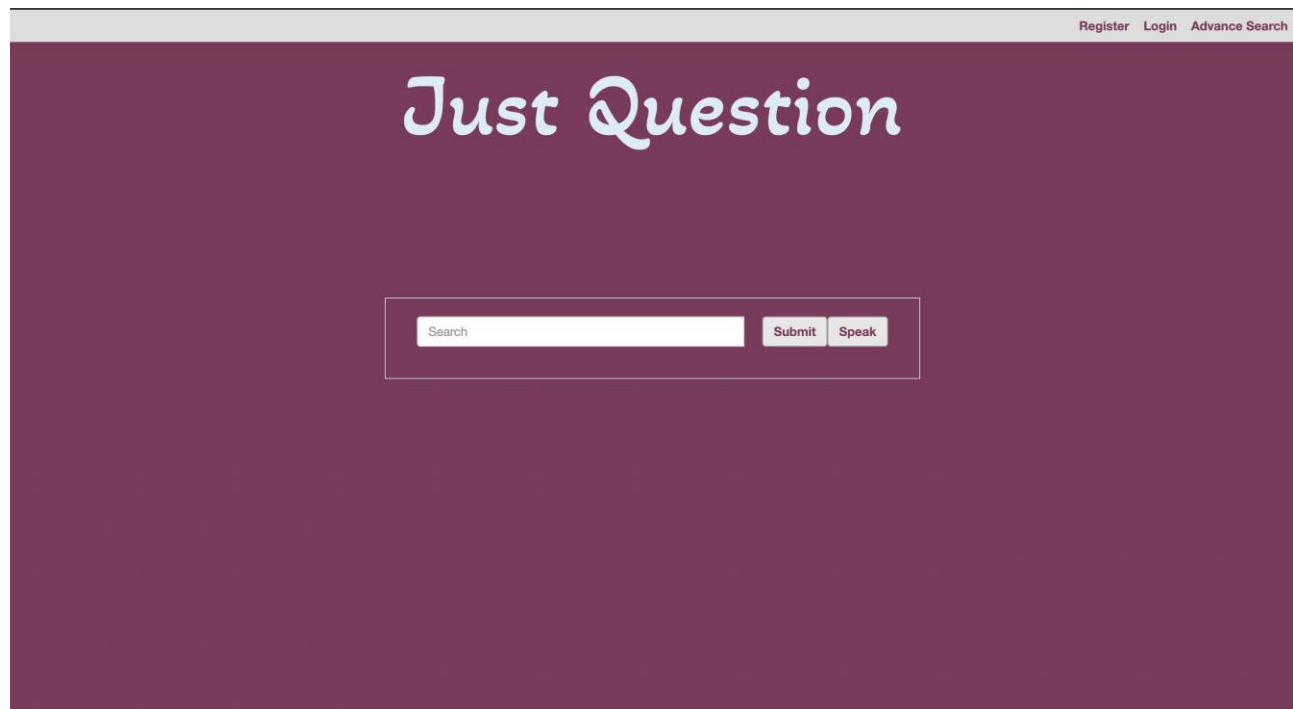
2. Website Architecture:

I have included both my frontend and backend design of the website in this section. Below is the outlier of my web architecture like how it appears on screen. It includes the web search engine, home, login, and signup forms etc. I have used Microsoft PowerPoint drawing tools to display the below web architecture. It describes the fields and shows what happens when we click on each field and where it redirects to.



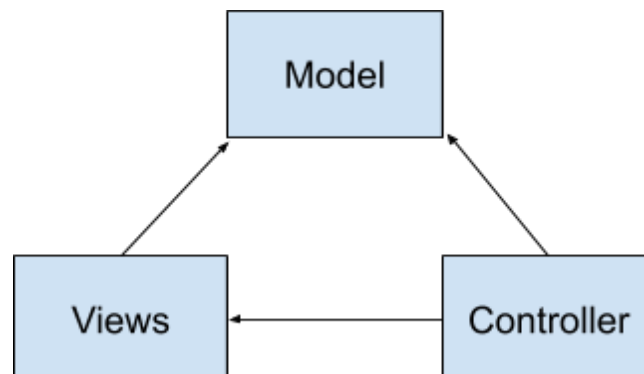
i) Frontend design:

I have used HTML, CSS, Bootstrap and JavaScript for my frontend design to make the website look like it does. I have created a search box, home page of my website, login and Register forms at the landing page. Users should register their details first to login. Also unauthorized users are also able to see the search results, download, summary options but they are not able to see the advanced search, add new data and some other fields. Once the users can login with their credentials, they can be able to see the advanced search, add new data, summary and add claim, vote up and vote down buttons for the claim. I have explained each functionality in the below implementation section. Below picture shows how my website frontend page looks like:



ii) Backend design:

The back end of a website consists of a server, an application, and a database. To do this, I have chosen PHP language and Laravel framework. Database creation in Laravel can be done by running the `php artisan make table` command, the table will be created and once we did changes in that DB table file, we can run `php artisan migrate` to migrate the table. Similarly, we can create controllers and models using the `php artisan make` commands. The controller is the one which connects to the DB, models, and returns to the views. This is a MVC(model,view,controller) architecture. The routes in the Laravel are stored in `web.php`. In the `web.php` we can see the view to be displayed using controller class or model methods.



3. Data:

Professor has given the dataset for Thesis and dissertations search engine. This dataset contains around 3973 folders. Each folder contains json files, text and pdf documents. We must use a for loop to obtain the fields from each json file. Each json file contains various fields related to books like handle number, title, author, department, degree grantor, degree level, degree name, description abstract, publisher, source URL etc. I have used curl command to index all the json documents data into the elastic search. The content of the file is in projectdata.php. Then I run the view of the data, it will go into our elastic search. I have written a query for my search functionality when we search for a word in a search engine the results should be displayed using multiple fields like title, degree name, author, abstract, publisher, type, degree level and contributor department. It will display the word which is there in all the above fields. Similarly, I have implemented the advanced search here I have written a query which multi matches with the any of the two fields of title and author.

4. Implementation:

4.1 Search box and search button:

There is a search box at the landing page of my website and when a user searches for a word it redirects to the SERP page where the user finds the summary and download button and advanced search is also available it is accessible to the user only after they login. In background, the code written for the elastic search in projectdata.php helped in indexing the data and the query written to implement the search functionality helps to redirect the user to SERP page. It also has a speak button besides a search button. The speak button implements speech to text recognition. You can find the code in Resources->views->pages->index.blade.php

4.2 User Account Registration:

The user can sign up and register themselves. The data is directed to database once the user signs up. The data like first name, last name, email id is stored in the database. Once the user registers they get a confirmation mail to their mail and it directs them to the login page. In the controller a method is designed which implements the functionality of checking if the user already exists by checking for the email id in database. You can find the code in Controllers -> Auth folder -> Main Controller->process_signup() and Resources->views->pages->register.blade.php

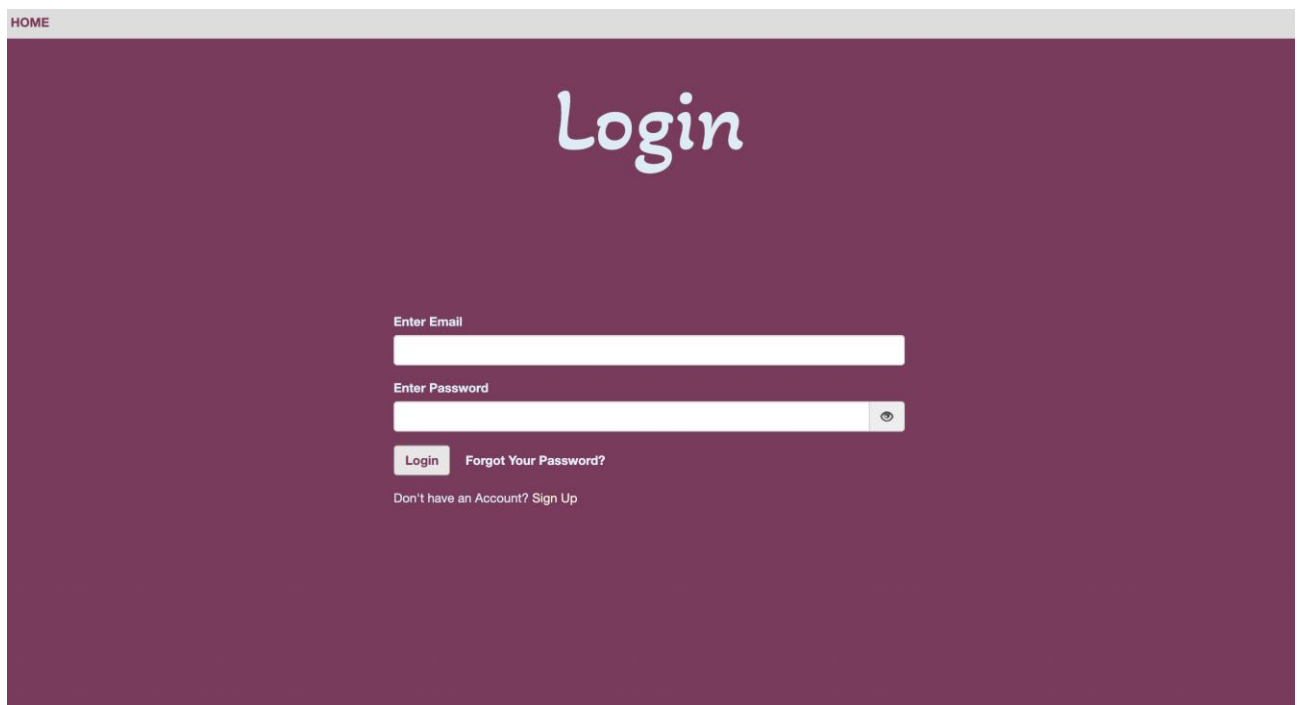
4.3 User Account Login:

The authenticated users can login using their registered mail id. The code is written in main controller where the `checklogin()` method is designed to implement the functionality of authenticating the registered users and `successlogin()` method directs user to the home page of the website where they see options like update profile, logout, search box, advanced search feature etc. You can find the code in

Resources->views->pages->login.blade.php

Controllers -> Auth folder -> MainController->`checklogin()`

Controllers -> Auth folder -> MainController->`successlogin()`



HOME

Login

Enter Email

Enter Password

[Forgot Your Password?](#)

[Don't have an Account? Sign Up](#)

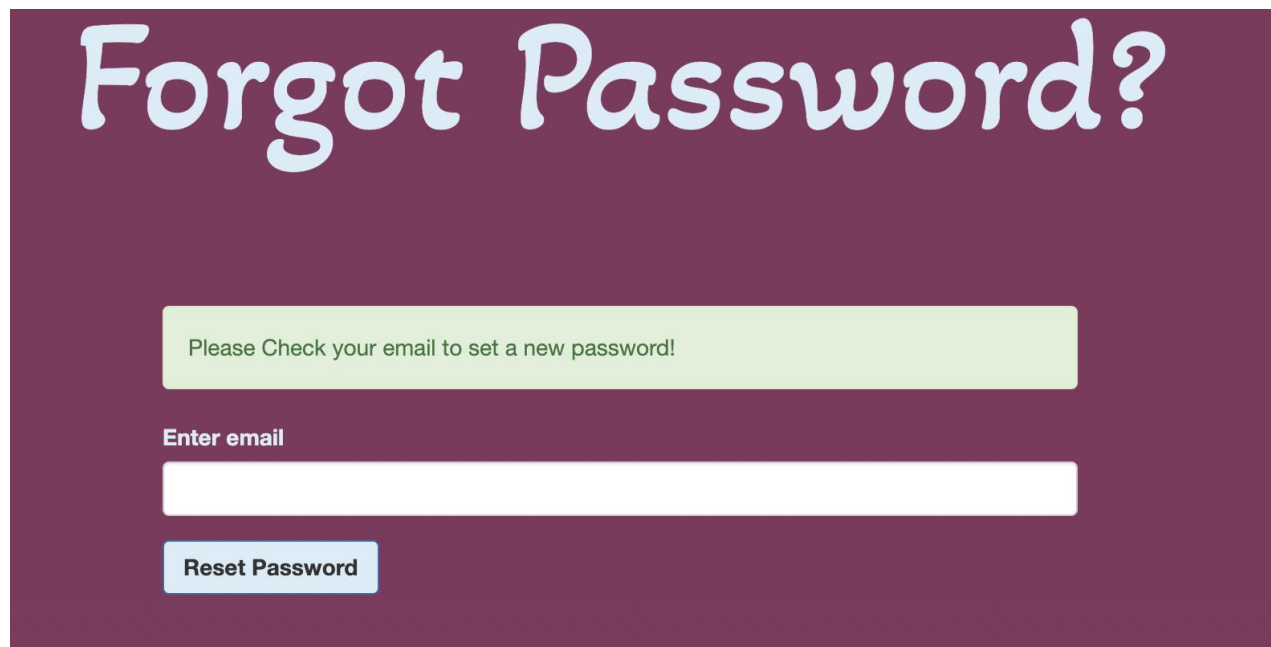
4.4 Password reset (Forgot Password):

The users can update or reset their password in case if they forget it .If the users ~~must~~ reset their password a confirmation mail will be sent to their email when they click on forgot my password button. In the received mail, they must click the link provided and they will be redirected to forgot password view and they can add their new password and that automatically gets updated in the database. Then the user can login using their new password. forgot_password method is written in Main Controller where the user mail id is verified and can change their password accordingly. setnewpassword is another method in main controller which takes the users' mail id as parameter and thus authenticated user can update his password. You can find the code in

Controllers -> Auth folder -> Main Controller-> forgot_password

Controllers -> Auth folder -> Main Controller-> setnewpassword
and in

resources -> views-> pages -> forgotpassword.blade.php



The image shows a web form titled "Forgot Password?" in a large, white, stylized font on a dark purple background. Below the title, there is a light green rectangular box containing the text "Please Check your email to set a new password!". Underneath this box, the text "Enter email" is displayed in a small, white font. Below the text is a long, white rectangular input field for the email address. At the bottom of the form, there is a light blue rectangular button with the text "Reset Password" in a dark blue font.

Set Password

New Password

Confirm Password

Your password must be 6-20 characters long, can contain letters and numbers, and must not contain spaces, special characters, or emoji.

SET PASSWORD

Login

4.5 Users' homepage:

Authenticated users i.e., the users who logged in can see the home page. Once the user logs in, there is a window which says “welcome (users' first name)” and few buttons on the right side of the box which are update profile, log out. Home page contains the search engine with search and speak buttons besides the search box, Advanced search, and favorites fields.

The code for this homepage is in successlogin.blade.php file.

Favorites Update Profile Logout

Just Question

A place to share knowledge and better understand the world

Welcome Sridivya!!

Search Submit Speak

Advanced Search

4.6 Main search function:

I have used two views to implement the search functionality one-for the logged in users and other for the ones who have not logged in because the authenticated users have enhanced features. Therefore, search functionality works for the people who are not logged in to the website and as well as to the authenticated users. The key components that connect the front end of my search engine to Elasticsearch are the various functions that are designed to implement the search functionality by using functions like search and variables like \$GET. Further, to implement the search function, I have used various Elasticsearch functions like index (gives the index name project data), type (document), multi match which takes various fields at once and implements the search, fields (what are the specified fields). You can find this code in

Controllers -> Auth folder -> Main Controller-> search

Controllers -> Auth folder -> Main Controller-> loginsearch

And in

Resources->views->pages->index.blade.php

Resources->views->pages->successlogin.blade.php

Search

Submit Speak

Advanced Search

Add new Data

Total results found: 185

Searched for: machine

Show 10 entries

Search:

Download

Title

A Comparison of Parents Who Initiated Due Process Hearings and Complaints in Maine

Download

Click for more details

Summary

A descriptive study of assigned and unassigned mentoring relationships of first year special education administrators in Virginia

Download

Click for more details

Summary

SERP.BLADE.PHP

Total results found: 185

Searched for: machine

Show 10 entries

Title	<input type="button" value="Download"/>
A Comparison of Parents Who Initiated Due Process Hearings and Complaints in Maine	<input type="button" value="Download"/>
<input type="button" value="Click for more details"/>	
<input type="button" value="Summary"/>	
<input type="button" value="AddFavorite"/>	
A descriptive study of assigned and unassigned mentoring relationships of first year special education administrators in Virginia	<input type="button" value="Download"/>
<input type="button" value="Click for more details"/>	

LOGINSERP.BLADE.PHP

4.7 Advanced search function:

This functionality will be available only for the users who logged in to the website. The advanced search has various fields where the users can filter their search results by book title, author. The above advanced search function takes the two fields 'title' and 'authors' and gives back the search results according to them.

The results are then generated in a separate page and I' have echoed the table to display the results. This is implemented in the SERP.php (Search Results Page).

Search by Filters

Title

Author

University

Name of the Degree

Department

The advanced search is implemented like the main search function using the multi match function. 'multi_match' => ['query' => \$q, 'fields'=>['title','author']]. The content can be found in the file advanced_search.blade.php

You can find the code of Main Controller in
Controllers -> Auth folder -> Main Controller->process_advsearch

4.8 SERP:

The content of the SERP can be found in the SERP.php file and the content of the LOGIN SERP (which opens when the user is authenticated) can be found in the LOGIN SERP.php file. The SERP contains a search bar where the users can search for various books. It displays 10 results per page where the results are paginated. The search terms are also highlighted where we used a user defined highlight words function.

4.9 Document summary page:

Summary

Title : A framework for finding and summarizing product defects, and ranking helpful threads from online customer forums through machine learning

Author : Jiao, Jian

University : Virginia Polytechnic Institute and State University

Publisher : Virginia Tech

Abstract : The Internet has revolutionized the way users share and acquire knowledge. As important and popular Web-based applications, online discussion forums provide interactive platforms for users to exchange information and report problems. With the rapid growth of social networks and an ever increasing number of Internet users, online forums have accumulated a huge amount of valuable user-generated data and have accordingly become a major information source for business intelligence. This study focuses specifically on product defects, which are one of the central concerns of manufacturing companies and service providers, and proposes a machine learning method to automatically detect product defects in the context of online forums. To complement the detection of product defects, we also present a product feature extraction method to summarize defect threads and a thread ranking method to search for troubleshooting solutions. To this end, we collected different data sets to test these methods experimentally and the results of the tests show that our methods are very promising: in fact, in most cases, they outperformed the current state-of-the-art methods.

Department : Computer Science

Year : 2013-06-05

Degree : PHD

Degree Level : doctoral

Every book is linked to a summary page which has the details of the book/document like title, contributed author, handle number, university, publisher, abstract etc. There are two views for summary page one is `summ.blade.php` and the other is `summary.blade.php`. These two views are designed for authenticated and non-authenticated users because logged in users have the privilege to add claim and can also vote-up and vote-down buttons. The summary page has all the necessary information of the book. The content can be found in `summary.blade.php` and `summ.blade.php`. The related backend code is provided in

Controllers -> Auth folder -> Main Controller->summary

Controllers -> Auth folder -> Main Controller->summ

4.10 XSS vulnerability filtering:

```
$term = strip_tags($_POST['q']);
```

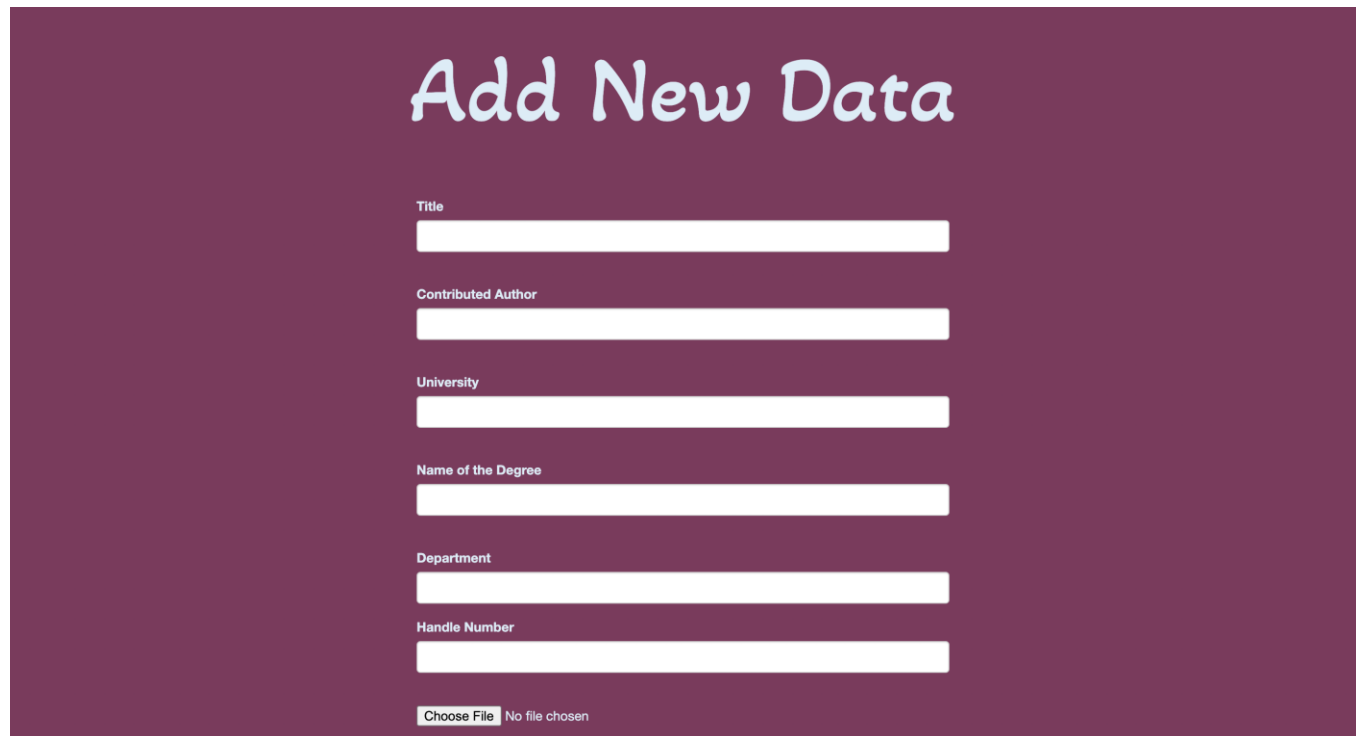
This functionality strips the tags and displays only text for searching.

For the SERP to display the actual term (after sanitization) on top, I have used the `csrf()` token field and `strip_tags()` function to filter the tags and displays the word and it looks like this : `$term= strip_tags($_POST['q']);`; where `q` is my result. The content of the functionality can be found in the `SERP.php` file and `LOGINSERP.php`

4.11 Insert a new entry:

This functionality will be available for both the users who have logged in as well as the ones who have not logged in to the website. The users can add any pdf file to the database only if they login. There is an alert that says 'we have indexed these' the books have been successfully indexed. The users can add a book title, author's name, university, name of the degree, department, and the handle number to the Elasticsearch. The content of this can be found in the `upload.blade.php` file and the code for the backend is written in main controller. You can find the code in

Controllers -> Auth folder -> Main Controller->add_data

A screenshot of a web form titled "Add New Data" on a dark purple background. The form contains six text input fields stacked vertically, each with a label above it: "Title", "Contributed Author", "University", "Name of the Degree", "Department", and "Handle Number". At the bottom of the form, there is a "Choose File" button and the text "No file chosen".

4.12 Pagination:

My pagination is based on the concept of how the control module fetches the total results from the Model (data), then sorts the first N elements and fetches another N results when the user clicks the next page link. I have used a List, JavaScript, Bootstrap and Ajax to implement the pagination. This functionality is implemented in SERP.blade.php file and LOGINSERP.php file

4.13 Highlighting search terms:

I have implemented this functionality using javascript mark color. This will find the words that we search in the search engine and displays with the highlighted color and can choose an. This functionality is implemented in both SERP.blade.php file and advanced_Search pages and LOGINSERP.php file.

Searched for: **bio**

Show **10**  entries

Title

(Re)presenting Human Population Database Projects: virtually designing and siting **bio** medical informatics ventures

[Click for more details](#)

[Summary](#)

A **Bio**-Economic Model of Long-Run Striga Control with an Application to Subsistence Farming in Mali

[Click for more details](#)

[Summary](#)

4.14 Save/Delete items to favorite list:

This functionality will be available only for the users who are logged in to the website. I have created a favorite table in database for the users to like their favorite books/thesis. The saved favorite items are added to the table automatically when the user clicks on add to favorite button near the book/thesis and these liked items are again redirected to an external link(favorites) on the users' home page where there is much detailed description of the result. The user can delete the items from the favorites and thereby the item is deleted from the favorite list and table. The user can also delete all the items in the favorite list by clicking on delete all button. The functionality is implemented by the method in controller. Lastly, if the user clicks on add to favorite button it is disabled and enabled only after it is deleted from the favorite list.

The content of this view is in favoritelist.blade.php
Resources->views->pages->favoritelist.blade.php and in
Controllers -> Auth folder -> Main Controller->addfav



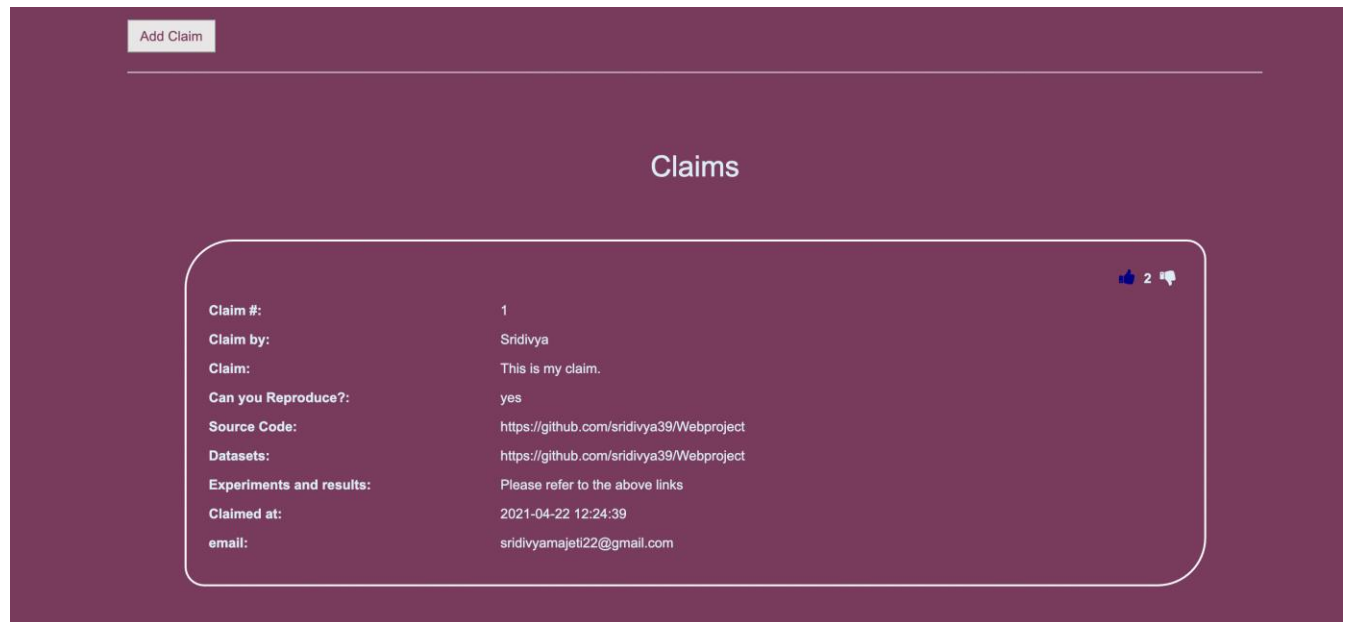
Title	Author	Year	Details	Delete
Application of Alternative Technologies to Eliminate <i>Vibrios</i> spp. in Raw Oysters	Hu, Xiaopei	2004-12-15	Summary	Delete
A framework for finding and summarizing product defects, and ranking helpful threads from online customer forums through machine learning	Jiao, Jian	2013-06-05	Summary	Delete

4.15 Discussion Board

This functionality is visible only to the authenticated users. Once the user clicks on the linked summary page. The user sees an Add claim button and clicking that he can see a set of questions related to the thesis and other authenticated users can vote up and vote down the claim.

The content is given in summary.blade.php file and the backend code are provided in Main Controller.

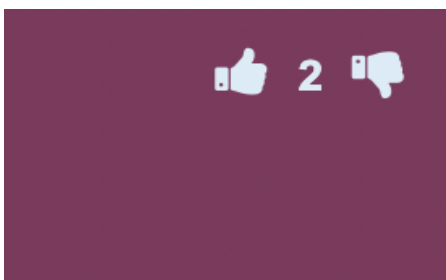
Controllers -> Auth folder -> Main Controller->process_claim



4.16 Voting

This functionality is visible only to the authenticated users. Once the user clicks on the linked summary page the user can see an add claim button, he can either add a claim or like/dislike the existing claims. The difference between up and down votes are displayed. If the existing claim is liked or disliked by the user then, the like/dislike button is turned to blue color to notify the user. I have updated the claim table with two more columns i.e., vote flag and vote count to implement this functionality. The content for the view is provided in summary blade (summary.blade.php file) itself as it is interlinked with the claim. The back end of the functionality is in the Main Controller.

Controllers -> Auth folder -> Main Controller->voteUpAndDown



4.17 Spell check:

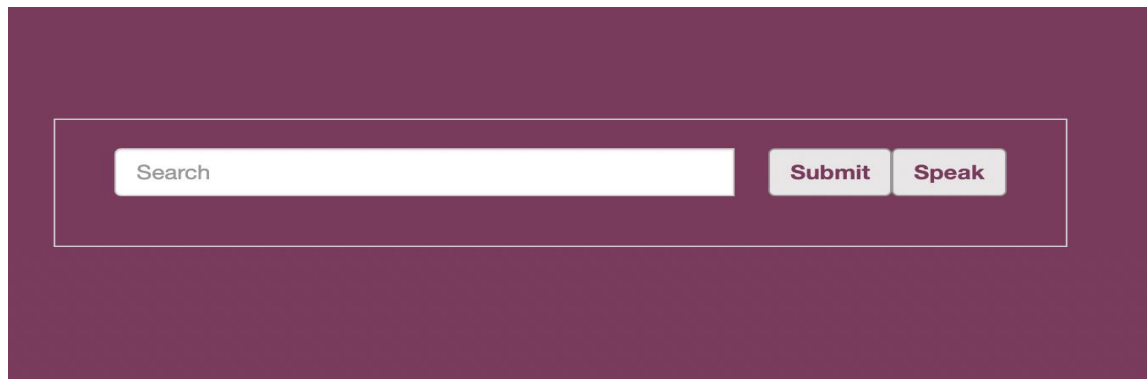
I have implemented the Spell check functionality in my landing page, search engine, home page. This automatically validates the text typed in by the user and searches for the related word in the ETD (indexed data). I designed this using fuzziness parameter in the written search query and have accomplished this feature. You can find the related code in:

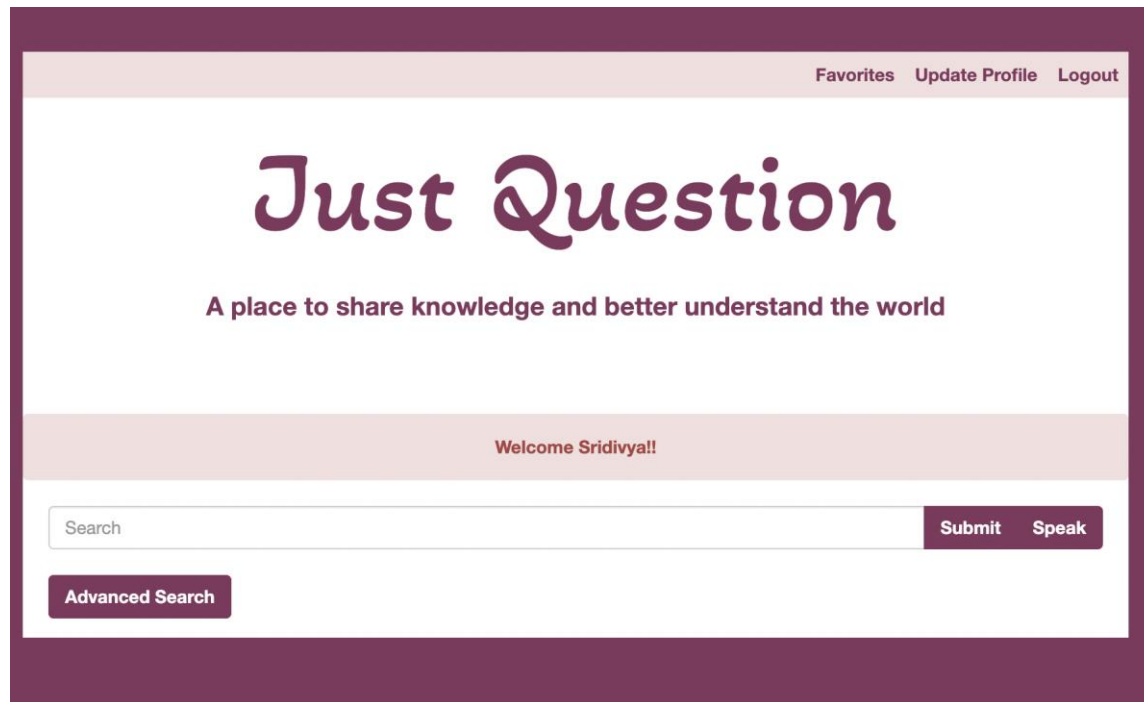
Controllers -> Auth folder -> Main Controller-> search
Controllers -> Auth folder -> Main Controller-> loginsearch
And in
Resources->views->pages->index.blade.php
Resources->views->pages->successlogin.blade.php

4.18 Speech-to-text API:

I have implemented the speech to text API functionality in my landing page, home and SERP pages. I have used HTML5 Speech Recognition API to implement this. I have added a button named Speak besides the submit button when that is clicked it records and then converts the speech to text. I have implemented speech to text recognition to enable this functionality. The code content is available in index.blade.php, successlogin.blade.php, LOGINSERP.php and SERP.php files. You can find the related code in:

Controllers -> Auth folder -> Main Controller-> search
Controllers -> Auth folder -> Main Controller-> loginsearch
And in
Resources->views->pages->index.blade.php
Resources->views->pages->successlogin.blade.php





4.19 Downloading PDF files:

This functionality will be available for both the users who have logged in as well as the ones who have not logged in to the website. I have created a download button, where the users can download the pdf documents of that book. To implement this, I have created a button and linked it with the handle number. In some folders there are multiple pdfs I have used if condition and considers the very first document in that folder. The functionality is implemented in my SERP.php, LOGIN SERP.php file



5. Challenges and Lessons:

The challenges I have faced in curating this website was

1. Creating the summary page and adding a discussion board in it.

I faced difficulties in obtaining the unique value of each book as the SERP page is generated when the user searches for word. whereas, the summary should work when the user clicks on the SERP page. So, I passed the same query string to that page as well so it obtains the unique value (handle number) and echo all the related values to that book using it.

2. Claim, vote up and vote down

I faced difficulties is in making the discussion board. Then I chose the unique value in the users table and then designed this feature. Similarly, for like and dislike I have decided to use AJAX, java script and added two more columns to the claim table itself and got this working.

3.Indexing the project data.

I faced trouble in indexing the data as obtaining the json format files from all the folders was a bit challenging until I realized that there is one unique column that is handle number. So, I wrote a for loop and a query to index all the files accordingly.

These are few challenges that I faced during the project.

6. Additional comments:

I was new to this web programming subject. I had no clue how it works and I had no knowledge in web technologies before taking up this course. I only knew basic php,html,css. But the course materials were helpful and I followed every step and did a lot of homework for this course and exposed myself to a variety of languages.

I used JavaScript, bootstrap for CSS, AJAX for styling and designing, php, html and used Laravel framework which is an MVC architecture which gave me clear knowledge on what happens in the backend as well as frontend. I also got familiar with queries and databases. This course will be helpful for me in future.

Milestone submitted on Git

link: <https://github.com/sridivya39/Webproject>