Project: A digital library search engine for electronic thesis and dissertations

Student name: Triveni Sangama Saraswathi Edla

Student UIN: 01160800

GitHub link: https://github.com/Triveniedla/webprogramming-Triveniedla

Overview:
Nowadays people are using electronic thesis and dissertations instead of hard copies. With the current pandemic and social distancing, there is an increased usage of online resources for electronic copies. In this project, a website is designed which provides a digital library search engine for electronic thesis and dissertation (ETD). The Django framework is used to build the website. Django is a high-level python web framework that enables rapid development of the secure and maintainable website and uses model-template-view architecture pattern. A combination of HTML, CSS, and JavaScript is used to design the website and all the requirements are met.

1. Milestone Accomplishments

List ALL specifications of this project and specify whether certain specifications are fulfilled or not.

Table 1: Overview of status for All milestone specifications.

Fulfilled	#	**Specification**
		Milestone 1 Specifications
Yes	1	The website should provide a search box at the landing page. The searching function may not be working.
Yes	2	There should be a search button next to the search box.
Yes	3	Users must be able to register new accounts using email addresses.
Yes	4	Password must be encrypted before storing in the database.
Yes	5	Users can not register duplicate accounts using the same email address, or phone number.
Yes	6	Users should be able to log into your website using the accounts they registered.
Yes	7	Users should be able to reset their passwords if they forget it.
Yes	8	The user login process must use the HTTP POST method.
Yes	9	User information shall be stored in a MySQL database.
Yes	10	The website should have a homepage for each user, where they can view their profiles, change passwords, and update information.
		Milestone 2 Specifications
Yes	1	Users should be able to get a confirmation email to verify their email addresses for registration or password reset.

Yes	2	The website has an “Advanced Search” function so users can search multiple fields.
Yes	3	The advanced search should return results satisfying multiple specifications.
Yes	4	The website should index at least 5000 “documents” (a document can be metadata of an image or metadata of an ETD).
Yes	5	The search engine accepts a text query in the search box.
Yes	6	The search engine should return search results on the search engine result page (SERP), which can be links to documents or images.
Yes	7	The search engine should display the number of returned items on SERP.
Yes	8	The SERP should contain a search box.
Yes	9	The search engine can prevent XSS vulnerability by removing tags existing in the query.
Yes	10	Users should be able to insert a new entry (document) and search engine will index it.
		Milestone 3 Specifications
Yes	1	The search engine can return paginated results.
Yes	2	The search engine can highlight results that contain search terms.
Yes	3	The SERP should display the actual term (after sanitization) shown on top.
Yes	4	Users can click each item on SERP and go to either an external link or a page containing more information of the item.
Yes	5	Users can save items in search result to their profiles.
Yes	6	Users have to login first to save search history to their profiles.
Yes	7	reCAPTCHA should be used for both the logging in and the signing up page.
		Milestone 4 Specifications
Yes	1	Users can delete items from their favorite list.
Yes	2	Items in the favorite lists should be descriptive (cannot be just a link) and are linked to an external page or summary page of the item.
Yes	3	The search engine implements at least one of the features spell check, autocomplete, Speech-to-text API, or other APIs permitted by the instructor.
Yes	4	There is a button from which users can download documents (or images) from the summary page or from the SERP (or both).
		Extra credits
No	1	RESTful API [2 points]
Yes	2	Logged in users can like a document like Facebook. [4 points]

2. Architecture

The architecture of the electronics thesis and dissertation (ETD) website is shown in Figure 1. On the landing page, the user can search for ETD, login, or sign up. Signing up will send a confirmation email to verifying the account and the verified users can log in. The ReCaptcha feature is implemented for both logging in and signing up. The users with a verified account can reset the password at the login page using the password reset link or after logging in. Logging in will provide an extra feature to change the password manually or using a reset link, which will send a link to the email id to reset the password.

The users can search for ETD without logging in or after logging in from the search engine page. The search engine page has a search box, advanced search features, and speech-to-text feature and users can use these features to search with a keyword. The search engine will return ETD documents paginated in the search engine results page (SERP) and more details of each ETD document can be found in the more details page, which is provided as a button under each document. The SERP page also has the search box, advanced search features, and speech-to-text features, which will enable the user to search from the SERP page as well. The logged-in user will have an additional option to save the favorite, automatic saving of search history and like details.

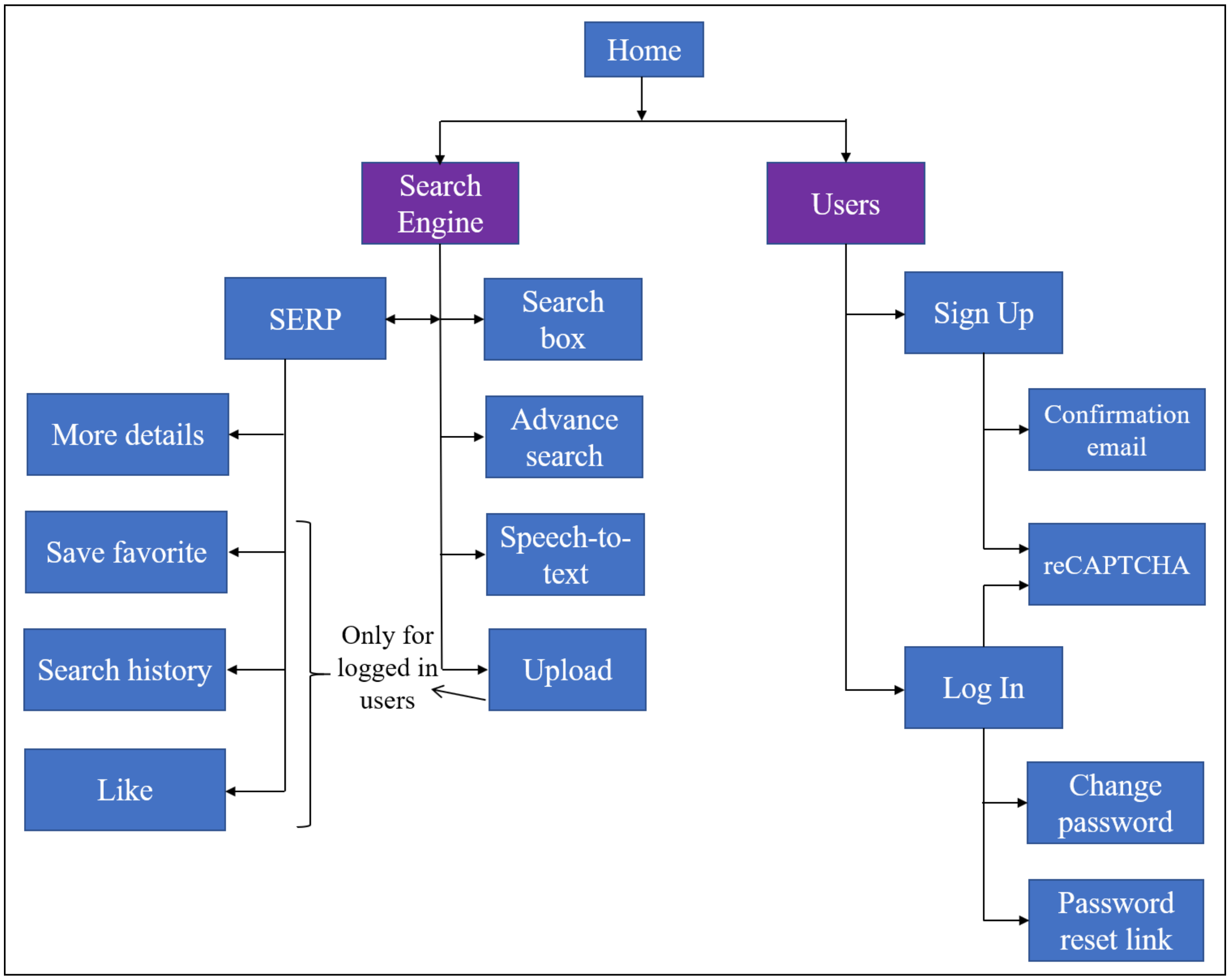


Figure 1 Architecture of the electronics thesis and dissertation (ETD) website. Users can search without sign up and signed up users will have additional options to save the favorites, search history, and like details. Signed up users will get a confirmation email to verify their account and reCAPTCHA validation is needed for both logging in and signing up. Users with a valid account with the ETD website can reset the password

using the link at the login page or reset after logging in. The SERP page will provide paginated results of the ETD and have a search box, advance search, and speech-to-text.

3. Data

3.1. Source of data for the website

The elastic thesis data (ETD) is used for building the website. The ETD database is provided by the course instructor and it is indexed into the Elasticsearch server. It must be noted that each ETD has a unique number stored in the field name: “handle”. A total of 3950 ETD documents are indexed into the Elasticsearch server.

3.2.Fields used for indexing

Following fields are indexed into the Elasticsearch to be used in the website. The field names in the elastic search are shown below.

- "contributor_author"
- "date_accessioned"
- "date_available",
- "date_issued",
- "identifier_other",
- "identifier_uri",
- "identifier_sourceurl",
- "identifier_oclc",
- "description",
- "description_abstract",
- "description_provenance",
- "description_sponsorship",
- "format_medium",
- "publisher",
- "rights",
- "subject",
- "subject_lcc",
- "subject_lcsh",
- "title",
- "type",
- "language_iso",
- "relation",
- "contributor_department",
- "description_degree",
- "contributor_committeechair",
- "contributor_committeecochair",
- "contributor_committeemember",
- "degree_name",
- "degree_level",

- "degree_grantor",
- "degree_discipline",
- "handle",
- "relation_haspart",
- "date_adate",
- "date_sdate",
- "date_rdate"

3.3. Fields used for regular search

In the regular search, the user can enter the text to search and the query is sent to Elasticsearch to find if the word matches with any part of the title or abstract and returns the matched ETD.

3.4. Fields used for the advanced search

The fields included in the advanced search are department, author, committee chair, degree, and dates between which the ETD was issued.

3.5. Database tables used for the website

Table 2. Tables used for the website

SQL table	Functionality	Mandotory columns in the table (name, type)
users_user	Save the user details	id (auto-increment), last_login (automatic), password (encrypted), date (automatic), email, is_staff (Boolean), is_active (Boolean)
users_saveitemmodel	Save favorite ETD using handle number	id (auto-increment), handle(int), date (automatic), user_id(user's id)
users_ searchhistorymodel	Save user search history	id (auto-increment), searchtext (string), date (automatic), user_id(user's id) , date1(auto), date2 (auto) , description_degree, contributer_author, contributer_committeechair, contributer_department
users_ likeitemmodel	Save liked ETD using handle number	id (auto-increment), handle(int), date (automatic), user_id(user's id)
users_ handlemodel	Save last handle number for indexing a new ETD	id (auto-increment), handle(int), date (automatic), user_id(user's id)

Table 2 shows tables used to store the details of the users, favorite ETDs, search history, liked ETDs, and last handle number of upload. The details of the tables are described below:

- The users_user table will have the columns: auto-incremented id, encrypted password, last logged in, email id of the user, and is_active. The is_active column is Boolean which will become true if the user has activated his account.
- The users_saveitemmodel table stores the favorite ETDs saved by the user using the unique handle number of ETD. The columns of the tables are the autoincremented user_id, unique handle number of ETD and date is automatically assigned. The user_id is the id in the users_user table. If the user saves multiple handle numbers then, there will be multiple rows with the same user_id and different handle numbers. The table can also stores handle numbers saved by all users.
- The users_searchhistorymodel table saves the search history of the user including the advanced search text and the date of search. This table has auto-incremented id, columns with search, and advance search text.
- The users_likeitemmodel table saves the liked item by the user using the unique handle number. This table has similar columns as the users_saveitemmodel table.
- The users_ handlemodel table saves the last handle number of the ETDs stored in the elastic search. This table has similar columns as users_saveitemmodel table. The handle number in the first row is the last handle number of the ETDs in the Elasticsearch server and it will be incremented when a new ETD is uploaded.

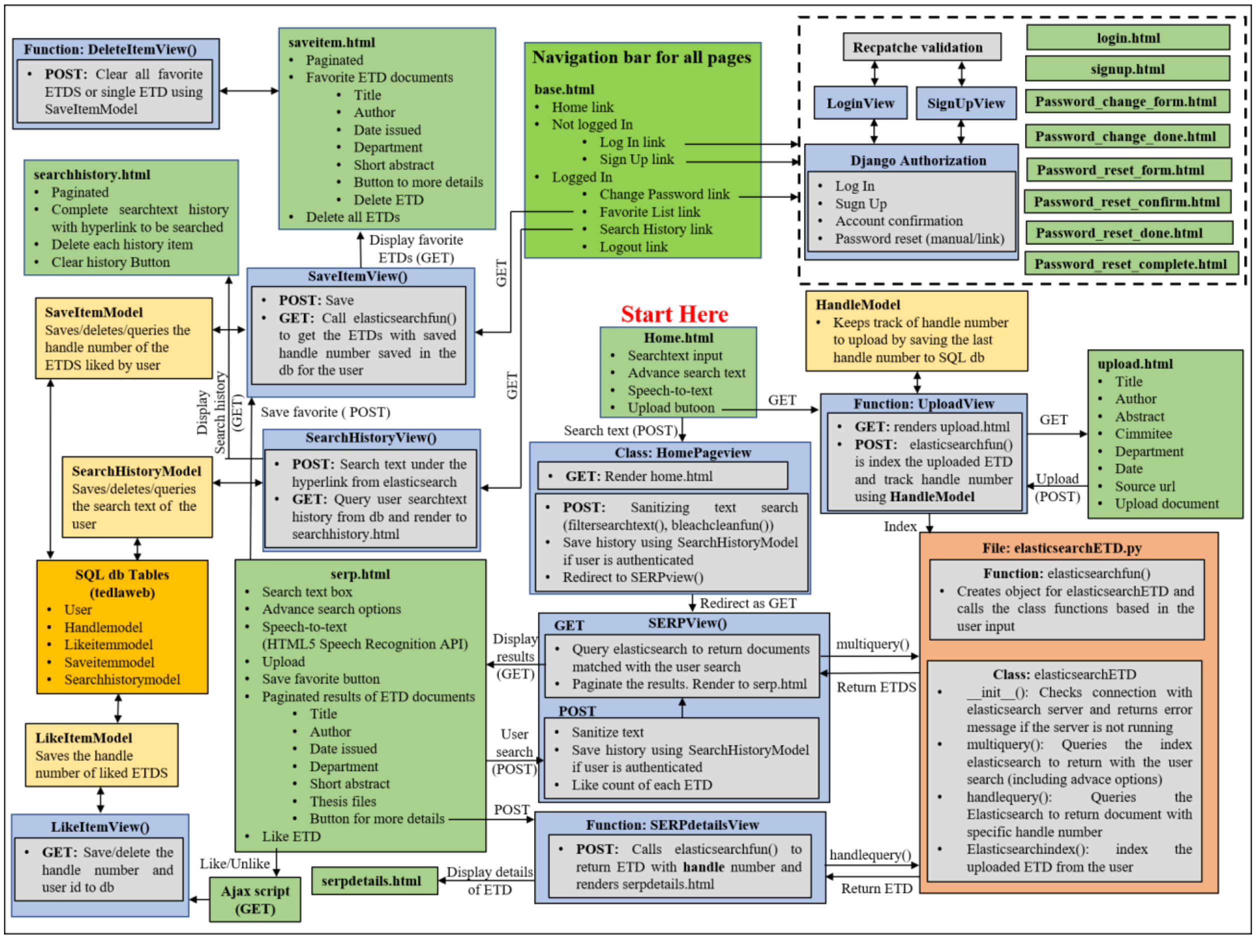
4. Implementation

In this section, how each page is presented to the user at the front end is shown using pictures and details of the implementation are discussed. The website is developed using the Django web framework [1], which uses a model-view-template framework and the entire framework is based on python. The model-view-template does follow functionalities:

- The model interacts with the database to save/query/delete the required row from the database tables. MySQL is used as the database.
- The templates have HTML code that builds the visual aspects of the website that can be seen and experienced by users at the frontend. The Django web framework provides a range of tools and libraries to help to build forms to accept input from site visitors, and then process and respond to the input.
- Views are Python classes or functions that take a web request and return a web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, or anything.

Figure 2 is the complete outline of the entire website, showing the interactions between models, views, templates, and SQL database built using Django. The green boxes indicate the HTML templates and important contents displayed on the web page are indicated. The blue background boxes are the views that take the web request and return a web response. The yellow boxes indicate the models that are used by views to interacts with the SQL database. The orange box is the SQL database and the tables used for building the website. The figure also shows details of direction from the buttons and links in the HTML template to different views and the interaction between different views and models. The interaction between views, templates, and models are discussed in the implementation section, and below are the locations where the files are stored:

- All the models used for this website is stored in the models.py file in the /users folder in the Github account
- The templates are stored in the /templates folder.
- All the views are stored in the views.py file under /users folder.
- Views and templates use the forms feature of Django which helps build forms to accept input from site visitors, and then process and respond to the input. The forms will be used in the templates to present the user with forms to submit user requests and the forms are used in the views to process the data entered by the user in the forms. The forms are stored in forms.py file under /users folder.
- The URL patterns between the template and views are designed in the urls.py files under /users folder.



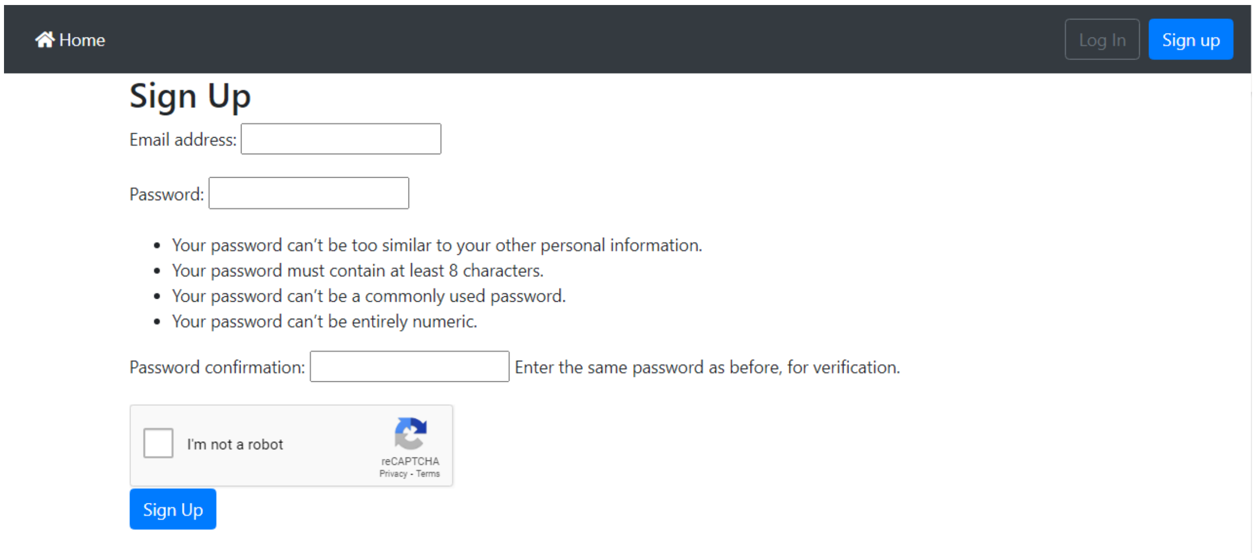
Function: DeleteItemView()
POST: Clear all favorite ETDS or single ETD using SaveItemModel
saveitem.html
Paginated
Favorite ETD documents
Title
Author
Date issued
Department
Short abstract
Button to more details
Delete ETD
Delete all ETDs
Navigation bar for all pages
base.html
Home link
Not logged In
Log In link
Sign Up link
Logged In
Change Password link
Favorite List link
Search History link
Logout link
Recpatche validation
LoginView
SignUpView
Django Authorization
Log In
Sugn Up
Account confirmation
Password reset (manual/link)
login.html
signup.html
Password_change_form.html
Password_change_done.html
Password_reset_form.html
Password_reset_confirm.html
Password_reset_done.html
Password_reset_complete.html
searchhistory.html
Paginated
Complete searchtext history with hyperlink to be searched
Delete each history item
Clear history Button
Display favorite ETDs (GET)
SaveItemView()
POST: Save
GET: Call elasticsearchfun() to get the ETDs with saved handle number saved in the db for the user
GET
SaveItemModel
Saves/deletes/queries the handle number of the ETDS liked by user
Display Search history (GET)
Save favorite (POST)
Start Here
Home.html
Searchtext input
Advance search text
Speech-to-text
Upload butoon
HandleModel
Keeps track of handle number to upload by saving the last handle number to SQL db
upload.html
Title
Author
Abstract
Cimmitee
Department
Date
Source url
Upload document
GET
SearchHistoryView()
POST: Search text under the hyperlink from elasticsearch
GET: Query user searchtext history from db and render to searchhistory.html
SearchHistoryModel
Saves/deletes/queries the search text of the user
Search text (POST)
Class: HomePageview
GET: Render home.html
POST: Sanitizing text search (filtersearchtext(), bleachcleanfun())
Save history using SearchHistoryModel if user is authenticated
Redirect to SERPview()
Function: UploadView
GET: renders upload.html
POST: elasticsearchfun() is index the uploaded ETD and track handle number using HandleModel
GET
Upload (POST)
Index
SQL db Tables (tedlaweb)
User
Handlemodel
Likeitemmodel
Saveitemmodel
Searchhistorymodel
serp.html
Search text box
Advance search options
Speech-to-text (HTML5 Speech Recognition API)
Upload
Save favorite button
Paginated results of ETD documents
Title
Author
Date issued
Department
Short abstract
Thesis files
Button for more details
Like ETD
Redirect as GET
Display results (GET)
SERPView()
Query elasticsearch to return documents matched with the user search
Paginate the results. Render to serp.html
POST
Sanitize text
Save history using SearchHistoryModel if user is authenticated
Like count of each ETD
User search (POST)
multiquery()
Return ETDS
File: elasticsearchETD.py
Function: elasticsearchfun()
Creates object for elasticsearchETD and calls the class functions based in the user input
Class: elasticsearchETD
__init__(): Checks connection with elasticsearch server and returns error message if the server is not running
multiquery(): Queries the index elasticsearch to return with the user search (including advace options)
handlequery(): Queries the Elasticsearch to return document with specific handle number
Elasticsearchindex(): index the uploaded ETD from the user
LikeItemModel
Saves the handle number of liked ETDS
LikeItemView()
GET: Save/delete the handle number and user id to db
Like/Unlike
Ajax script (GET)
serpdetails.html
Display details of ETD
POST
Function: SERPdetailsView
POST: Calls elasticsearchfun() to return ETD with handle number and renders serpdetails.html
handlequery()
Return ETD

Figure 2. The complete outline of the entire website, showing the interactions between models, views, templates, and SQL database built using Django. The green boxes indicate the HTML templates and important contents displayed on the web page are indicated inbox. The blue background boxes are the views that take the web request and return a web response. The yellow box indicates the models that are used by

views to interacts with the SQL database. The orange box is the SQL database and the tables used in the website.

4.1.Account registration (including reCAPTCHA and confirmation email, if applicable)

For account registration, the authentication functionality provided by Django is used. The authentication handles user accounts, groups, permissions, and cookie-based user sessions. This section of the documentation explains how the default implementation works out of the box, as well as how to extend and customize it to suit your project's needs. The Django authentication system handles both authentication and authorization. Briefly, authentication verifies a user is who they claim to be, and authorization determines what an authenticated user is allowed to do. Here the term authentication is used to refer to both tasks.



Sign Up

Email address:

Password:

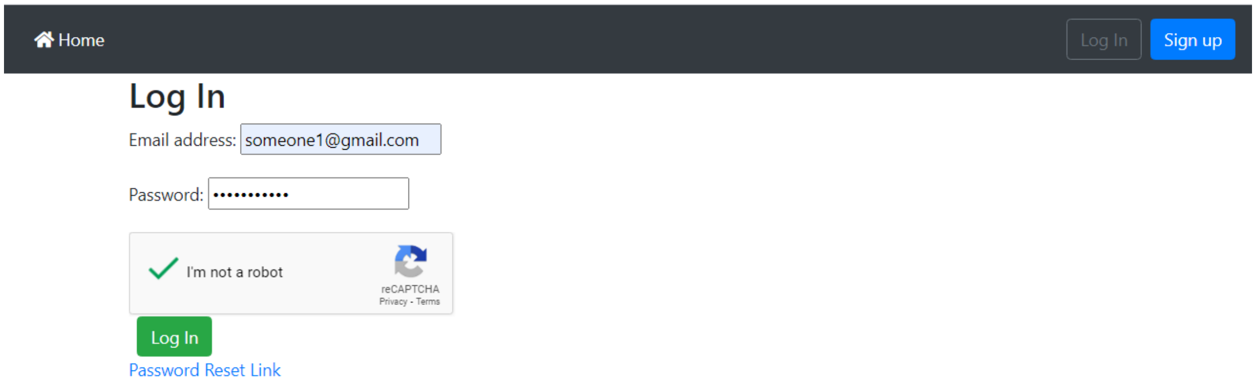
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

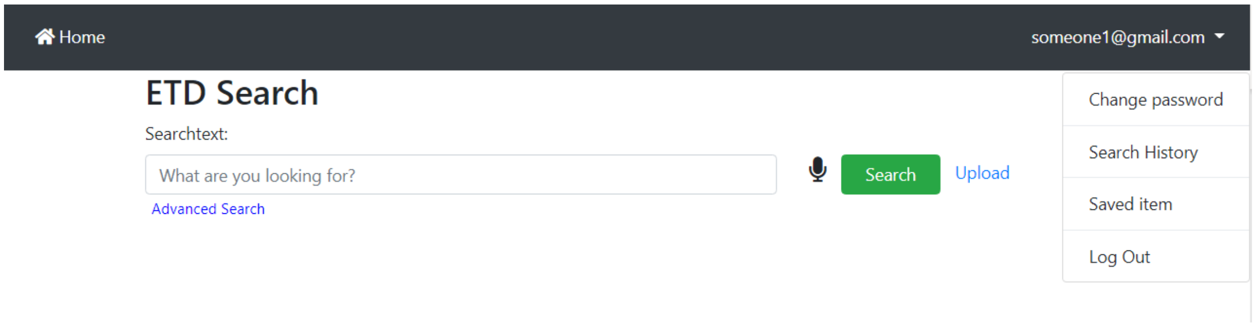
The above figure shows the sign-up page, which is designed using signup.html.The user must enter the email address, password, and password confirmation to register their account. The inbuilt UserCreationForm of Django is used to accept user credentials. The form will interact with the database and return an error message if the user is already registered. If the user is not registered, the credentials are sent to SignUpView() function and the function will generate a link with a unique token using account_activation_token() function available in token_generator.py. The link will be sent to the email id used by the user. Before submitting the results, the user has to validate the reCAPTCHA. The reCAPTCHA is implemented using the reCAPTCHA functionality provided by Google [2] and it is validated in the SignUpView() function by comparing with the secret key provided by the google API.

Once the user clicks the account using the link sent to his email, the request will be sent to activateaccount() function in the views.py. The function validates by decoding the token generated using token_generator.py and verifying with the users_user table in the database.

4.2.Account login (including reCAPTCHA)



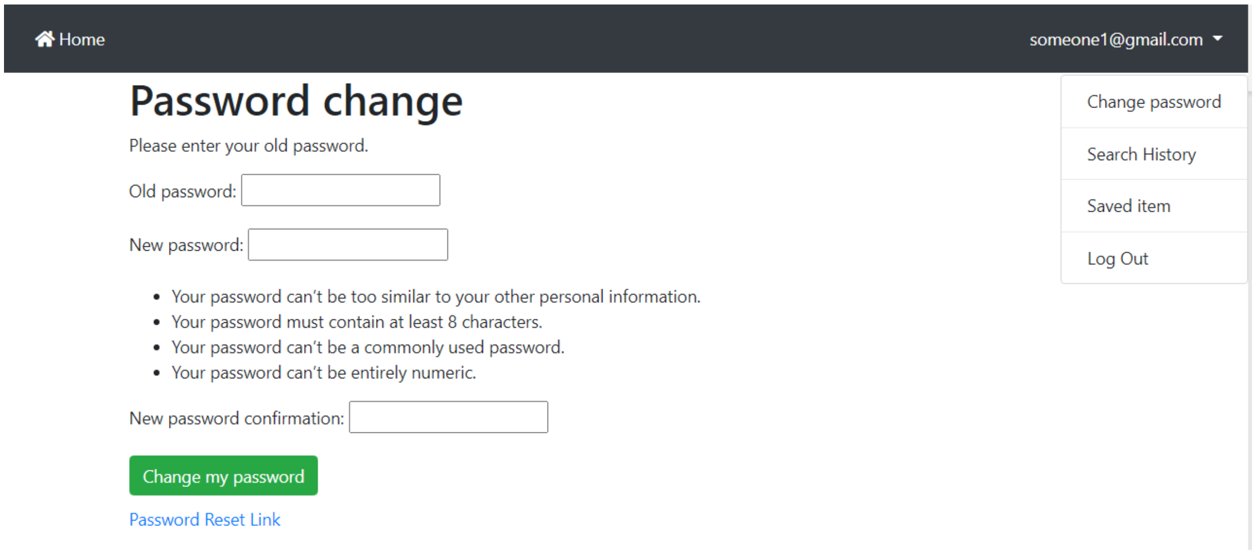
The authentication functionality provided by Django is used for login as well. The above figure shows the login page of the website. The credentials entered by the user is validated with the database in the LogInView() function and using AuthenticationForm() provided by the Django web framework. The reCAPTCHA is also validated in the login page similar to the signup page. After logging in, the landing page of the webpage looks as shown below.



All the pages have a navigation bar as shown at the top. The navigation bar has a home button which helps the user to redirect to the main page. On the right side of the web page, user can see their email id, if they click on it, they will see the fields of change password (where user can change password), search history (whatever searched by user will be saved in search history field), saved item (Where users can save their favorite links), and Log out (where user can log out from the webpage).

4.3.Password reset (including email, if applicable)

The user can change the password using the change password page as shown in the above figure. The user can change the password using the usual method of verifying the old password and creating a new password or using the password reset link. The password reset link is also provided on the login page and when the user clicks the reset link, a link is sent to the user's email address using PasswordResetView(urls.py) provided by Django. Once the user changes the password using the usual method or using the password reset, the PasswordResetConfirmView (urls.py) provided by Django is used for resetting the password in the database.



Password change

Please enter your old password.

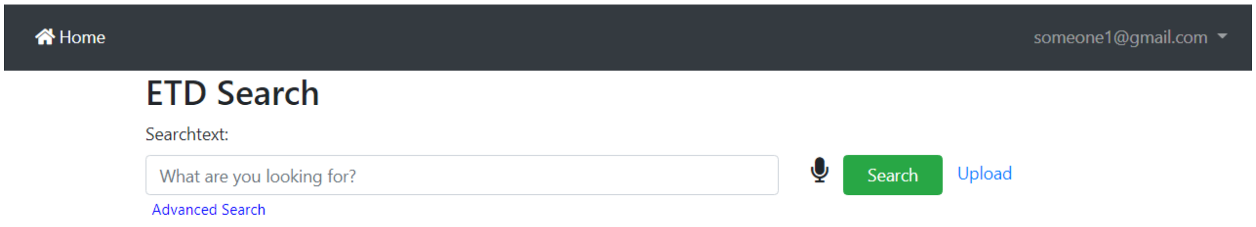
Old password:

New password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation:

4.4.Users' homepage

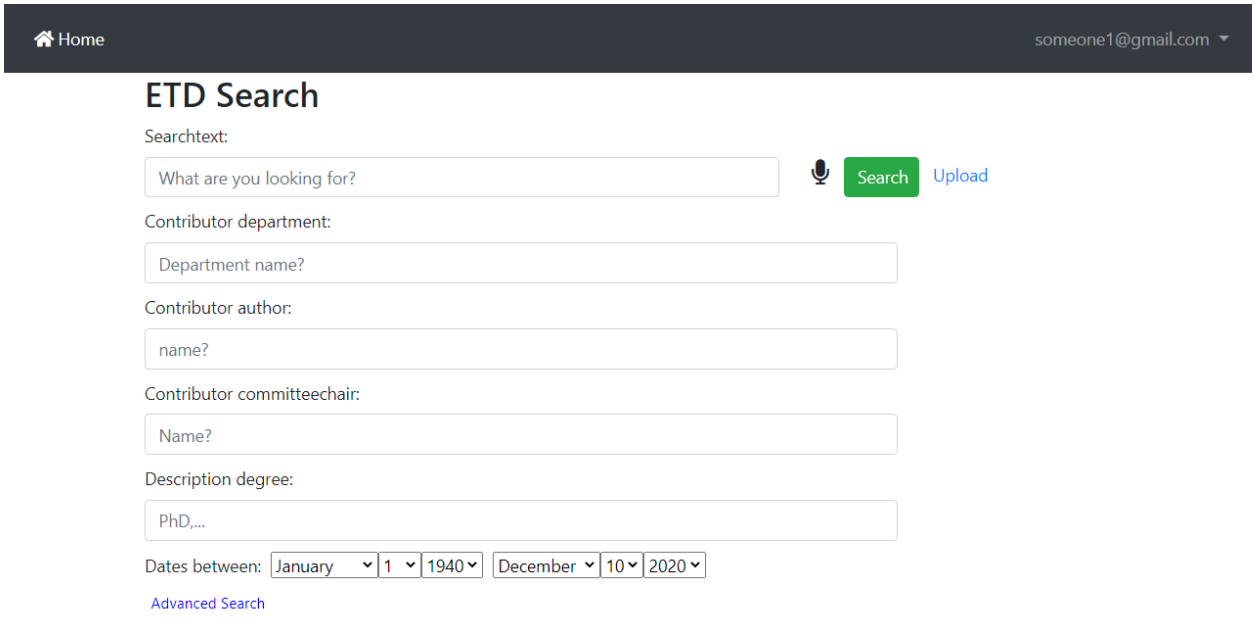


The above figure shows the home page of the website. There is a search bar with a search button where user can type keywords which displays the results belonging to that keyword. The user can enter the text of search, advance search, upload a document, and use speech-to-text for searching a text. The search entered by the user in either the search box or advanced search is handled by HomePageView() function. The details of each feature on the home page are discussed in their respective section.

4.5 Main search function

The search box in the main search function is implemented using the HomeForm class and the input text searched by the user is handled by class: HomePageView(). The view class sanitizes the text searched by the user using the bleach module available in python and sanitization is implemented using the bleachclean() and the filtersearchtext() functions. After sanitizing, the search text is redirected to SERPView() function. The bleachclean() and filtersearchtext() functions are used for both main search and advanced search functionalities.

4.6 Advanced search function



The items used in the advance search feature is shown above. The input boxes for the search and advance search is created using the HomeForm class present in forms.py. On the home page, the advanced search features are hidden using a script. The text entered in the boxes are sanitized using bleachclean() and filtersearchtext() functions. The filtersearchtext() function is customized in such a way that the website can deal if only some of the advanced search features are entered. However, the text must be entered in the main search text box.

The search and advance search redirected to serp page. The HTML code for search and advance search is available in advancesearch.html template.

4.7 SERP

The search text searched by the user either using the simple search or advance search is queried from the Elasticsearch and results are presented on the search engine results page (SERP) as shown below. The SERP page presents the ETDs that is matched with the search text. Each ETD has title author, date issued, department, short abstract, save button, like button, and button to download the documents of the ETD. The results are paginated. The entire functionality is implemented in SERPView function. The SERPView will call the elasticsearchfun() function present in the elasticSearchETD.py file under /users folder.

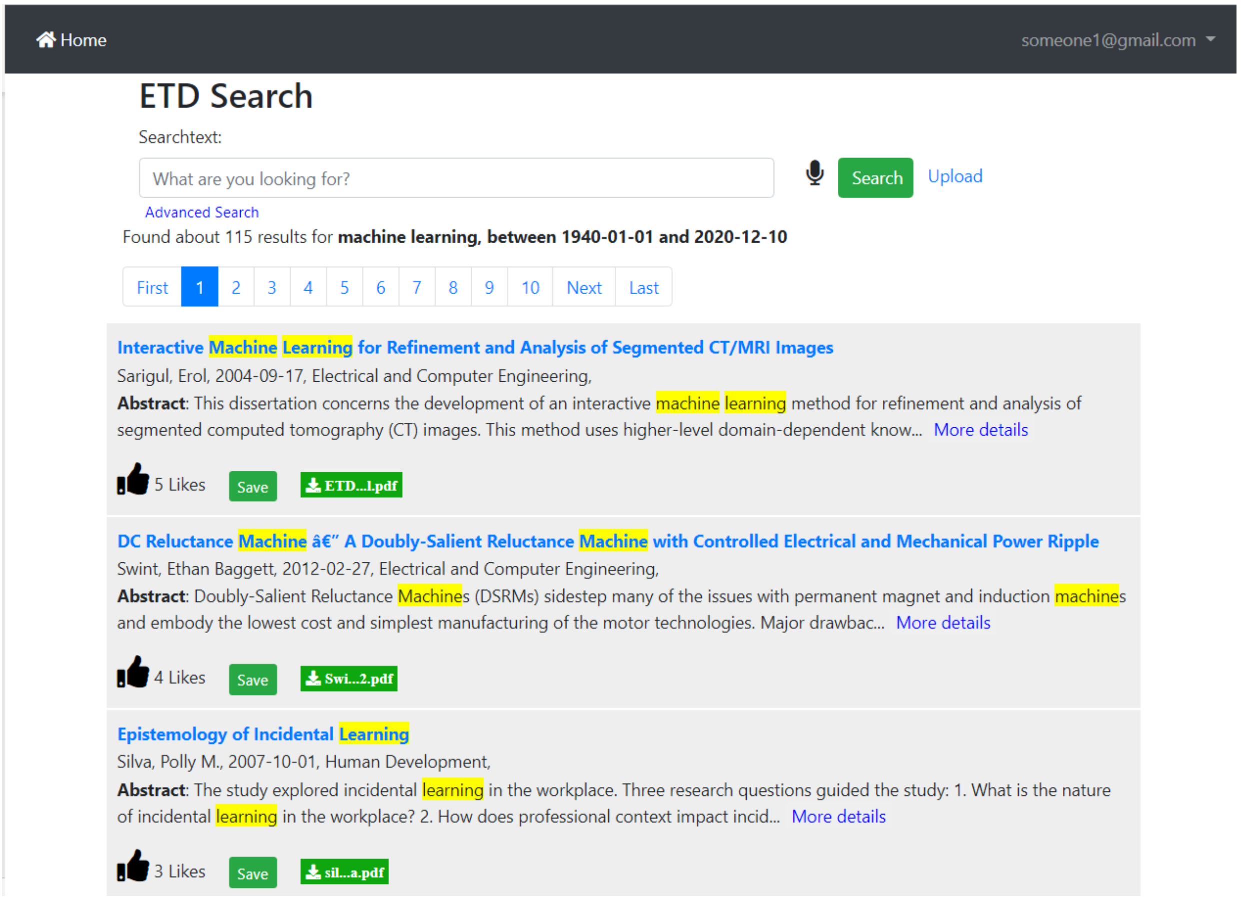
The elasticSearchETD.py file contains the main function elasticsearchfun() and class elasticsearchETD. elasticsearchfun() function will create an object for elasticsearchETD class. The elasticsearchETD class initializes a connection with the Elastisearch and it has methods with multiple options such as:

multiquery(): It searches the given search text in the title and abstracts of the ETDs and returns ETDs that match the search text.

handlequery(): This function returns ETD with a particular handle number. The handle number is unique for each ETD.

elasticsearchindex(): This function uploads the ETD from the user.

The SERPView() function calls multiquery() through elasticsearchfun() function to get the ETDs. The ETDs are paginated using the Paginator package [3] available with python. The serp.html templates displays all the ETDs got from elasticsearchfun() in SERPView() function.



ETD Search

Searchtext: What are you looking for?

Found about 115 results for **machine learning, between 1940-01-01 and 2020-12-10**

Interactive Machine Learning for Refinement and Analysis of Segmented CT/MRI Images
Sarigul, Erol, 2004-09-17, Electrical and Computer Engineering,
Abstract: This dissertation concerns the development of an interactive machine learning method for refinement and analysis of segmented computed tomography (CT) images. This method uses higher-level domain-dependent know... More details
5 Likes Save ETD...l.pdf

DC Reluctance Machine â€" A Doubly-Salient Reluctance Machine with Controlled Electrical and Mechanical Power Ripple
Swint, Ethan Baggett, 2012-02-27, Electrical and Computer Engineering,
Abstract: Doubly-Salient Reluctance Machines (DSRMs) sidestep many of the issues with permanent magnet and induction machines and embody the lowest cost and simplest manufacturing of the motor technologies. Major drawbac... More details
4 Likes Save Swi...2.pdf

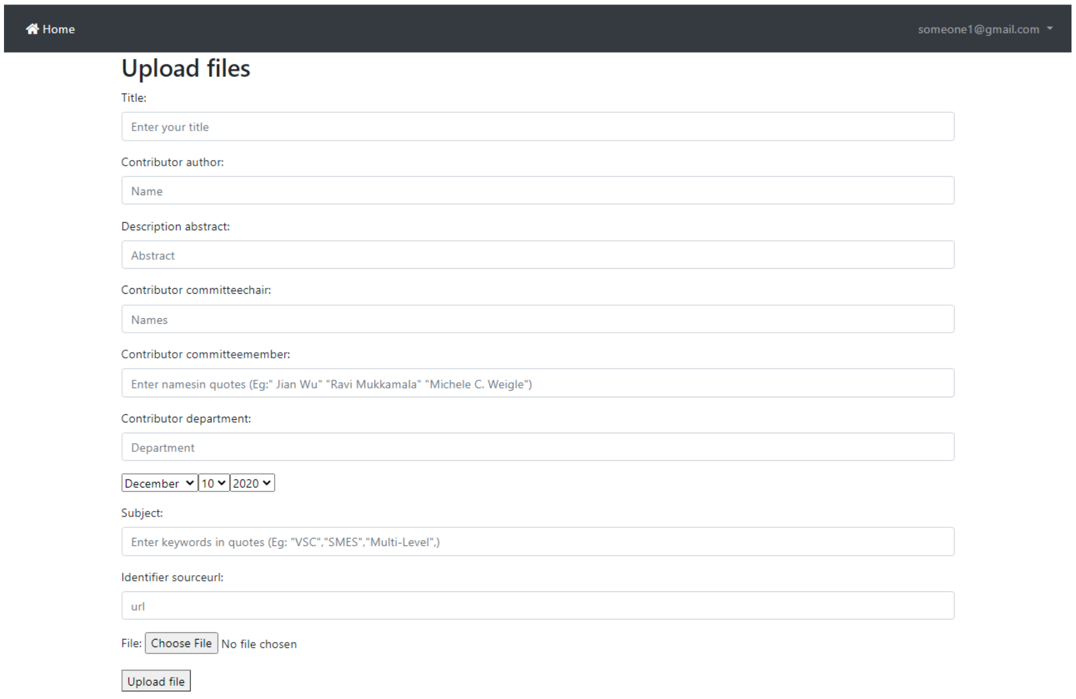
Epistemology of Incidental Learning
Silva, Polly M., 2007-10-01, Human Development,
Abstract: The study explored incidental learning in the workplace. Three research questions guided the study: 1. What is the nature of incidental learning in the workplace? 2. How does professional context impact incid... More details
3 Likes Save sil...a.pdf

4.8 XSS vulnerability filtering

The cross-site scripting (XSS) protection is provided by the Django templates [3]. The Django templates provide XSS vulnerability filtering by adding CSRF token in the templates. Further, the text entered by the users is cleaned using the bleach package [4] available with python. The bleach will clean if there is any script in the user text before searching in the Elasticsearch.

4.9 Insert a new entry

The below figure shows the entries that the user can upload. The user can also upload a document associated with the ETD.



Upload files

Title: Enter your title

Contributor author: Name

Description abstract: Abstract

Contributor committeechair: Names

Contributor committeemember: Enter namesin quotes (Eg:" Jian Wu" "Ravi Mukkamala" "Michele C. Weigle")

Contributor department: Department

December 10 2020

Subject: Enter keywords in quotes (Eg: "VSC","SMES","Multi-Level",)

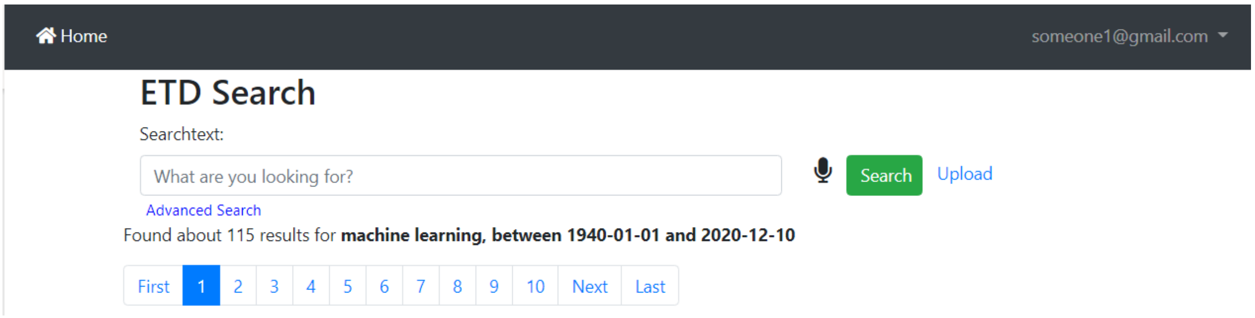
Identifier sourceurl: url

File: Choose File No file chosen

Upload file

The upload is implemented using the UploadView() function and the upload.html template. The UploadView function indexes to the Elasticsearch using the handlequery() function in the elasticsearchETD.py file. The UploadView() gives a unique handle number to the uploaded ETD. The handle number to upload is tracked in the handlemodel table of the database. The HandleModel class in the models.py function is used to interact with the handlemodel table.

4.10 Pagination



Found about 115 results for **machine learning, between 1940-01-01 and 2020-12-10**

The pagination is implemented using the Paginator package [5] available in python. The pagination is implemented in the paginationfun() function available in views.py. This function is used in the SERP page, favorite items, and search history. The pagination is implemented in pagination.html and the code from stack overflow is used and customized for this purpose [4].

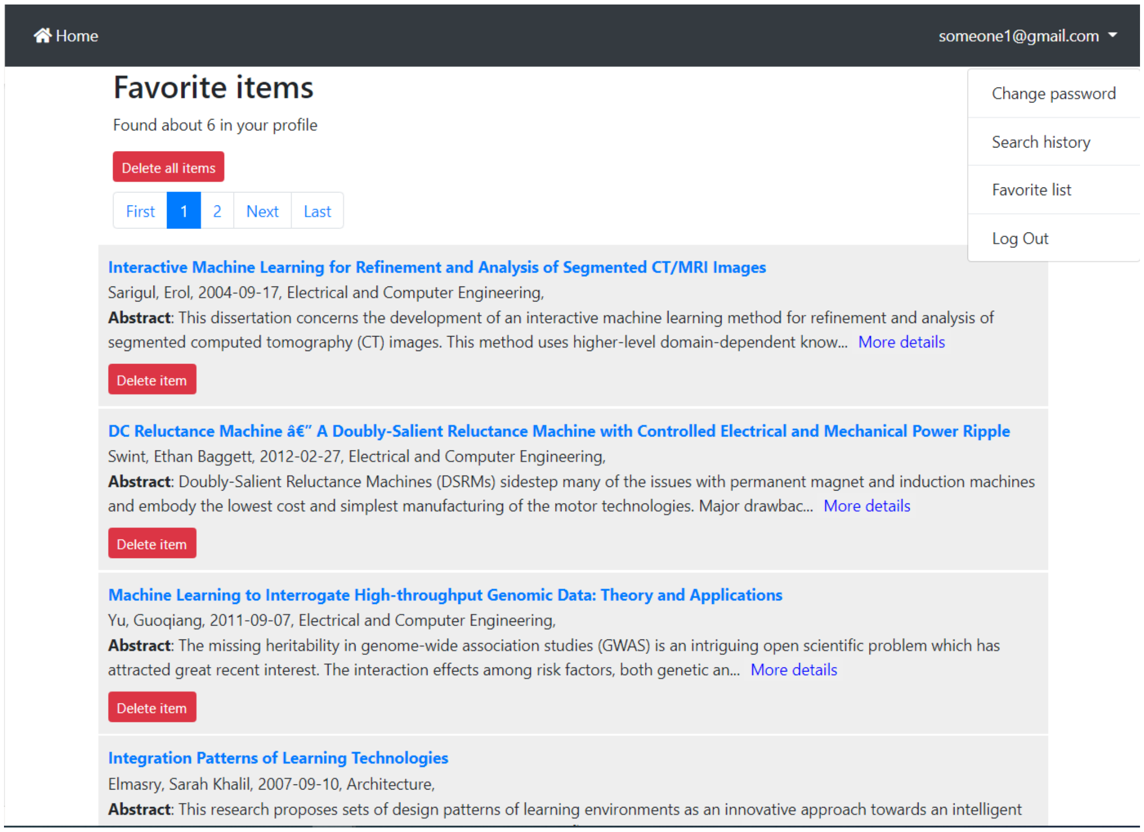
4.11 Highlighting search terms

To highlight the text, the template tags available with Django are used [5] and the code available in the stack overflow [6] is taken as reference and customized for this website. The highlight search text is implemented using the highlightsearch() function in highlight_search.py under users/templatetags/ folder.

4.12 Save items to user's profiles

The favorites items can be saved using the save button present under each ETD documents on the serp page. The save item is implemented using SaveItemView and SaveItemModel. When a user clicks the save button under any of the ETD items, the handle number of the ETD is sent to SaveItemView() function, and the function saves the handle number to the saveitemmodel table in the MySQL database for the current user. The SaveItemView() function uses SaveItemModel to query from the saveitemmodel table in the MySQL database.

To view the saved item lists, the user can go to the favorite list as shown below. The SaveItemView() will get all the handle numbers of the current user from the saveitemmodel table from the MySQL database. Then handlequery() function in elasticsearchETD.py is called for each handle number to get the respective ETD. The favorite items page is paginated using the paginationfun() function.



Favorite items

Found about 6 in your profile

Interactive Machine Learning for Refinement and Analysis of Segmented CT/MRI Images
Sarigul, Erol, 2004-09-17, Electrical and Computer Engineering,
Abstract: This dissertation concerns the development of an interactive machine learning method for refinement and analysis of segmented computed tomography (CT) images. This method uses higher-level domain-dependent know... More details
Delete item

DC Reluctance Machine â€" A Doubly-Salient Reluctance Machine with Controlled Electrical and Mechanical Power Ripple
Swint, Ethan Baggett, 2012-02-27, Electrical and Computer Engineering,
Abstract: Doubly-Salient Reluctance Machines (DSRMs) sidestep many of the issues with permanent magnet and induction machines and embody the lowest cost and simplest manufacturing of the motor technologies. Major drawbac... More details
Delete item

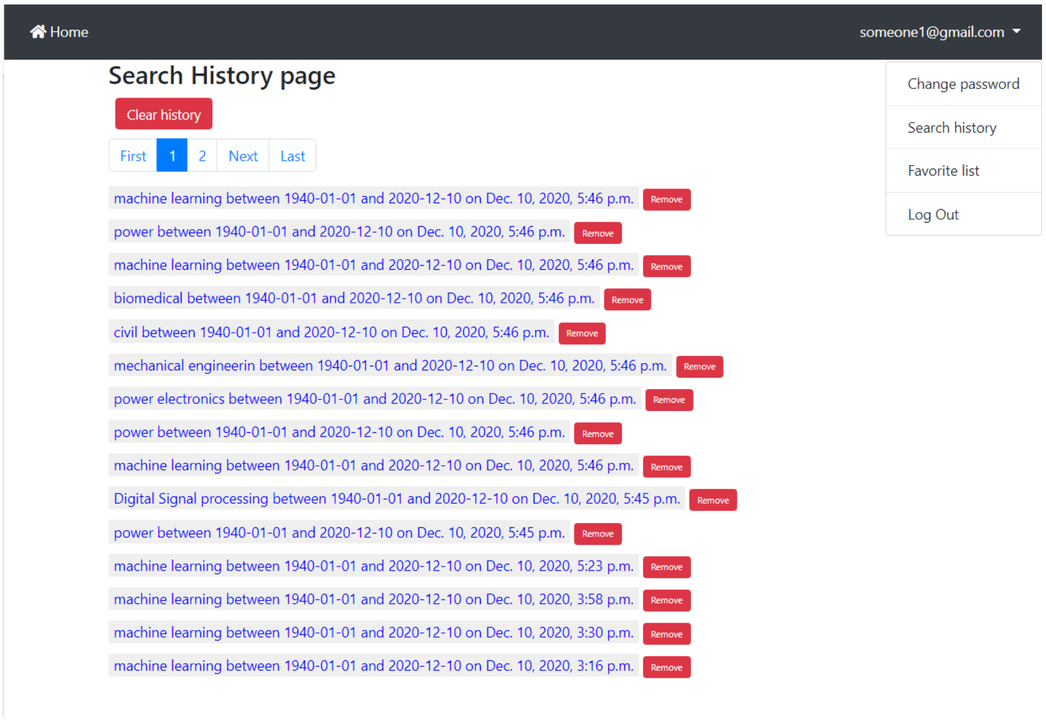
Machine Learning to Interrogate High-throughput Genomic Data: Theory and Applications
Yu, Guoqiang, 2011-09-07, Electrical and Computer Engineering,
Abstract: The missing heritability in genome-wide association studies (GWAS) is an intriguing open scientific problem which has attracted great recent interest. The interaction effects among risk factors, both genetic an... More details
Delete item

Integration Patterns of Learning Technologies
Elmasry, Sarah Khalil, 2007-09-10, Architecture,
Abstract: This research proposes sets of design patterns of learning environments as an innovative approach towards an intelligent

Individual favorite items can be deleted or all the items can be deleted from the saveitemmodel table from the MySQL database. This is implemented in DeleteItemView() function and using SaveItemModel.

4.13 Search history

The user searched in the home page or the serp page is automatically saved and can be accessed as shown below.



Search History page

machine learning between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
power between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
machine learning between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
biomedical between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
civil between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
mechanical engineerin between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
power electronics between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
power between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
machine learning between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:46 p.m. Remove
Digital Signal processing between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:45 p.m. Remove
power between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:45 p.m. Remove
machine learning between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 5:23 p.m. Remove
machine learning between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 3:58 p.m. Remove
machine learning between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 3:30 p.m. Remove
machine learning between 1940-01-01 and 2020-12-10 on Dec. 10, 2020, 3:16 p.m. Remove

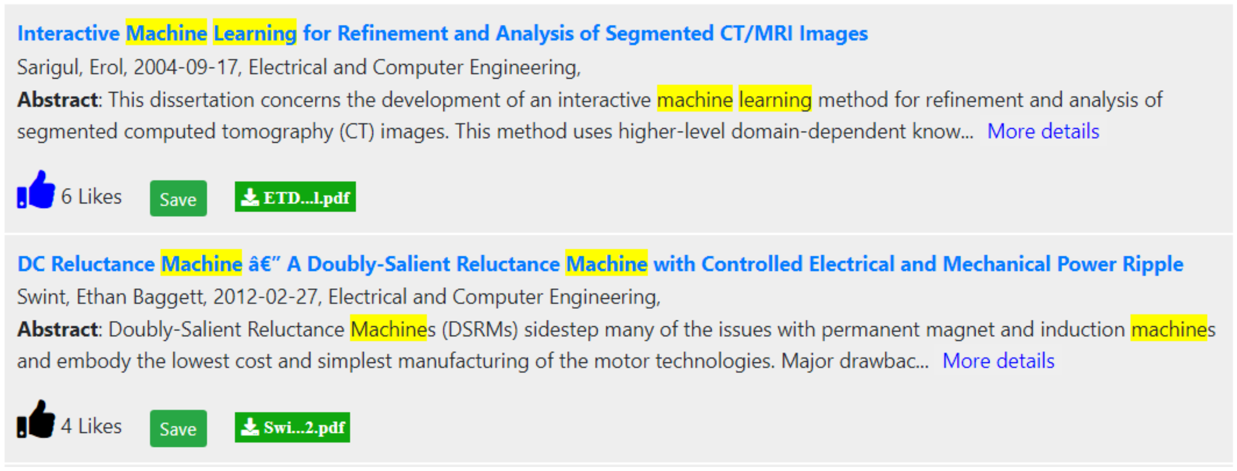
To implement search history: the SearchHistoryView, the SearchHistoryModel and the searchhistorymodel table in the MySQL database. The searchhistorymodel table stores normal search and advance search text details. The entries can be deleted individually or all the searche history can be cleared. This is implemented in ClearHistoryView() function and using SearchHistoryModel.

4.14 Google Map API or Speech-to-text API

The Speech-to-text API is implemented using the HTML5 web search API [7] and it is implemented in advancesearch.html template.

4.15 Like/Unlike an ETD

Any of the ETD can be liked as shown below. The liked ETD is indicated by the blue thumbs-up emoji and the not liked button is indicated by the black thumbs-up emoji.


Interactive Machine Learning for Refinement and Analysis of Segmented CT/MRI Images
Sarigul, Erol, 2004-09-17, Electrical and Computer Engineering,
Abstract: This dissertation concerns the development of an interactive machine learning method for refinement and analysis of segmented computed tomography (CT) images. This method uses higher-level domain-dependent know... More details

6 Likes Save ETD...l.pdf

DC Reluctance Machine â€" A Doubly-Salient Reluctance Machine with Controlled Electrical and Mechanical Power Ripple
Swint, Ethan Baggett, 2012-02-27, Electrical and Computer Engineering,
Abstract: Doubly-Salient Reluctance Machines (DSRMs) sidestep many of the issues with permanent magnet and induction machines and embody the lowest cost and simplest manufacturing of the motor technologies. Major drawbac... More details

4 Likes Save Swi...2.pdf

The like/unlike is implemented using AJAX [8] in javascript. Ajax automatically updates the liked item and number of likes without refreshing the page [9]. This is implemented in LikeItemView() function and the handle number of the liked ETDs are saved to the likeitemmodel table in the MySQL database using LikeItemModel.

It must be noted that the user can save an ETD, like an ETD, upload an ETD only if the user is logged in. The search history is automatically saved only if the user is logged in.

5. Challenges and Lessons

The challenge was to decide which web framework to use. Initially, I have used Laravel but there is a steep learning curve for Laravel, the performance is slow, and fewer user communities compared to Django. After the first milestone, I have shifted to Django, which is completely based on Python and Python has very good packages to implement certain features quickly and there are a large number of user communities compared to Laravel. Further, I am very comfortable with Python and used it quite a bit in the past.

If I get a chance to do it over again, I would also focus on CSS to make the web page look even better.

6. Additional comments

The course is very good and I learned several web development concepts.

References

[1] "Django web framework:," [Online]. Available: https://www.djangoproject.com/.
[2] "Google reCAPTCHA," [Online]. Available: https://www.google.com/recaptcha.
[3] "Paginator package:," [Online]. Available: https://pypi.org/project/paginate/.
[4] " XSS vulnerability:," [Online]. Available: https://docs.djangoproject.com/en/3.1/topics/security/,.
[5] "Bleach package:," [Online]. Available: https://pypi.org/project/django-bleach/.
[6] "Pagination:," [Online]. Available: https://stackoverflow.com/questions/30559086/django-pagination-how-to-limit-the-pages, .

[7] " Template tags:," [Online]. Available: https://docs.djangoproject.com/en/3.1/howto/custom-template-tags/,.

[8] "Template tags code:," [Online]. Available: https://stackoverflow.com/questions/56128231/how-to-highlight-searched-queries-in-result-page-of-django-template, .

[9] "Speech-to-text API:," [Online]. Available: https://www.labnol.org/software/add-speech-recognition-to-website/19989/.

[10] "AJAX in DJango," [Online]. Available: https://simpleisbetterthancomplex.com/tutorial/2016/08/29/how-to-work-with-ajax-request-with-django.html.

[11] "Handling Ajax request in Django," [Online]. Available: https://www.geeksforgeeks.org/handling-ajax-request-in-django/.

[12] "Django security:," [Online]. Available: https://docs.djangoproject.com/en/3.1/topics/security/.