Project: Digital Library Search Engine with Wiki Cards

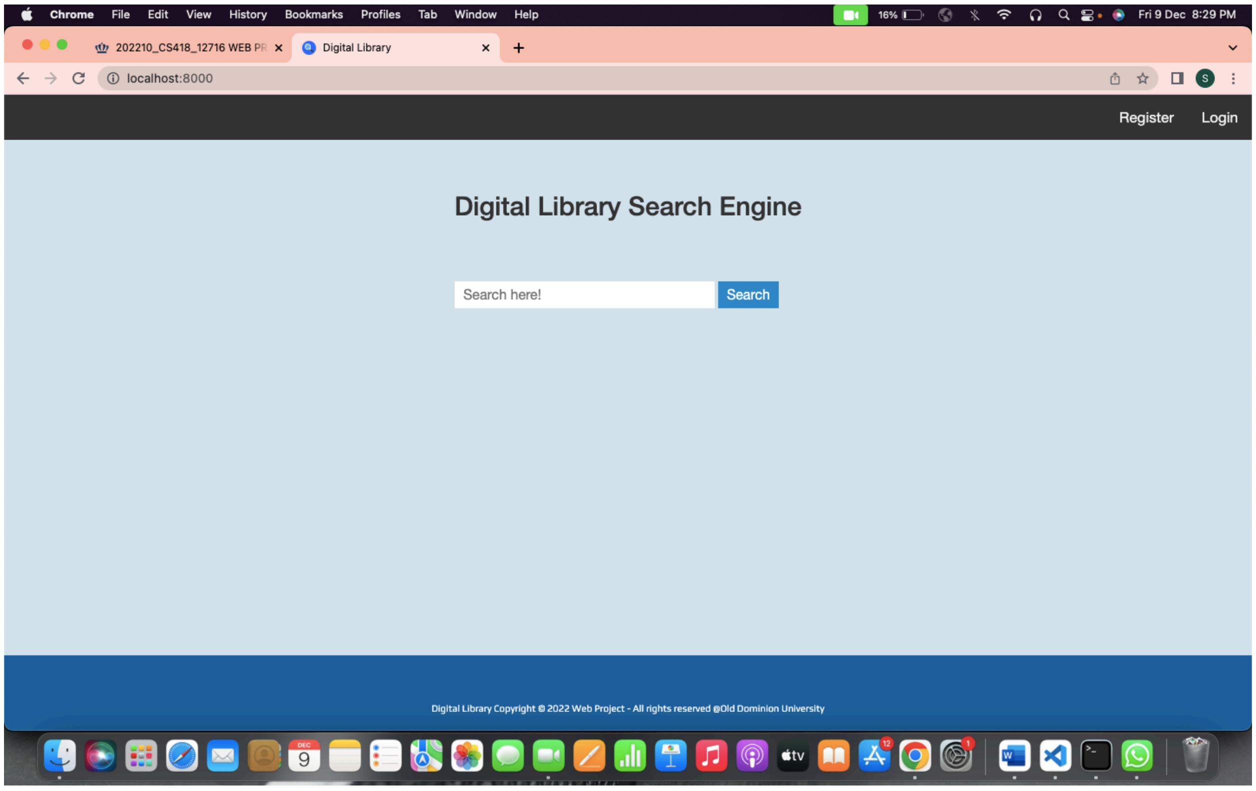
Student name: Sanjana Bolla

Student UIN: 01210054

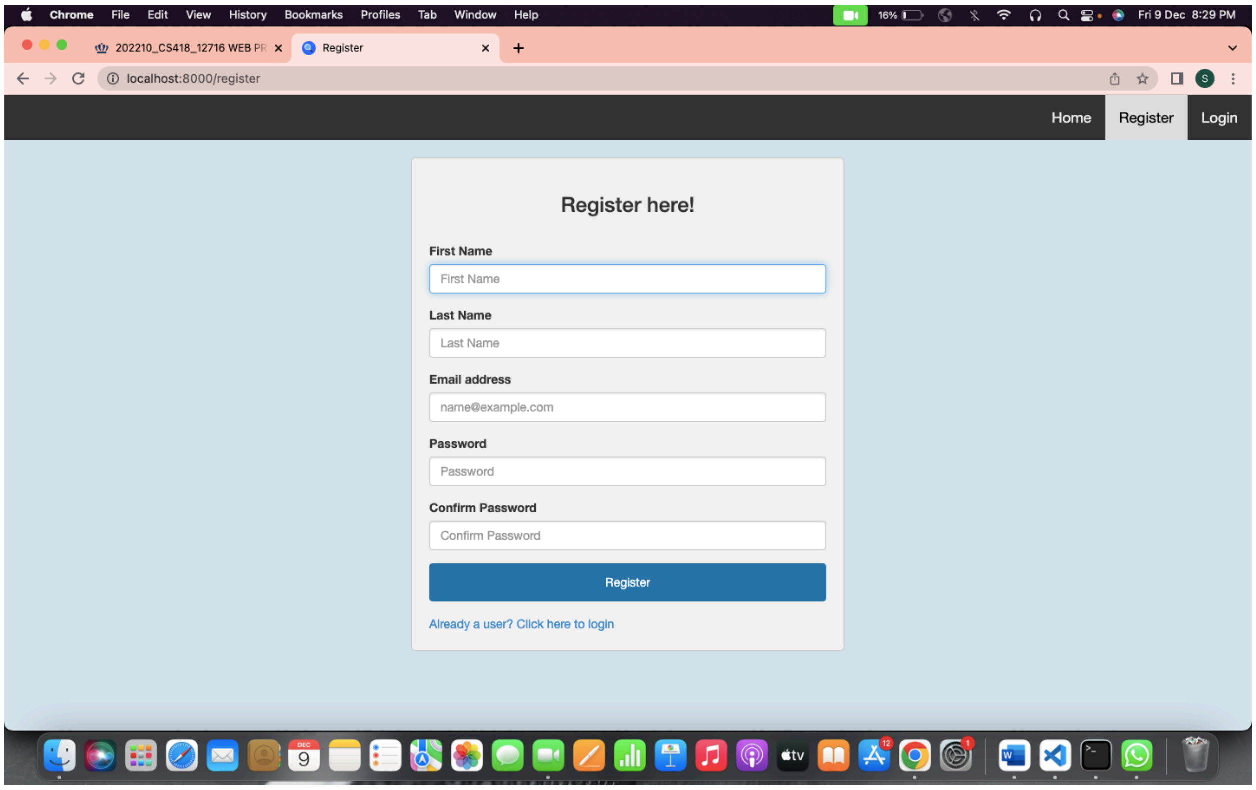
1. Overview

The aim of this project is to build a Digital Library search engine for electronic theses and dissertations (ETDs). The first page of the website contains the Register and the Login options along with a search bar for users who aren't logged in or who don't have any registered accounts. The users who haven't logged in can search for the ETDs and view them, but they aren't allowed to upload any new ETDs. Once the user logs in, they have the search bar on the landing page, where they can search for any keywords and their relevant ETD's titles along with other metadata like author, university, abstract will be printed. The titles that are printed are hyperlinks to the summary page of the ETD. The summary page consists of all the metadata like title, author, advisor , year published, abstract, degree, university and a link to open the pdf of that ETD. The logged in users can also upload a new ETD and search for it in the search bar. I've used HTML, CSS, Bootstrap and JavaScript for the front-end of the website whereas for the backend I've used PHP. This website is built using Laravel, which is a PHP Framework for Web Artisans and Elasticsearch, which is a distributed, full-text search engine with an HTTP web interface and JSON objects. A RESTful API is implemented so that users can query the search engine on the terminal/ command line using curl. Each user is given with a random key and one can query the search engine on terminal using that specific key only. The user can even customize the number results to be printed on the screen.

Home page:


localhost:8000
Register
Login
Digital Library Search Engine
Search here!
Search
Digital Library Copyright © 2022 Web Project - All rights reserved @Old Dominion University

Register page:



Register here!

First Name

First Name

Last Name

Last Name

Email address

name@example.com

Password

Password

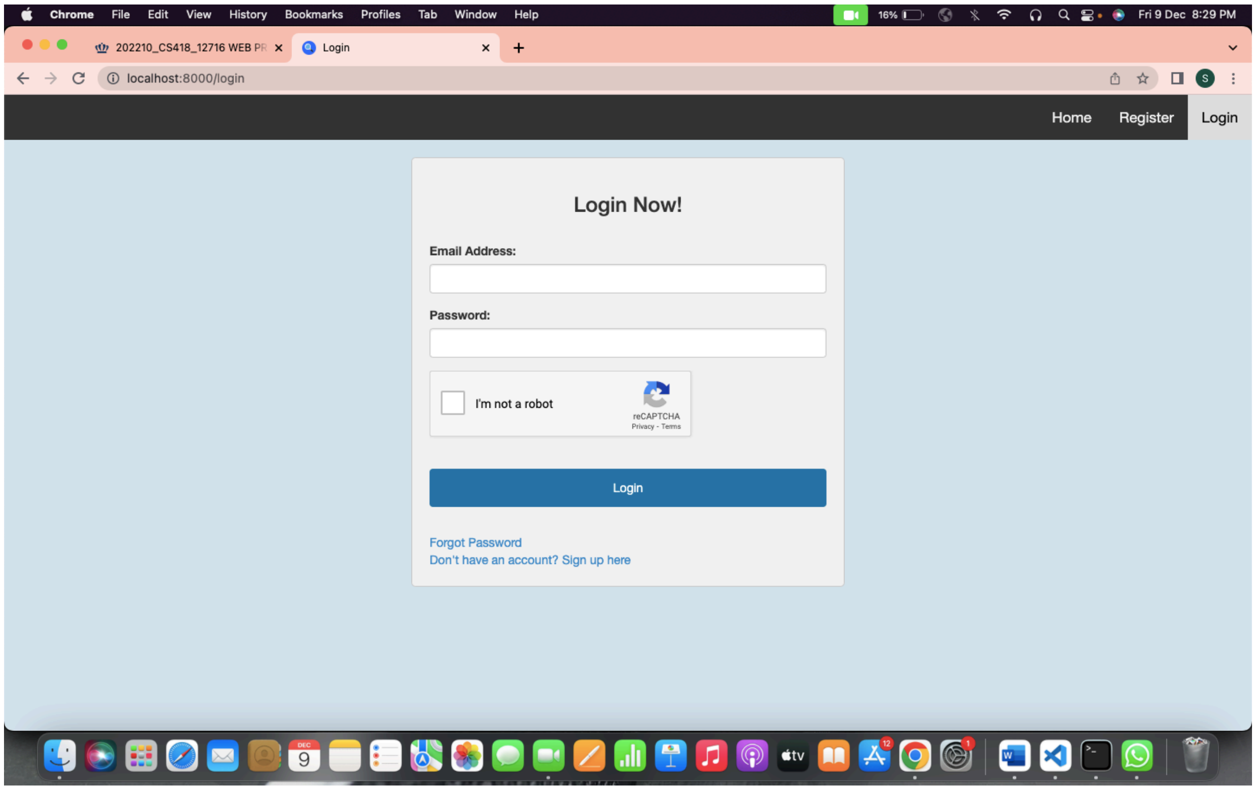
Confirm Password

Confirm Password

Register

Already a user? Click here to login

Login page:



Login Now!

Email Address:

Password:

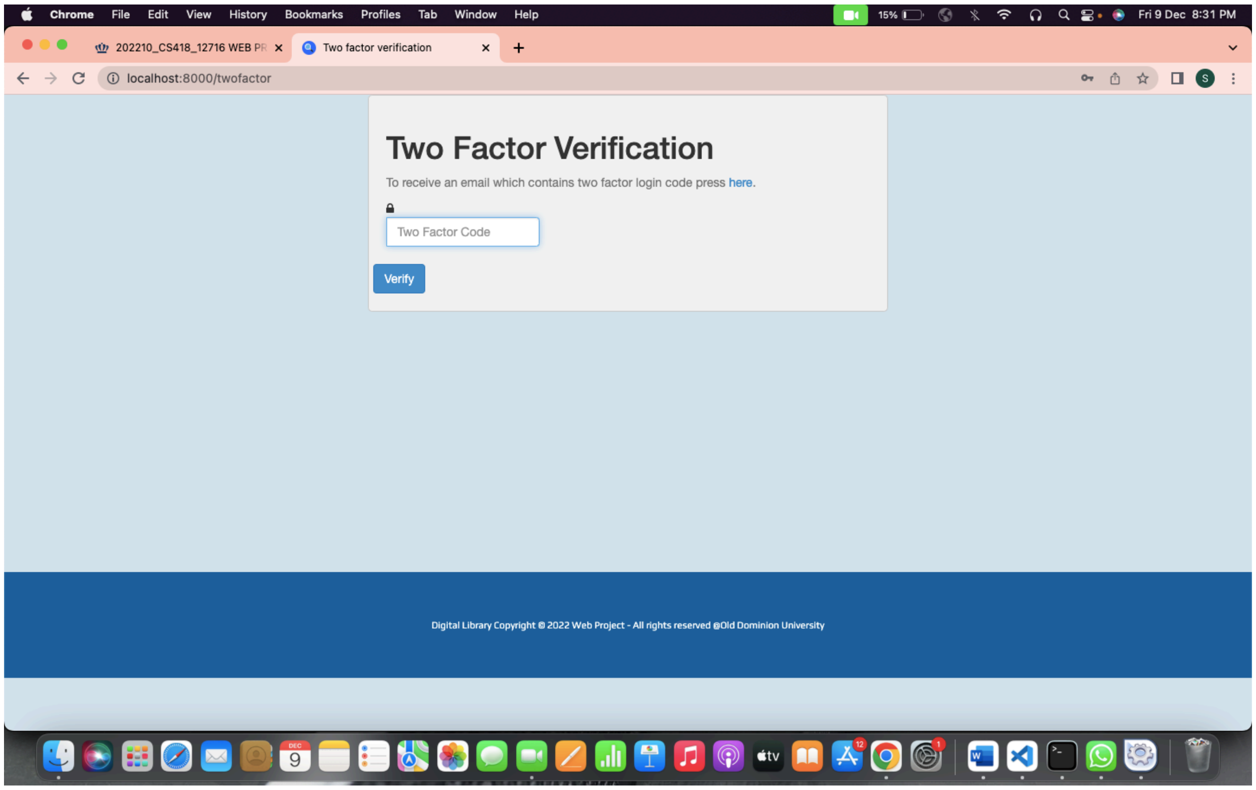
I'm not a robot

Login

Forgot Password

Don't have an account? Sign up here

Two factor verification page:



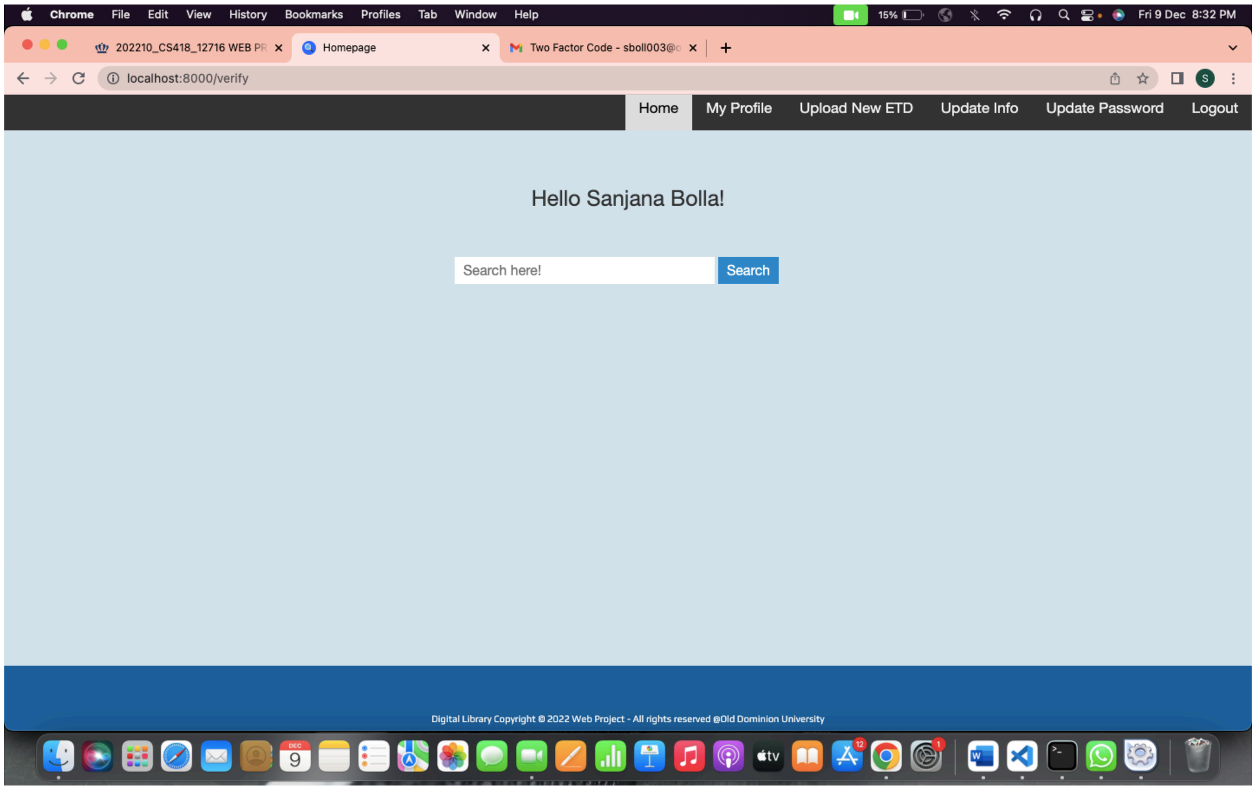
localhost:8000/twofactor

Two Factor Verification

To receive an email which contains two factor login code press here.

Digital Library Copyright © 2022 Web Project - All rights reserved @Old Dominion University

Home page after logging in:

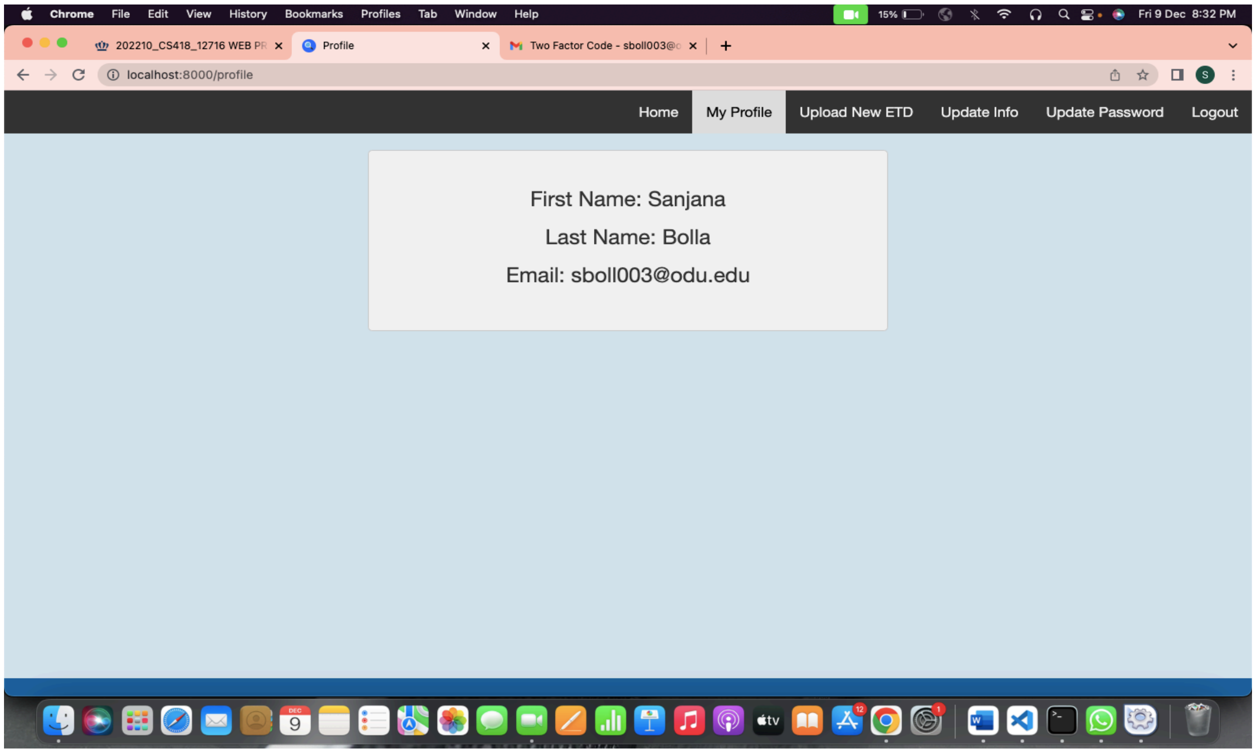


localhost:8000/verify

Hello Sanjana Bolla!

Digital Library Copyright © 2022 Web Project - All rights reserved @Old Dominion University

Profile page:



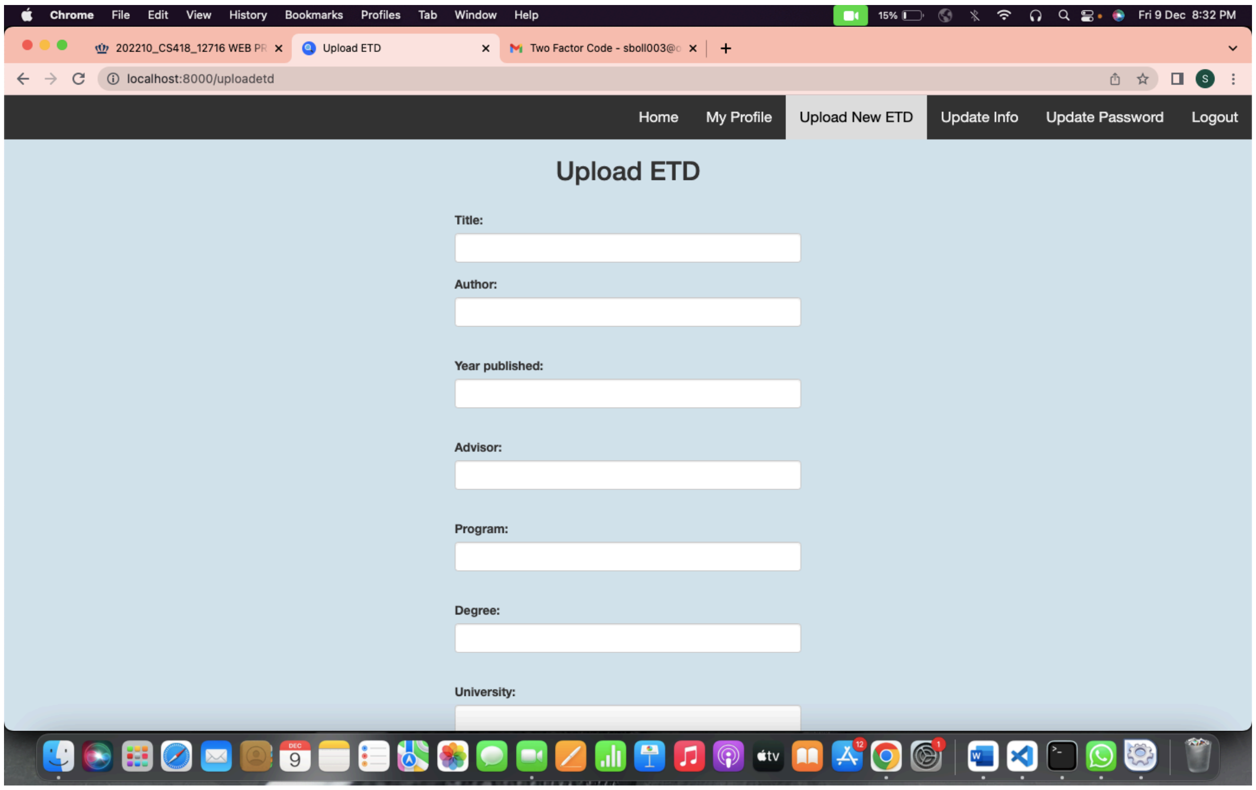
localhost:8000/profile

First Name: Sanjana

Last Name: Bolla

Email: sboll003@odu.edu

Upload new ETD page:



localhost:8000/uploadetd

Upload ETD

Title:

Author:

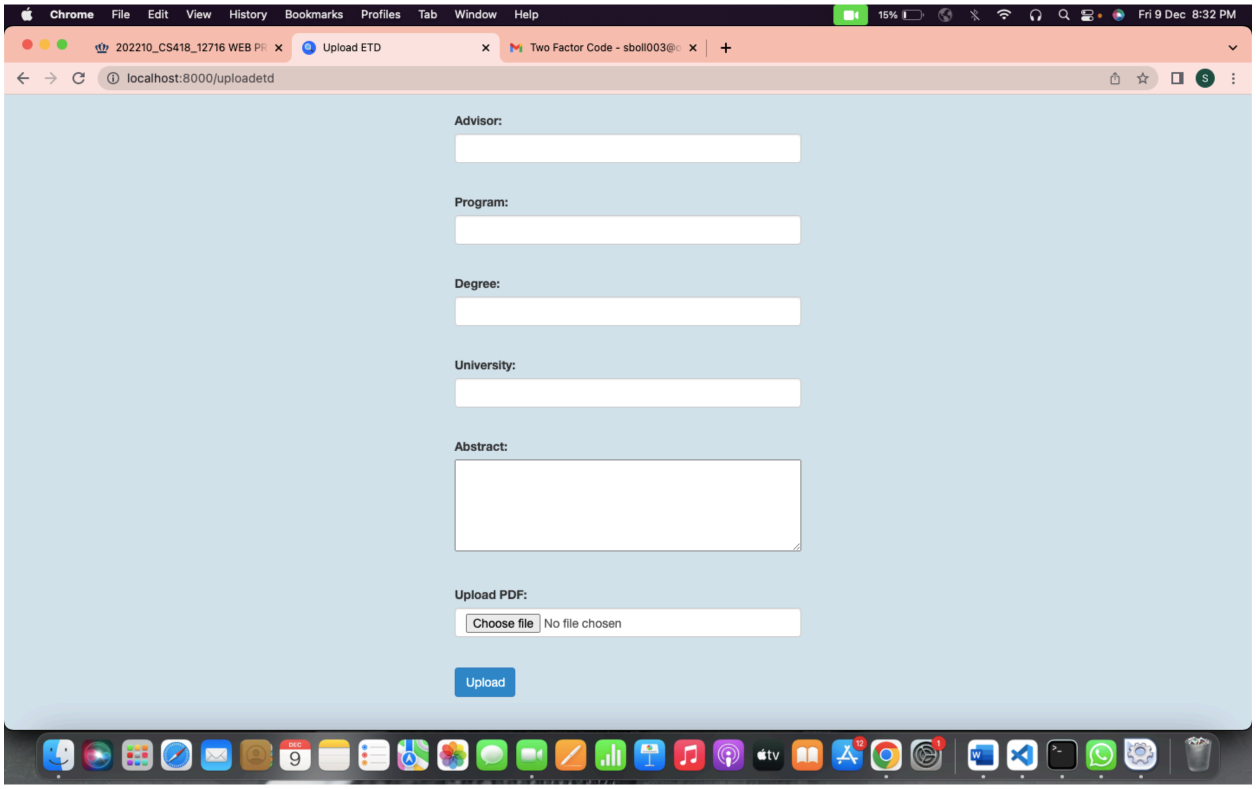
Year published:

Advisor:

Program:

Degree:

University:


localhost:8000/uploadetd

Advisor:

Program:

Degree:

University:

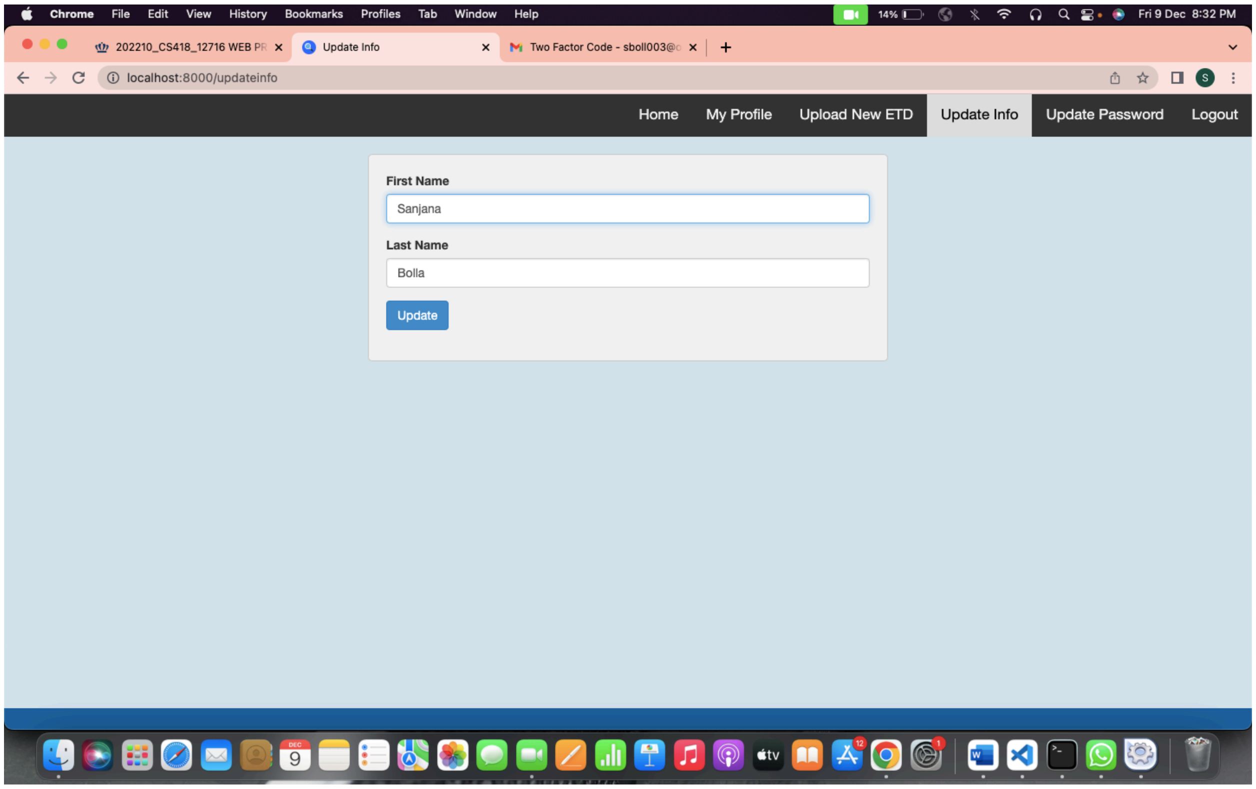
Abstract:

Upload PDF:

Choose file No file chosen

Upload

Update Information page:


localhost:8000/updateinfo

Home | My Profile | Upload New ETD | Update Info | Update Password | Logout

First Name

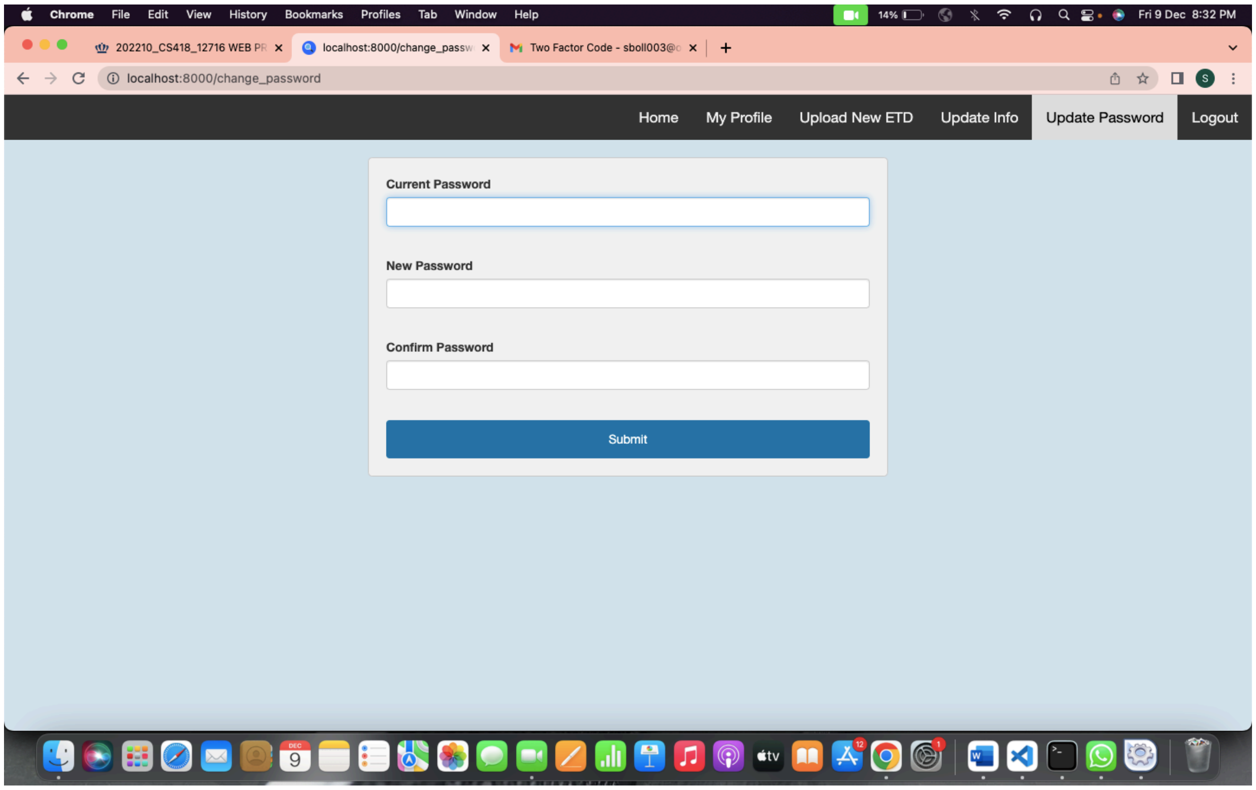
Sanjana

Last Name

Bolla

Update

Update Password page:


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 9 Dec 8:32 PM
202210_CS418_12716 WEB PR
localhost:8000/change_passw
Two Factor Code - sboll003@
localhost:8000/change_password
Home
My Profile
Upload New ETD
Update Info
Update Password
Logout

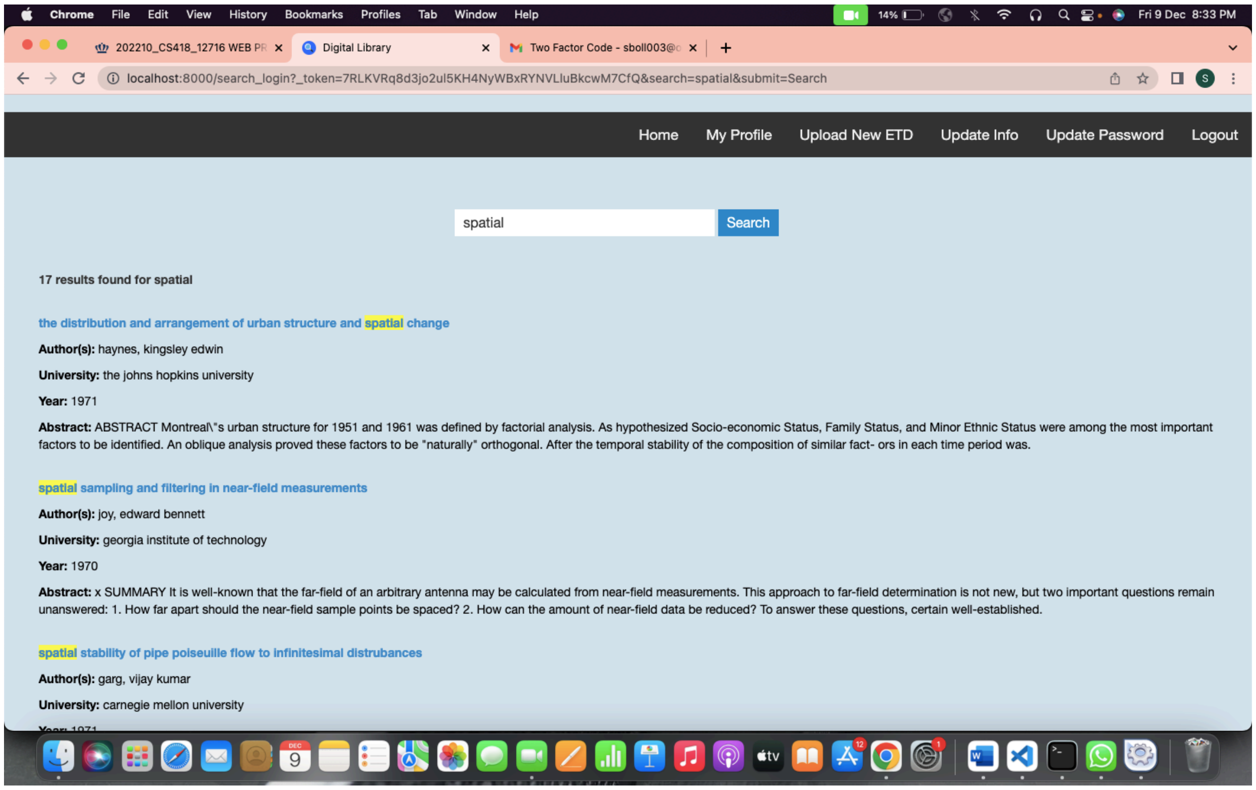
Current Password

New Password

Confirm Password

Submit

Search page:


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 9 Dec 8:33 PM
202210_CS418_12716 WEB PR
Digital Library
Two Factor Code - sboll003@
localhost:8000/search_login?_token=7RLKVRq8d3jo2ul5KH4NyWBxRYNVLluBkcwM7CfQ&search=spatial&submit=Search
Home
My Profile
Upload New ETD
Update Info
Update Password
Logout
spatial
Search

17 results found for spatial

the distribution and arrangement of urban structure and spatial change

Author(s): haynes, kingsley edwin

University: the johns hopkins university

Year: 1971

Abstract: ABSTRACT Montreal\"s urban structure for 1951 and 1961 was defined by factorial analysis. As hypothesized Socio-economic Status, Family Status, and Minor Ethnic Status were among the most important factors to be identified. An oblique analysis proved these factors to be "naturally" orthogonal. After the temporal stability of the composition of similar fact- ors in each time period was.

spatial sampling and filtering in near-field measurements

Author(s): joy, edward bennett

University: georgia institute of technology

Year: 1970

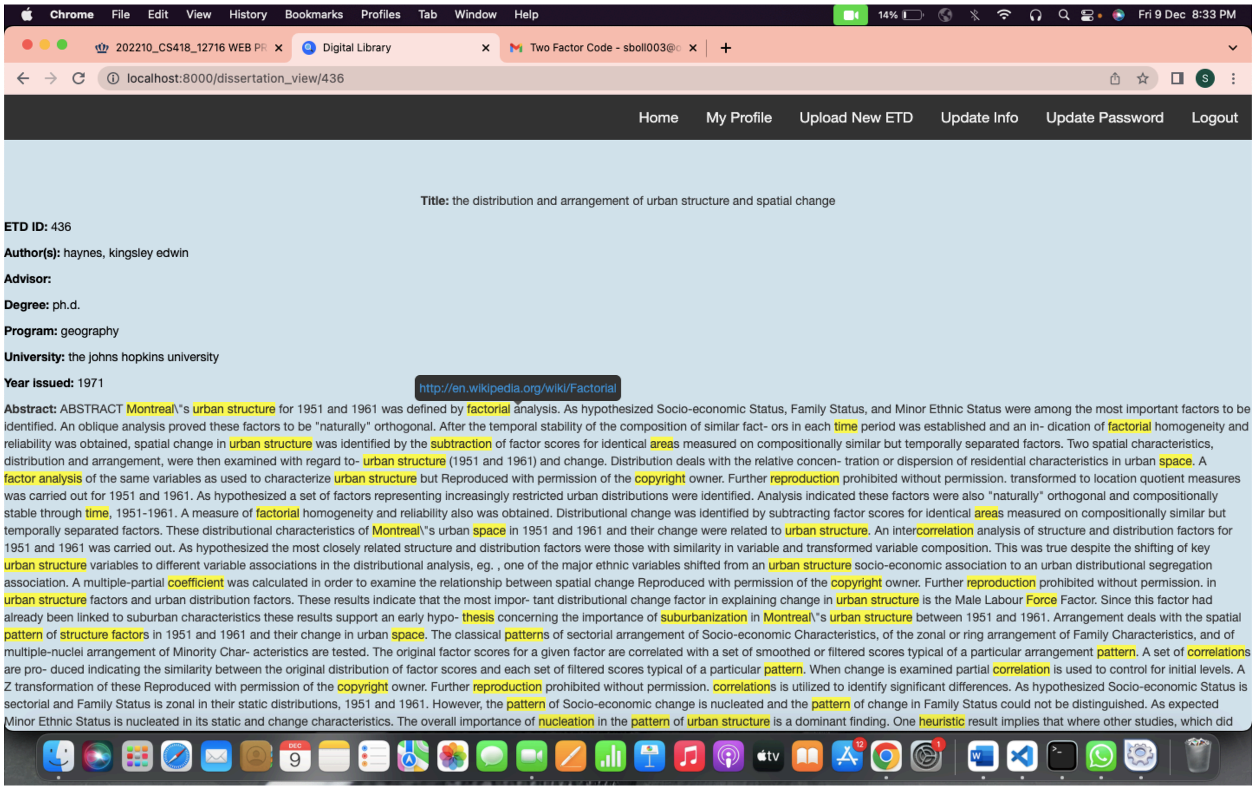
Abstract: x SUMMARY It is well-known that the far-field of an arbitrary antenna may be calculated from near-field measurements. This approach to far-field determination is not new, but two important questions remain unanswered: 1. How far apart should the near-field sample points be spaced? 2. How can the amount of near-field data be reduced? To answer these questions, certain well-established.

spatial stability of pipe poiseuille flow to infinitesimal distrubances

Author(s): garg, vijay kumar

University: carnegie mellon university

Summary page:


localhost:8000/dissertation_view/436
Home
My Profile
Upload New ETD
Update Info
Update Password
Logout
http://en.wikipedia.org/wiki/Factorial

Title: the distribution and arrangement of urban structure and spatial change

ETD ID: 436

Author(s): haynes, kingsley edwin

Advisor:

Degree: ph.d.

Program: geography

University: the johns hopkins university

Year issued: 1971

Abstract: ABSTRACT Montreal\"s urban structure for 1951 and 1961 was defined by factorial analysis. As hypothesized Socio-economic Status, Family Status, and Minor Ethnic Status were among the most important factors to be identified. An oblique analysis proved these factors to be "naturally" orthogonal. After the temporal stability of the composition of similar fact- ors in each time period was established and an in- dication of factorial homogeneity and reliability was obtained, spatial change in urban structure was identified by the subtraction of factor scores for identical areas measured on compositionally similar but temporally separated factors. Two spatial characteristics, distribution and arrangement, were then examined with regard to- urban structure (1951 and 1961) and change. Distribution deals with the relative concen- tration or dispersion of residential characteristics in urban space. A factor analysis of the same variables as used to characterize urban structure but Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. transformed to location quotient measures was carried out for 1951 and 1961. As hypothesized a set of factors representing increasingly restricted urban distributions were identified. Analysis indicated these factors were also "naturally" orthogonal and compositionally stable through time, 1951-1961. A measure of factorial homogeneity and reliability also was obtained. Distributional change was identified by subtracting factor scores for identical areas measured on compositionally similar but temporally separated factors. These distributional characteristics of Montreal\"s urban space in 1951 and 1961 and their change were related to urban structure. An intercorrelation analysis of structure and distribution factors for 1951 and 1961 was carried out. As hypothesized the most closely related structure and distribution factors were those with similarity in variable and transformed variable composition. This was true despite the shifting of key urban structure variables to different variable associations in the distributional analysis, eg. , one of the major ethnic variables shifted from an urban structure socio-economic association to an urban distributional segregation association. A multiple-partial coefficient was calculated in order to examine the relationship between spatial change Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. in urban structure factors and urban distribution factors. These results indicate that the most impor- tant distributional change factor in explaining change in urban structure is the Male Labour Force Factor. Since this factor had already been linked to suburban characteristics these results support an early hypo- thesis concerning the importance of suburbanization in Montreal\"s urban structure between 1951 and 1961. Arrangement deals with the spatial pattern of structure factors in 1951 and 1961 and their change in urban space. The classical patterns of sectorial arrangement of Socio-economic Characteristics, of the zonal or ring arrangement of Family Characteristics, and of multiple-nuclei arrangement of Minority Char- acteristics are tested. The original factor scores for a given factor are correlated with a set of smoothed or filtered scores typical of a particular arrangement pattern. A set of correlations are pro- duced indicating the similarity between the original distribution of factor scores and each set of filtered scores typical of a particular pattern. When change is examined partial correlation is used to control for initial levels. A Z transformation of these Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. correlations is utilized to identify significant differences. As hypothesized Socio-economic Status is sectorial and Family Status is zonal in their static distributions, 1951 and 1961. However, the pattern of Socio-economic change is nucleated and the pattern of change in Family Status could not be distinguished. As expected Minor Ethnic Status is nucleated in its static and change characteristics. The overall importance of nucleation in the pattern of urban structure is a dominant finding. One heuristic result implies that where other studies, which did

2. Milestone Accomplishments

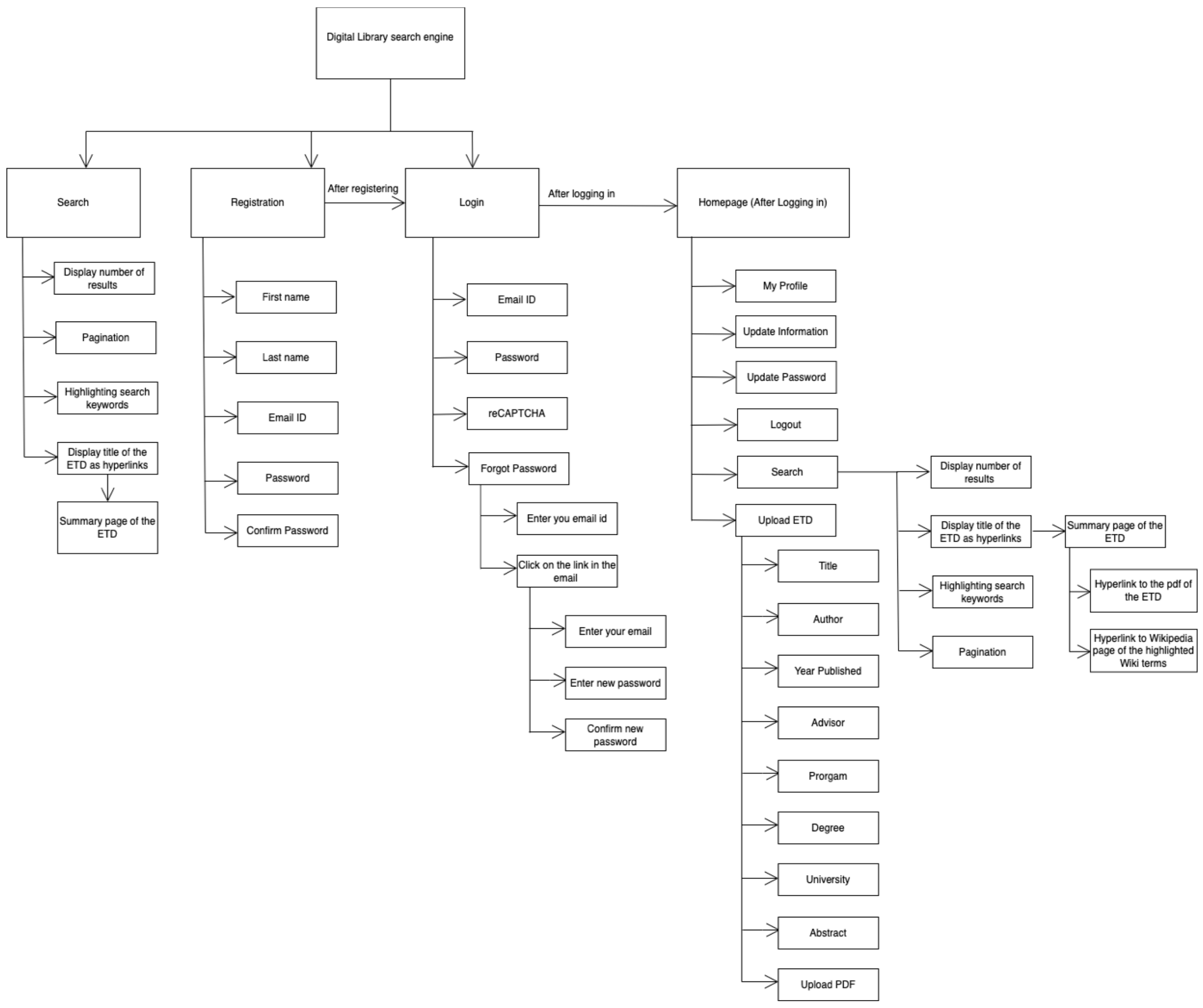
Table 1: Status of milestone specifications.

Fulfilled	Milestone	Feature#	Specification Description
Yes	1	1	Users can register new accounts using email addresses.
Yes	1	2	Users are identified by email address.
Yes	1	3	Password is encrypted before storing in the database.
Yes	1	4	Users cannot register duplicate accounts using the same email address.
Yes	1	5	Users can log into the website using the accounts that they registered with.
Yes	1	6	Users can reset their passwords in case they forget it.
Yes	1	7	Users can update the password after they login.
Yes	1	8	2-factor-authentication is used when the user tries to login. The verification code is sent to the email using which the user is logging in.
Yes	1	9	The website has a homepage where users are given options to view their profile, update their information, like first name, last name and password, and log out of the account.
Yes	1	10	Search box in the landing page after the user logs into their account.
Yes	2	11	The website should index all documents in the ETD500 repository and create a JSON object.
Yes	2	12	Users can query the search engine without logging in.
Yes	2	13	The search engine accepts a text query in the search box and return results on the search engine result page (SERP).

Yes	2	14	Each item in SERP should link to a page for a document.
Yes	2	15	The search engine should display the number of returned items on top of search results.
Yes	2	16	The search engine should display the actual keywords after filtering and a search box on the top.
Partially completed	2	17	The search results should be paginated.
Yes	2	18	The query terms should be highlighted in the search results.
Yes	2	19	Use Wikifier to obtain Wikipedia terms appearing in ETD abstract and store the terms and the Wikipedia URLs in the JSON object.
Yes	2	20	When a user clicks a search result, he should see a summary page showing all metadata fields (including the abstract) of an ETD.
Yes	2	21	A regular user should be able to insert a new document, including key metadata (title, author, year, advisor, university, degree, program) and upload the PDF.
Yes	2	22	Search the title of the new paper and it should show up in the search results.
Yes	3	23	Add reCAPTCHA to the login page.
Yes	3	24	The search engine can prevent XSS vulnerability by sanitizing the JavaScript in the queries and showing the actual term on top of the search results.
Yes	3	25	The search engine can highlight the matched terms in the search result page.
Yes	3	26	In the summary page, highlight the wiki terms and when the user moves the mouse over a highlighted term, a popup window is shown to display the term and a clickable link to the Wikipedia page.
Yes	3	27	A RESTful API is implemented so that users can query the search engine programmatically and obtain a customizable number of results. The system will generate a random key to the user and the user must use this key to make queries to the search engine.
Yes	3	28	Design and implement a favicon for the website.
Yes	3	29	Design and implement a consistent footer that appears at the bottom of all site pages.
Yes	3	30	A browser compatible design: when shrinking the browser window, the layouts of panels can automatically change from a grid to vertical layout.

3. Architecture

In the first page of the Digital Library Search Engine website, the user has two options: Register and Login and there's also a search box for the users who haven't registered or logged in yet. They can search for any keyword(s) and the results will be displayed below the search box. The number of results is displayed to the user and if there are more than 10 results, then the results are paginated. The keyword searched for will be highlighted in the search results. In the search engine results page, the titles of the ETDs are displayed as hyperlinks to the summary page of that ETD. The summary page consists of all the metadata like title, author, advisor , year published, abstract, degree, university and a link to open the pdf of that ETD. The terms identified by the Wikifier are highlighted in the summary page and each highlighted term has a popup window which has a hyperlink to the Wikipedia page of that particular term. New users can register new accounts using Register and already registered users can login using the Login option. After logging in, the users are presented with another search box which has the same functionality as the previous one. The authenticated users can upload new ETD by entering the details like title, author, year published, advisor, program, degree university, abstract and upload a pdf file for the ETD. The Upload New ETD option is available only to the logged in users. There are other options available for the logged in users like: My Profile, update information, update password and logout from their profile.



4. Database Design

MySQL has been used for the database design. A database called **web** is created, in which a table called users is created. In the users table, the details of all the registered users like first name, last name, email id, password, etc., are stored. The password is encrypted before storing it in the table as you can see in the table below. The two-factor code is also stored in the table along with its expiry (two_factor_expires_at). There is a remember_token field, which is used to store the random key generated for each user, which is used to query the search engine using curl. The created_at and updated_at fields will let the admin know when that user account is created at and last updated at. The below table depicts the structure of the users table which includes the field's name, type, key and an example of the field which is taken from the database.

Field	Type	Key	Example
id	bigint(20)	Primary	1
firstname	varchar(255)		Sanjana
lastname	varchar(255)		Bolla
email	varchar(255)	unique	sboll003@odu.edu
password	varchar(255)		$2y$10$C4nl0VzVNaPid6nGlT0OIe
two_factor_code	varchar(255)		5887
two_factor_expires_at	datetime		2022-10-04 06:06:49
remember_token	varchar(100)		2Hstbe45edsrh3453fsefwg5yjjlpwl
created_at	timestamp		2022-10-06 19:02:34

updated_at	timestamp		2022-10-06 19:05:11

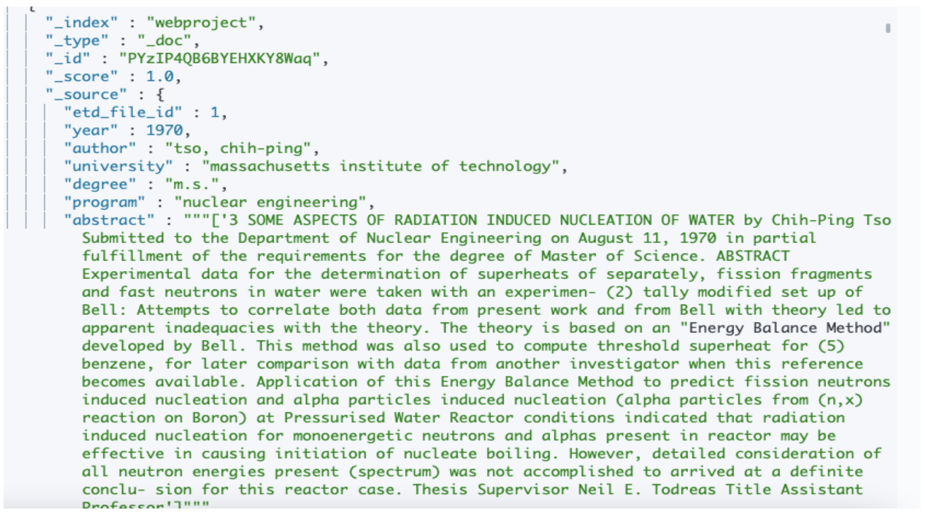
5. Elasticsearch Design

Elasticsearch mapping is the process of defining how a document and the fields it contains are stored and indexed. The mapping of the fields is shown in the table below:

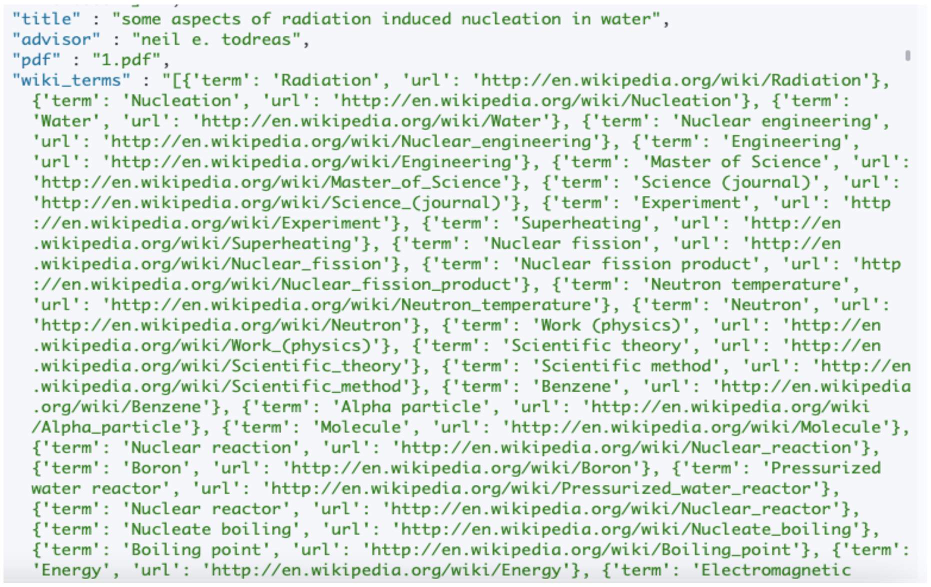
Field	Type
etd_file_id	long
title	text
author	text
advisor	text
year	long
abstract	text
degree	keyword
program	keyword
university	keyword
pdf	keyword
wiki_terms	text

Since etd_file_id and year are numbers, I have used long as the field type. Title, author, advisor, abstract and wiki_terms have been defined as text field type whereas degree, university, program and pdf are defined as keyword field type.

A screenshot of the example data is shown below:



```
"_index" : "webproject",
"_type" : "_doc",
"_id" : "PYzIP4QB6BYEHXKY8Waq",
"_score" : 1.0,
"_source" : {
  "etd_file_id" : 1,
  "year" : 1970,
  "author" : "tso, chih-ping",
  "university" : "massachusetts institute of technology",
  "degree" : "m.s.",
  "program" : "nuclear engineering",
  "abstract" : """['3 SOME ASPECTS OF RADIATION INDUCED NUCLEATION OF WATER by Chih-Ping Tso
    Submitted to the Department of Nuclear Engineering on August 11, 1970 in partial
    fulfillment of the requirements for the degree of Master of Science. ABSTRACT
    Experimental data for the determination of superheats of separately, fission fragments
    and fast neutrons in water were taken with an experimen- (2) tally modified set up of
    Bell: Attempts to correlate both data from present work and from Bell with theory led to
    apparent inadequacies with the theory. The theory is based on an "Energy Balance Method"
    developed by Bell. This method was also used to compute threshold superheat for (5)
    benzene, for later comparison with data from another investigator when this reference
    becomes available. Application of this Energy Balance Method to predict fission neutrons
    induced nucleation and alpha particles induced nucleation (alpha particles from (n,x)
    reaction on Boron) at Pressurised Water Reactor conditions indicated that radiation
    induced nucleation for monoenergetic neutrons and alphas present in reactor may be
    effective in causing initiation of nucleate boiling. However, detailed consideration of
    all neutron energies present (spectrum) was not accomplished to arrived at a definite
    conclu- sion for this reactor case. Thesis Supervisor Neil E. Todreas Title Assistant
    Professor']"""
```



```
"title" : "some aspects of radiation induced nucleation in water",
"advisor" : "neil e. todreas",
"pdf" : "1.pdf",
"wiki_terms" : "[{'term': 'Radiation', 'url': 'http://en.wikipedia.org/wiki/Radiation'},
  {'term': 'Nucleation', 'url': 'http://en.wikipedia.org/wiki/Nucleation'}, {'term':
  'Water', 'url': 'http://en.wikipedia.org/wiki/Water'}, {'term': 'Nuclear engineering',
  'url': 'http://en.wikipedia.org/wiki/Nuclear_engineering'}, {'term': 'Engineering',
  'url': 'http://en.wikipedia.org/wiki/Engineering'}, {'term': 'Master of Science', 'url':
  'http://en.wikipedia.org/wiki/Master_of_Science'}, {'term': 'Science (journal)', 'url':
  'http://en.wikipedia.org/wiki/Science_(journal)'}, {'term': 'Experiment', 'url': 'http
  ://en.wikipedia.org/wiki/Experiment'}, {'term': 'Superheating', 'url': 'http://en
  .wikipedia.org/wiki/Superheating'}, {'term': 'Nuclear fission', 'url': 'http://en
  .wikipedia.org/wiki/Nuclear_fission'}, {'term': 'Nuclear fission product', 'url': 'http
  ://en.wikipedia.org/wiki/Nuclear_fission_product'}, {'term': 'Neutron temperature',
  'url': 'http://en.wikipedia.org/wiki/Neutron_temperature'}, {'term': 'Neutron', 'url':
  'http://en.wikipedia.org/wiki/Neutron'}, {'term': 'Work (physics)', 'url': 'http://en
  .wikipedia.org/wiki/Work_(physics)'}, {'term': 'Scientific theory', 'url': 'http://en
  .wikipedia.org/wiki/Scientific_theory'}, {'term': 'Scientific method', 'url': 'http://en
  .wikipedia.org/wiki/Scientific_method'}, {'term': 'Benzene', 'url': 'http://en.wikipedia
  .org/wiki/Benzene'}, {'term': 'Alpha particle', 'url': 'http://en.wikipedia.org/wiki
  /Alpha_particle'}, {'term': 'Molecule', 'url': 'http://en.wikipedia.org/wiki/Molecule'},
  {'term': 'Nuclear reaction', 'url': 'http://en.wikipedia.org/wiki/Nuclear_reaction'},
  {'term': 'Boron', 'url': 'http://en.wikipedia.org/wiki/Boron'}, {'term': 'Pressurized
  water reactor', 'url': 'http://en.wikipedia.org/wiki/Pressurized_water_reactor'},
  {'term': 'Nuclear reactor', 'url': 'http://en.wikipedia.org/wiki/Nuclear_reactor'},
  {'term': 'Nucleate boiling', 'url': 'http://en.wikipedia.org/wiki/Nucleate_boiling'},
  {'term': 'Boiling point', 'url': 'http://en.wikipedia.org/wiki/Boiling_point'}, {'term':
  'Energy', 'url': 'http://en.wikipedia.org/wiki/Energy'}, {'term': 'Electromagnetic
```

6. Implementation

Register:

The users can register new accounts using the register page. Users should enter details like first name, last name, email address and password in order to register an account. If a user already exists with the same email address, then an alert will be displayed to the user stating that the email is taken, please try using another email as the users are identified by their email address. Once the user is successfully registered, their details will be stored in the database and the user will be redirected to the login page in order to login with their credentials. The code for implementing the register page can be found in Resources->Views->register.blade.php and app->Http->Controllers->RegisterController.php files.

Users are identified using their email:

The users are uniquely identified using their email address and this functionality is written in the login_auth function in the app->Http->Controllers->MainController.php file where the entered email is verified by checking it against the email in the database. Once the details are validated, then the user logs in and will be redirected to the homepage(index.php).

Users cannot register duplicate accounts:

While registering an account, if a user already exists with the same email address, then an alert will be displayed to the user stating that the email is taken, as the users are identified by their email address. The code for implementing this functionality is written in app->Http->Controllers->RegisterController.php and app->Http->Requests->RegisterRequest.php files. The rules which make sure that the email address is unique is written in the RegisterRequest.php file. The function which makes sure that the user is created in the database is written in the RegisterController.php file.

Password Encryption:

The password is encrypted before storing the database and this is done using the Laravel's Hash facade which provides the Bcrypt hashing. The code to implement this encryption is written is app->Models->User.php file in the setPasswordAttribute function.

Login:

Users can login using the email address and password associated with their account. If they email address and password are validated against the database and if they correct, then the user will be redirected to a two-factor authentication page else an alert will be displayed to the user stating that the details are incorrect. The code to implement the login page is written in Resources->Views->login.blade.php and the code to authenticate the login credentials is written in app->Http->Controllers->MainController.php in the login_auth function.

Two-factor authentication:

Once the user's login credentials are authenticated, the user will be redirected to the two-factor authentication page where they will have to click on the link to receive an authentication code to the email they have used to login. The user will then receive an email which has the verification code. The two-factor code is generated randomly using the generateTwoFactorCode function in the app->Models->User.php file. The user must enter the code in the two-factor authentication page and submit it in order to redirect to the homepage of the user. The code for implementing this is written in Resources->Views->twofactor.blade.php and app->Http->Controllers->TwoFactorController.php files.

Reset Password:

In case the user forgot their password, they can reset it using the forgot password option in the Login page. In the forgot password page, the user will have to enter their email id and a verification link will be sent to that email id using which the user can reset their password. Upon clicking that verification link, the user must enter their email id, new password and confirmation of the new password. The code for implementing this is written in Resources->Views->reset.blade.php , Resources->Views->verify.blade.php , Resources->Views->email.blade.php , app->Http->Controllers->ForgotPasswordController.php and app->Http->Controllers->ResetPasswordController.php files. The function which makes sure that the new password is encrypted and has the validation rules is written in the ResetPasswordController.php file. The ForgotPasswordController.php has the functions which are responsible for sending the verification email to the user. The verify.blade.php is the page responsible for the message displayed in the email and the email.blade.php is responsible for the displaying the page where the user must enter their email in order to receive the verification link. The reset.blade.php is opened when the user clicks on the verification link in the email.

Update Password:

The user has the option to update their password once they login. The update password link will redirect the user to the change password page where the user is supposed to enter their current password, new password and confirmation of the new password. If the current password doesn't match with the one in the database, then an alert will be displayed to the user stating that the current password is incorrect, and this functionality is written in the update_password function in the MainController.php file. Once the password is updated, the user will be redirected to the homepage (index.php) and a message will be displayed on screen stating that the password is updated successfully. The code for implementing this is written in Resources->Views->change_password.blade.php and app->Http->Controllers->MainController.php files.

Homepage:

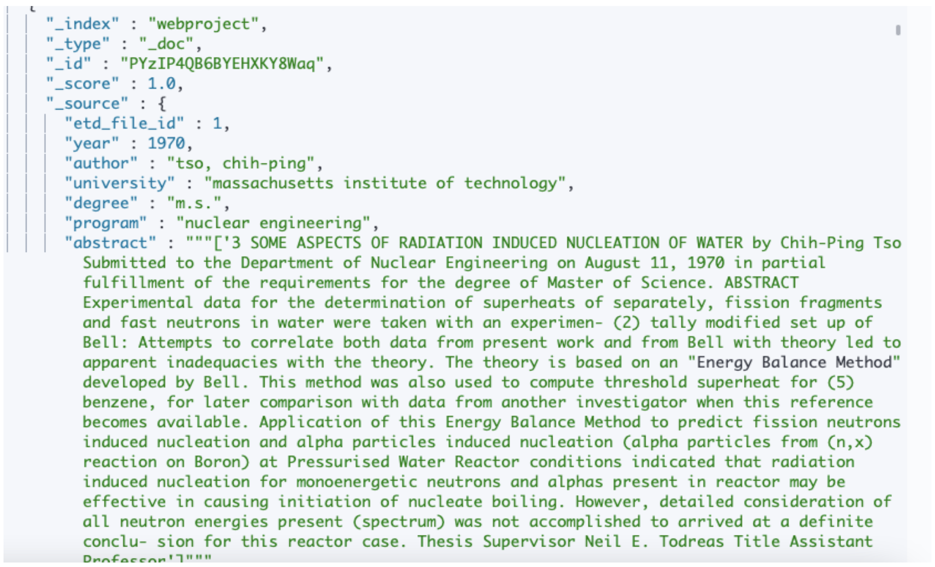
The homepage which is the landing page after the user logs in has options like My profile, Update Info, Update password and logout. In My Profile page, the user can view their details like first name, last name and email id associated with the account. The user can update their first name and last name in the Update Info page. Once the user's information is updated, they will be redirected to the homepage(index.php). The Update password page is used to update the user's password where they will have to enter their current password, new password and confirmation of the new password in order to update their password. The code to implement the homepage is in Resources->Views->index.blade.php and app->Http->Controllers->MainController.php files. The code to implement the information update is written in the Resources->Views->updateinfo.blade.php and app->Http->Controllers->MainController.php files. The code to view the My Profile page is written in Resources->Views->profile.blade.php

Search box:

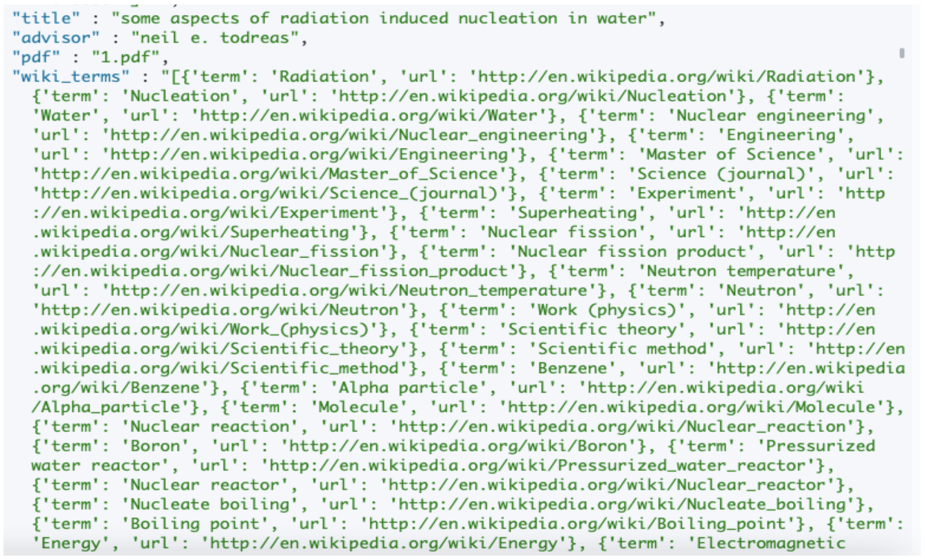
There is a search box in the homepage after the user logs in. The code to implement the search box is in the Resources->Views->index.blade.php file. The search box is not functional right now in this milestone, but I plan to use that in order search for abstracts for the Electronic Theses and Dissertations.

Index documents and generate json object:

I've used Elasticsearch, which is a distributed, full-text search engine with an HTTP web interface and JSON objects, in this project to index all the documents in the ETD500 repository. The metadata_abstract.csv file, uploaded in the GitHub was merged with the wikifier terms and URLs that were generated using the wikifier_updated.py code. The final merged file is then indexed in the Elasticsearch by uploading the .csv file in the elasticsearch. I've named the index as webproject. This index contains the 500 ETD's metadata, abstract and the wikifier terms and URLs in the form of JSON object. A sample screenshot of the json object taken from running the command: GET /webproject/_search is included below for reference.



```
    "_index" : "webproject",
    "_type" : "_doc",
    "_id" : "PYzIP4QB6BYEHXKY8Waq",
    "_score" : 1.0,
    "_source" : {
      "etd_file_id" : 1,
      "year" : 1970,
      "author" : "tso, chih-ping",
      "university" : "massachusetts institute of technology",
      "degree" : "m.s.",
      "program" : "nuclear engineering",
      "abstract" : """['3 SOME ASPECTS OF RADIATION INDUCED NUCLEATION OF WATER by Chih-Ping Tso
        Submitted to the Department of Nuclear Engineering on August 11, 1970 in partial
        fulfillment of the requirements for the degree of Master of Science. ABSTRACT
        Experimental data for the determination of superheats of separately, fission fragments
        and fast neutrons in water were taken with an experimen- (2) tally modified set up of
        Bell: Attempts to correlate both data from present work and from Bell with theory led to
        apparent inadequacies with the theory. The theory is based on an "Energy Balance Method"
        developed by Bell. This method was also used to compute threshold superheat for (5)
        benzene, for later comparison with data from another investigator when this reference
        becomes available. Application of this Energy Balance Method to predict fission neutrons
        induced nucleation and alpha particles induced nucleation (alpha particles from (n,x)
        reaction on Boron) at Pressurised Water Reactor conditions indicated that radiation
        induced nucleation for monoenergetic neutrons and alphas present in reactor may be
        effective in causing initiation of nucleate boiling. However, detailed consideration of
        all neutron energies present (spectrum) was not accomplished to arrived at a definite
        conclu- sion for this reactor case. Thesis Supervisor Neil E. Todreas Title Assistant
        Professor']"""
```



```
"title" : "some aspects of radiation induced nucleation in water",
"advisor" : "neil e. todreas",
"pdf" : "1.pdf",
"wiki_terms" : "[{'term': 'Radiation', 'url': 'http://en.wikipedia.org/wiki/Radiation'},
  {'term': 'Nucleation', 'url': 'http://en.wikipedia.org/wiki/Nucleation'}, {'term':
  'Water', 'url': 'http://en.wikipedia.org/wiki/Water'}, {'term': 'Nuclear engineering',
  'url': 'http://en.wikipedia.org/wiki/Nuclear_engineering'}, {'term': 'Engineering',
  'url': 'http://en.wikipedia.org/wiki/Engineering'}, {'term': 'Master of Science', 'url':
  'http://en.wikipedia.org/wiki/Master_of_Science'}, {'term': 'Science (journal)', 'url':
  'http://en.wikipedia.org/wiki/Science_(journal)'}, {'term': 'Experiment', 'url': 'http
  ://en.wikipedia.org/wiki/Experiment'}, {'term': 'Superheating', 'url': 'http://en
  .wikipedia.org/wiki/Superheating'}, {'term': 'Nuclear fission', 'url': 'http://en
  .wikipedia.org/wiki/Nuclear_fission'}, {'term': 'Nuclear fission product', 'url': 'http
  ://en.wikipedia.org/wiki/Nuclear_fission_product'}, {'term': 'Neutron temperature',
  'url': 'http://en.wikipedia.org/wiki/Neutron_temperature'}, {'term': 'Neutron', 'url':
  'http://en.wikipedia.org/wiki/Neutron'}, {'term': 'Work (physics)', 'url': 'http://en
  .wikipedia.org/wiki/Work_(physics)'}, {'term': 'Scientific theory', 'url': 'http://en
  .wikipedia.org/wiki/Scientific_theory'}, {'term': 'Scientific method', 'url': 'http://en
  .wikipedia.org/wiki/Scientific_method'}, {'term': 'Benzene', 'url': 'http://en.wikipedia
  .org/wiki/Benzene'}, {'term': 'Alpha particle', 'url': 'http://en.wikipedia.org/wiki
  /Alpha_particle'}, {'term': 'Molecule', 'url': 'http://en.wikipedia.org/wiki/Molecule'},
  {'term': 'Nuclear reaction', 'url': 'http://en.wikipedia.org/wiki/Nuclear_reaction'},
  {'term': 'Boron', 'url': 'http://en.wikipedia.org/wiki/Boron'}, {'term': 'Pressurized
  water reactor', 'url': 'http://en.wikipedia.org/wiki/Pressurized_water_reactor'},
  {'term': 'Nuclear reactor', 'url': 'http://en.wikipedia.org/wiki/Nuclear_reactor'},
  {'term': 'Nucleate boiling', 'url': 'http://en.wikipedia.org/wiki/Nucleate_boiling'},
  {'term': 'Boiling point', 'url': 'http://en.wikipedia.org/wiki/Boiling_point'}, {'term':
  'Energy', 'url': 'http://en.wikipedia.org/wiki/Energy'}, {'term': 'Electromagnetic
```

Query the search engine without logging in:

For querying the search engine without logging in, I've included a search box in the home page of the website. The code for this search box is included in the home.blade.php . Elasticsearch Client is installed for Laravel using the command: composer require elasticsearch/elasticsearch. Once the user types in the query/ keyword in the search box, they are redirected to the SERP page where the results are displayed as hyperlinks. The title of the ETD is displayed as a hyperlink along with metadata like author, year published, university and abstract. Only few lines of the abstract is visible in the SERP page whereas once the clicks on the hyperlink, which is the title, they are redirected to a summary page. The code to implement this functionality is written in Resources->views->search.blade.php and the functionality for searching the keyword or query in the indexed data is written in the **es** function in the App->Http->Controllers->ElasticsearchController.php.

Search engine accepts a text query in the search box and returns the results in the search engine results page:

Once the user enters the search word(s) in the search box, it is taken as a query and is matched with fields like title, author, advisor, program, degree, university, abstract, wiki_terms and year in the webproject index. All the ETDs whose metadata has the search word(s) are retrieved and printed to the search engine results page. The title of the ETD is displayed as a hyperlink along with metadata like author, year published, university and abstract. Only few lines of the abstract is visible in the SERP page whereas once the clicks on the hyperlink, which is the title, they are redirected to a summary page. The code to implement this functionality for searching the keyword or query in the indexed data is written in the **es** function in the App->Http->Controllers->ElasticsearchController.php, Resources->views->search.blade.php and Resources->views->serp.blade.php **.** The difference between search.blade.php and serp.blade.php is that search.blade.php is for displaying the SERP for the unauthenticated users and serp.blade.php is for authenticated users.

Each item in SERP should link to a page for a document:

Every result in the SERP is an ETD's title which is displayed as a hyperlink to a summary page. The author, university, year published, and few lines of abstract are also displayed along with their respective titles. Once the user clicks on the title, they will be redirected to the summary page which is viewed using the Resources->views->summary.blade.php and Resources->views->serp_summary.blade.php **.** In the summary page, all the metadata of the ETDs like title, author, advisor , year published, abstract, degree, university and a link to open the pdf of that ETD are displayed. This functionality of opening the summary page of that ETD by clicking on its title is done by passing the etd_file_id of that title in the URL and then retrieving that id's data and printing it in the summary page. This functionality is written in the summary and serp_summary blades and the dissertation_details() function in the App->Http->Controllers->ElasticsearchController.php**.** Once you click on the link of the pdf in the summary page, the respective pdf is opened. This functionality is implemented by passing the pdf name in the URL and retrieving the pdf with that name from the local storage and the code to implement this is written in the **pdf** function in App->Http->Controllers->ElasticsearchController.php**.**

Search engine should display the number of returned items on top of search results:

The number of results returned after searching for search word(s) is displayed below the search box. The count of the number of results is retrieved by using the count variable in the code, which basically calculates the total number of hits after the search word is checked against all the necessary fields in the webproject index. The total number of hits is calculated in the Resources->views->search.blade.php and Resources->views->serp.blade.php **.** There's a condition given that if the count is equals to 0, then No search found is displayed on the screen and else if count is greater than 0 then all the fields matching with the searched word(s) are retrieved and displayed on the SERP along with the count value which is the number of results.

Search engine should display the actual keywords after filtering and a search box on the top:

The code to include a search box on the top of the SERP is written in the Resources->views->search.blade.php and Resources->views->serp.blade.php**.** The search box is included using an HTML input type and in order to display the searched keywords in the search box, I've written an echo statement and printed the searched keyword in the input type's value. Also, if the user searches for a word along with special characters or extra spaces, they are being removed and only the keyword is displayed in the search box, and this is done using the trim() and preg_replace() functions. This is also implemented in the Resources->views->search.blade.php and Resources->views->serp.blade.php**.**

Search results should be paginated:

The number of results shown per page is limited to 12 results per page. If there are more than 12 results, the next results are printed in another page instead of printing all the results in the same page. The page numbers are displayed at the top of the page and when a user clicks on a particular page number, the results in that page will be displayed. This pagination is implemented using JavaScript. The code for pagination is included in the Resources->views->search.blade.php and Resources->views->serp.blade.php**.**

The query terms should be highlighted:

The query terms entered are highlighted in the search results using the highlightWords() function. I've written a function called highlightWords() and I'm passing the retrieved search results and query terms as parameters to the function and inside the function I'm updating the matched term with yellow background color and returning this highlighted word to the SERP. The highlightWords() function and other code for highlighting the query terms is written in the Resources->views->search.blade.php and Resources->views->serp.blade.php**.**

Use Wikifier to obtain Wikipedia terms and store the terms and Wikipedia URLs in the JSON object:

The Wikifier terms and their URLs are generated using the wikifier_updated.py code. The metadata_abstract.csv file, uploaded in the GitHub, which has the metadata and the abstracts, was merged with the wikifier terms and URLs. The final merged file is then indexed in the Elasticsearch by uploading the .csv file in the elasticsearch. I've named the index as webproject. This index contains the 500 ETD's metadata, abstract and the wikifier terms and URLs in the form of JSON object.

When a user clicks a search result, he should see a summary page showing all metadata fields (including abstract):

Once the user clicks on the title, they will be redirected to the summary page which is viewed using the **Resources->views->summary.blade.php** and Resources->views->serp_summary.blade.php **.** In the summary page, all the metadata of the ETDs like title, author, advisor , year published, abstract, degree, university and a link to open the pdf of that ETD are displayed. This functionality of opening the summary page of that ETD by clicking on its title is done by passing the etd_file_id of that title in the URL and then retrieving that id's respective metadata and printing it in the summary page. This functionality is written in the summary.blade.php, serp_summary.blade.php and the **dissertation_details** function in the App->Http->Controllers->ElasticsearchController.php Once you click on the link of the pdf in the summary page, the respective pdf is opened. This functionality is implemented by passing the pdf name in the URL and retrieving the pdf with that name from the local storage and the code to implement this is written in the **pdf** function in App->Http->Controllers->ElasticsearchController.php

A regular user should be able to insert a new document, including key metadata and upload the PDF:

An authenticated user has an advantage over the unauthenticated user and that is uploading new PDFs and data in the index and for this they'll have click on the Upload New ETD option in the navigation bar after they've logged in. I've used an HTML form to collect the data like title, author, year published, advisor, program, degree university, abstract and a pdf file for the ETD. Once they have been submitted, I've used the predefined variable in php called $_FILES in order to retrieve the file's name and moved the file to the local storage where the other PDFs are existing. I'm also making sure that the file's extension is .pdf by checking its extension. After retrieving the pdf file's name, I'm inserting the entered metadata details and pdf name into the webproject index. The code for implementing this is in Resources->views->upload_etd.blade.php**.**

Search the title of the new paper and it should show up in the search results:

Once the data entered by the user has been indexed, when I search for the title of the inserted ETD, it will show up in the SERP. On clicking the title, it'll be redirected to the summary page like any other search result. Since the data is indexed, it is retrieved like any other indexed data which matches with the searched query or word. In the summary page, all the metadata is printed along with a hyperlink to open the pdf that has been uploaded. The code to implement this functionality for searching the title is written in the **es** function in the App->Http->Controllers->ElasticsearchController, Resources->views->search.blade.php and Resources->views->serp.blade.php**.** If the user is authenticated, the serp.blade.php is executed else if the user is unauthenticated, the search.blade.php is executed.

Add reCAPTCHA to the login page:
A google version 2 reCAPTCHA widget is added to the Login page of the website. The user must check the reCAPTCHA checkbox after entering the login credentials in order to login to their accounts. The reCAPTCHA is added to the page using g-recaptcha tag, site key and secret key. The site key and secret key are the Google reCAPTCHA API keys. The g-recaptcha-response field is set to required in the validation rules of the login authentication function, login_auth() in App->Http->Controllers->MainController.php to make sure that the user will not be able to login without checking the reCAPTCHA checkbox. The reCAPTCHA verification code can be found in the passes() function in the App->Rules->ReCaptcha.php file.

The search engine can prevent XSS vulnerability by sanitizing the JavaScript in the queries and showing the actual term on top of the search results:

In order to prevent the XSS vulnerability, the trim() function is used along with the preg_replace() and strip_tags() functions. The strip_tags() function will remove all php and html tags removed from it. All the functions used together remove the special characters and JavaScript tags in the searched query. The code for this can be found in es() function in App->Http->Controllers->ElasticsearchController.php file.

The search engine can highlight the matched terms in the search result page:

The query terms entered are highlighted in the search results using the highlightWords() function. I've written a function called highlightWords() and I'm passing the retrieved search results and query terms as parameters to the function and inside the function I'm updating the matched term with yellow background color and returning this highlighted word to the SERP. The highlightWords() function and other code for highlighting the query terms is written in the Resources->views->search.blade.php and Resources->views->serp.blade.php**.**

In the summary page, highlight the wiki terms and when the user moves the mouse over a highlighted term, a popup window is shown to display the term and a clickable link to the Wikipedia page:

First, the abstract of the ETD is retrieved using the etd_file_id and is stored in the $abstract variable. $abstract has the abstract stored in the form of json encoded string, hence its decoded and is stored in the $decoded_array. After decoding, the wiki terms are stored in the $terms_array[] and their respective Wikipedia page URLs are stored in the $url_array[]. The $abstract is updated by calling the highlightWords() function whenever the wiki terms are encountered, which makes sure to highlight the wiki terms in the abstract. The highlightWords() function takes the abstract, wiki term and its respective URL as parameters and using a tooltip, the Wikipedia URL is displayed in the popup window when the mouse is hovered over the highlighted wiki term in the abstract. The code for this can be found in Resources->views->summary.blade.php file.

A RESTful API is implemented so that users can query the search engine programmatically and obtain a customizable number of results. The system will generate a random key to the user and the user must use this key to make queries to the search engine:

Once the user requests for a key using their email and password using the command in terminal: curl 'http://localhost:8000/api/generatekey/?email&password', a random key is generated for the user and is stored in the remember_token field of the users table in the database. This random key is printed in the terminal to the user. The code for this is written in App->Http->Controllers->MainController.php. The user will use this key in order to query the search engine. The user must enter the key, range and query terms in the following command: 'http://localhost:8000/api/search?query&range&key'. Range indicates the maximum number of results to be displayed as output in the terminal. The output is displayed in the form of json object. The code for this is written in api_search() function in App->Http->Controllers->MainController.php.

Design and implement a favicon for the website:
A favicon logo has been implemented using the favicon.io website. A .jpeg image has been converted to .ico format using the favicon.io website and the converted image has been added in the project's public folder. The image can be found in public->favicon.ico. This image is then added as an icon using the link tag with rel attribute equals to icon in the CSS of the website.

Design and implement a consistent footer that appears at the bottom of all site pages:

A consistent footer has been included in all the pages of the website. The CSS styling and copyright details to be included in the footer are present in the Resources->Views->footer.blade.php file which has been included in all the site pages.

A browser compatible design: when shrinking the browser window, the layouts of panels can automatically change from a grid to vertical layout:

In order to make a browser compatible design, the margin for html, body, forms and containers used for wrapping the data, have been given as auto in the CSS of the website and also the position as relative. The code for this can be found in the style tags of the html code in Resources->views->css.blade.php and index.blade.php.

7. What would you change if you re-do this project?

If I were to re-do the project, I would implement the project using ReactJS and NodeJS. I feel using NodeJS would have been more optimal rather than using PHP because NodeJS offers better speed compared to PHP. Also, I would have implemented pagination using a for loop and display the page number in the URL.

8. What do you want to say to the instructor?

I'm glad that I took this course as I've really learnt a lot while doing the project. The course is well designed, and the presentations are precise and clear. The presentations really helped me in finishing the project. One feedback I'd like to give is about the milestone specifications. I felt the specifications could be much clear as they created confusion at times.