

Linked Data- DER5101

Assignment 4 – Linked Data Application

Oisin McNally (14376701),

Conor Hayes (10354355) ,

Surya Balakrishnan (18231072),

Sai Krishna (18230229)

Roles for each member

Oisin McNally: *SparQL Query Designer*

Conor Hayes: *Application Designer*

Surya Balakrishnan: *Integration Engineer*

Sai Krishna: *Interface Designer*

We feel that each member of the groups contribution was equal and that the work was divided equally between each member. While we all had specific role, each team member assisted each other in all facets of the application.

1.0 Project Overview

The goal of this project is to design a search engine to provide information on drugs using a combination of two databases. Bio2RDF is a biological life-science database that provides a database of drug names while an abstract (short description) about the drug can be extracted from DBPedia.

The search engine should perform similar to Google where each time a character is typed into the search box the autocomplete terms should be displayed. The only autocomplete terms available will be drugs from the Bio2RDF database.

When a term is selected, the search button can be pressed to send a query to DBPedia and retrieve a description about the drug and some additional information if it is available.

Bio2RDF

Each time the text in the search box is updated, the text should be parsed and injected into the highlighted section of the query. The query should then be executed and a list of the matching drug names displayed in the autocomplete list.

```
PREFIX db: <http://bio2rdf.org/drugbank_vocabulary:>
PREFIX dcterms: <http://purl.org/dc/terms/>
SELECT ?drug_name # get drug name
WHERE {
  ?drug a db:Drug . # the subject is of class drug
  ?drug dcterms:title ?drug_name # all drugs with a title
  FILTER(regex(str(?drug_name), "^pen", "i")) # filter for names that begin with "pen"
  FILTER(regex(str(?drug_name), "[A-Za-z]*$", "i")) # filter for names with no spaces
}
LIMIT 10 # limit to 10 results
```

DBPedia

The term from the search box should be parsed into the highlighted section and the SPARQL query executed when the search button is pressed. This will query DBPedia for IRI's containing the search term and return the abstract and additional information about the drug.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
prefix resource: <http://dbpedia.org/resource/>
SELECT *
WHERE {
  ?resource dbo:abstract ?abstract # get all subjects with an abstract
  OPTIONAL { ?resource dbp:legalStatus ?status } # get legalStatus if available
  OPTIONAL { ?resource dbo:thumbnail ?thumbnail } # get thumbnail if available
  FILTER(regex(str(?resource), "Methylphenidate$", "i")) # get resource that ends with drug name
  FILTER langMatches(lang(?abstract), 'en') # filter for english descriptions only
}
LIMIT 20 # limit to 20 results
```

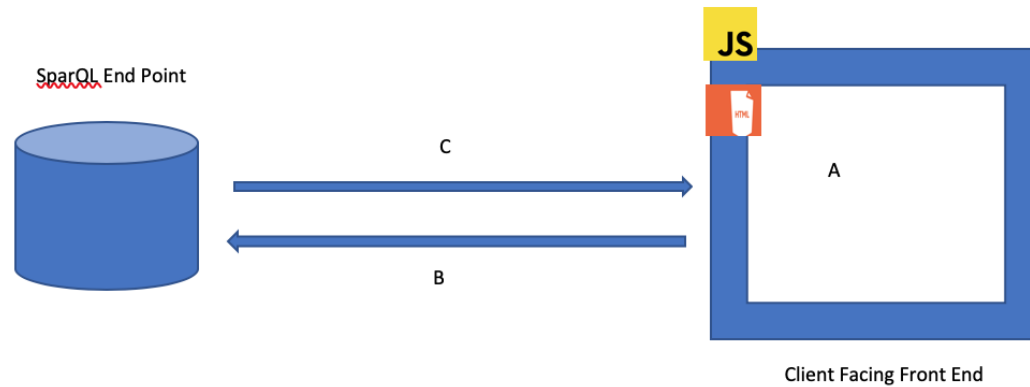
2.0 System Architecture

In order to get a running application working in the required time frame it was required to keep the application architecture as simple as possible. Therefore because of this we have proposed two architectures. One describes the actual architecture of the proof of concept while the other will describe an ideal system we would have designed if more time was available

For the proof of concept, we kept the architecture very simple. As you can see in figure 1, that we have a simple front end facing the user. This frontend is powered by HTML and JavaScript. Here the HTML script calls a JavaScript function, this in turn takes the input from the search bar in the main page and creates a SparQL query. This query is then sent to the required endpoint and the data is retrieved. Once the data is retrieved we then format the return JSON object and place it in a table for the user to view.

For the future work architecture, we expanded heavily on the proof of concept. Our main end goal would have been to essentially build our own sparql endpoint using data gathered from several other endpoints, extracting the necessary data, cleaning the data and then opening that database open to the public. Once this was done we would have ideally then built a drug search engine application on top of this database for all users on the web to use. Due to time constraints this did not become a reality, but we have outlined both architecture for this concept and the proof of concept below.

Proof of Concept Architecture

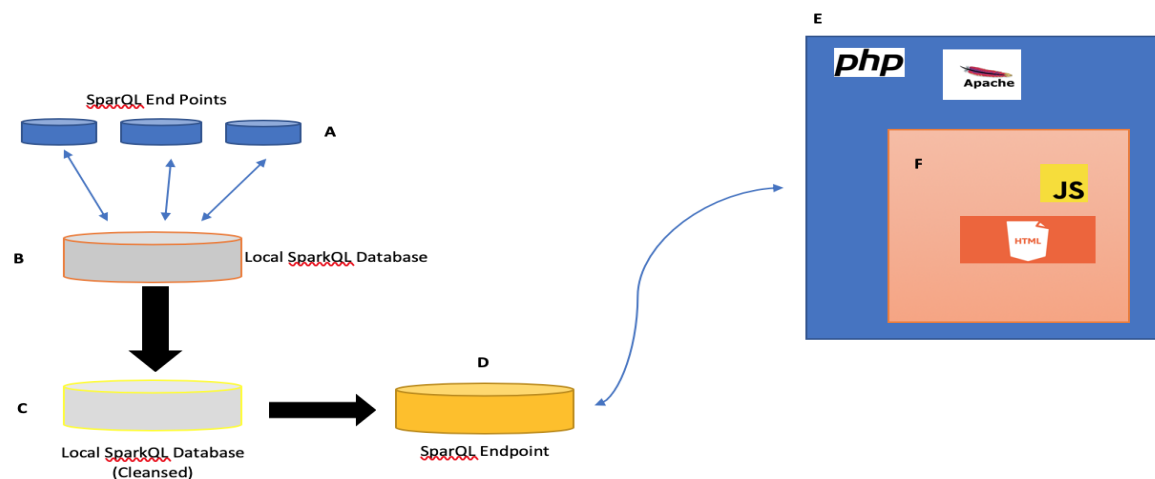


A – User inputs a search word and hits the search button. The Client front end is powered by HTML and the query generated takes the search query A and uses JavaScript to create a query.

B – Once the query is created, a JavaScript script executes the query. The query then hits the SparQL Endpoint (DBpedia)

C - Once the query has been executed, the results are gathered form the SparQL Endpoint and returned to the client front end. Once returned the results are parsed using JSON and we then can display the results to the user via HTML again

Future Work Architecture



A – Here, we would query more than one sparql endpoint and gather as much data as possible from the endpoints available on the topic of drugs

B – We would then download all the available data and put this into one large sparql database

C – Once we have gathered the data, we would then cleanse the data and put it in the required structure. This would ensure all the required data such as drug information, chemical compounds, picture of the chemical compound and what the drug is commonly used to treat.

D – Once our database is cleaned and has applicable data, we would then open this database up as an endpoint of tidy data for any user to query on the web. This would provide a compact and distinct source of information on drugs

E – Lastly, we would build a robust web application that sits on an apache server using tools like php to gather session as to add more functionality compared to our proof of concept. The finished product would look something similar to our proof of concept to the user but we would be using a strong backend with our own clean data source.

```
<script>
function myFunction() {

    document.getElementById("drugresult").style.display = "block";
    document.getElementById("drugresult").style = "margin: 0px;
width: 915px; height: 248px;"
    var drug = document.getElementById("textInput").value;
    //var drug = 'Penicillin' or 'Vicodin' or 'Morphine' or
    //var XMLHttpRequest = require("xmlhttprequest").XMLHttpRequest;
    var xhr = new XMLHttpRequest();
    const Http = new XMLHttpRequest();

const url='https://dbpedia.org/sparql?default-graph-uri=http%3A%
2F%2Fdbpedia.org&q=PREFIX%20dbo%3A%20%3Chttp%3A%2F%
2Fdbpedia.org%2Fontology%2F%3E%20%D0%A0%D0%APREFIX%20dbp%3A%20%
3Chttp%3A%2F%2Fdbpedia.org%2Fproperty%2F%3E%20%D0%A0%D0%Aprerfix%
20resource%3A%20%3Chttp%3A%2F%2Fdbpedia.org%2Fresource%2F%3E%20%
D0%A0%D0%ASELECT%20%*%20%20%D0%A0%D0%QAWHERE%20%7B%20%D0%A0%D0%
O%A3%2Fresource%20dbo%3Aabstract%20%3Fabstract%20%23%20get%20all%
20subject%20with%20and%20abstract%20%D0%A0%D0%AOPTIONAL%20%7B%20%
20%3Fresource%20dbp%3AlgalStatus%20%3Fstatus%20%7D%20%23%20get%
20legalStatus%20if%20available%20%D0%A0%D0%AOPTIONAL%20%7B%20%
3Fresource%20dbo%3Athumbnail%20%3Fthumbnail%20%7D%20%23%20get%
20thumbnail%20if%20available%20%D0%A0%D0%OAfilter(regex(str(%
3Fresource)%20c%22+ drug + '%24%22%2Cc%22i%22))%20%23%20get%
20resource%20that%20ends%20with%20drug%20name%20%D0%A0%D0%
OAFILTER%20langMatches(lang(%3Fabstract)%20c%27en%27)%20%23%
20filter%20for%20english%20descriptions%20only%20%D0%A0%D0%O%D0%A0%7D%
20%D0%A0%D0%OALIMIT%20%20%23%20LIMIT%20to%20%20%20%
20results&format=application%2Fsparql-results%
2Bjson&CXMLe_dir_for_subj=&i=1&CXMLe_dir_for_hrefs=&timeout=
30000&debug=false&run=%2ORun%20Query%20
&fbclid=IwAR2RmSuCb1lKGmjSVlq_x_WfcVjpiYHXG_YiiYzBEMtsNjnru9S9poIt
iFpU';

Http.open("GET", url);
Http.send();


Http.onreadystatechange=(e)=>{
//console.log(Http.responseText)
var drug_info = Http.responseText;
//var obj = JSON.parse(drug_info)
//console.log(typeof drug_info)
var resultJson = JSON.parse(drug_info);
var drug_text = resultJson.results.bindings[0].abstract.value;
document.getElementById("drugresult").value = drug_text;
//document.getElementById("demo").innerHTML = drug_text;
```

4.0 Front End Tasks

The programming language used for the front end design tasks are html5 and css3. The major reason for going ahead with these were they're the latest one in the market. It plays a crucial role in building websites with adaptive response which changes in size accordingly when the size of the browser is changed. It also provides the capability to view this content effortlessly in mobile browser also without the issue of bloated texts and images. The issue faced in the front end task in providing the result output in the desired format. The problem was in building an adaptive response for the result and hiding the result text field until the search query has been given by the user. These were sorted by incorporating the appropriate commands towards the object id to hid the result and display them in adaptive response once the search has been made. There are 4 html pages in total which can be accessed from the navigation bar. The index page is one where the queries are given by the user. The about developer page provides the details about the developer and about project provides detail about the project. Finally the screencast demo of the project is provided in the 4th page with a feature of even to view the video in picture in picture mode. **Sample html code is as follows,**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any
other head content must come *after* these tags -->
    <title>Drughoo! your drug search destination</title>
    <link rel="icon" href="nav.png">

    <!-- Bootstrap -->
    <link href="css/styles.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5
elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page
via file:// -->
    <!--[if lt IE 9]>
      <script
src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js">
      </script>
      <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js">
      </script>
    <![endif]>
  </head>

  <body data-spy="scroll" data-target="#navbar" bgcolor =
"#FFA07A">
    <nav id="navbar" class="navbar navbar-default navbar-inverse
navbar-fixed-top">
      <div class="container-fluid">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#bs-example-navbar-
collapse-1" aria-expanded="false">
            <span class="sr-only">Toggle Navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="index.html">
            
          </a>
        </div>

        <!-- .navbar-header -->

        <!-- Collect the nav links, forms, and other content for
toggling -->
        <div class="collapse navbar-collapse" id="bs-example-
navbar-collapse-1">
```

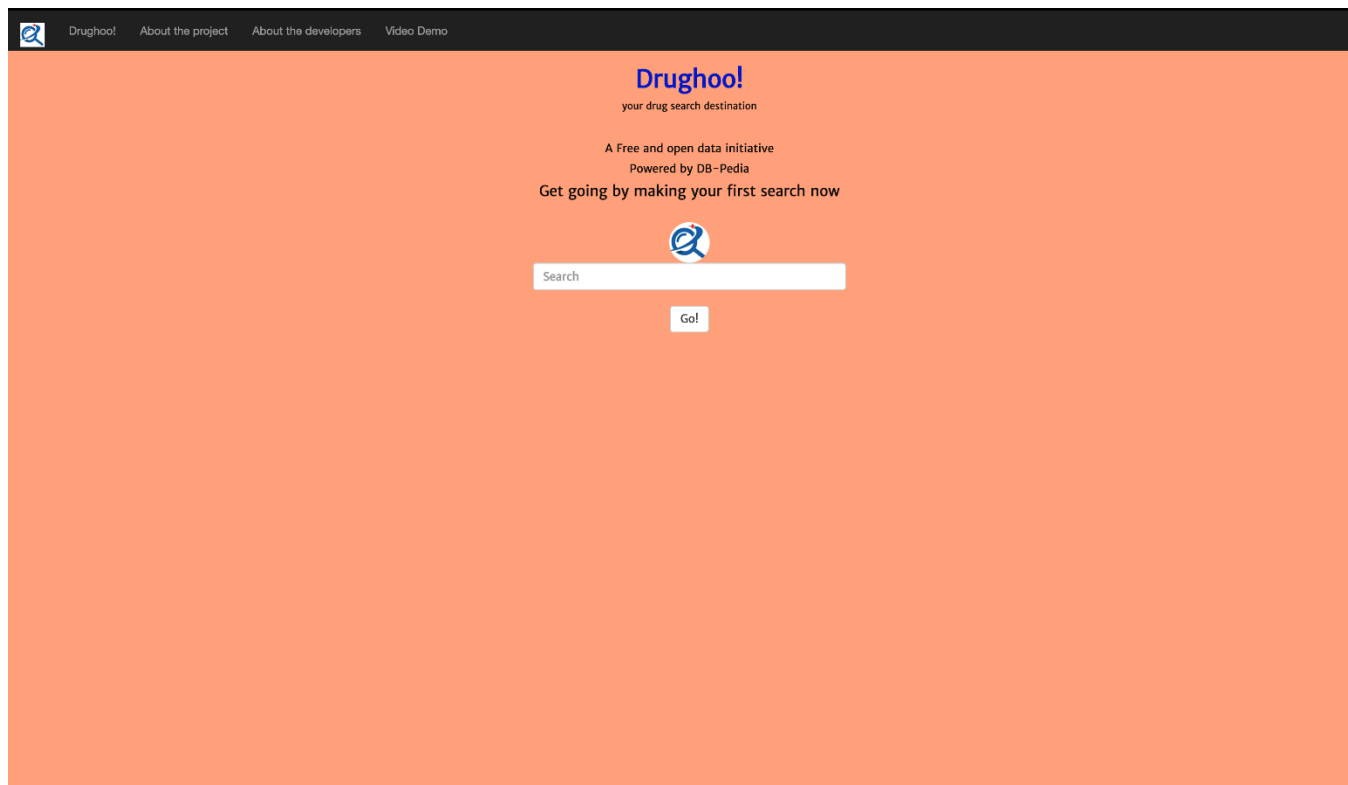
5.0 Installation instructions

- In order to test run the application, just extract the zip and run the index.html. Provide the required drug query in the search bar and click on go. **Wait for a minute for the result to be extracted and displayed.**
- To write the code any text editor is sufficient. But in order to compile it, bootstrap has to be installed.


6.0 Working Screencast and Screenshot

In order to see the screencast, kindly look into the VID.mp4 or the Video Demo option from the navigation bar or directly selecting the video.html.

1.Main Page



2.Sample Search


 [Drughoo!](#) [About the project](#) [About the developers](#) [Video Demo](#)

Drughoo!

your drug search destination

A Free and open data initiative
Powered by DB-Pedia


Get going by making your first search now




Go!

Penicillin (PCN or pen) is a group of antibiotics which include penicillin G (intravenous use), penicillin V (oral use), procaine penicillin, and benzathine penicillin (intramuscular use). Penicillin antibiotics were among the first medications to be effective against many bacterial infections caused by staphylococci and streptococci. Penicillins are still widely used today, though many types of bacteria have developed resistance following extensive use. About 10% of people report that they are allergic to penicillin; however, up to 90% of this group may not actually be allergic. Serious allergies only occur in about 0.03%. All penicillins are β -lactam antibiotics. Penicillin was discovered in 1928 by Scottish scientist Alexander Fleming. People began using it to treat infections in 1942. There are several enhanced penicillin families which are effective against additional bacteria; these include the antistaphylococcal penicillins, aminopenicillins and the antipseudomonal penicillins. They are derived from *Penicillium* fungi.

3.About Project

 [Drughoo!](#) [About the project](#) [About the developers](#) [Video Demo](#)

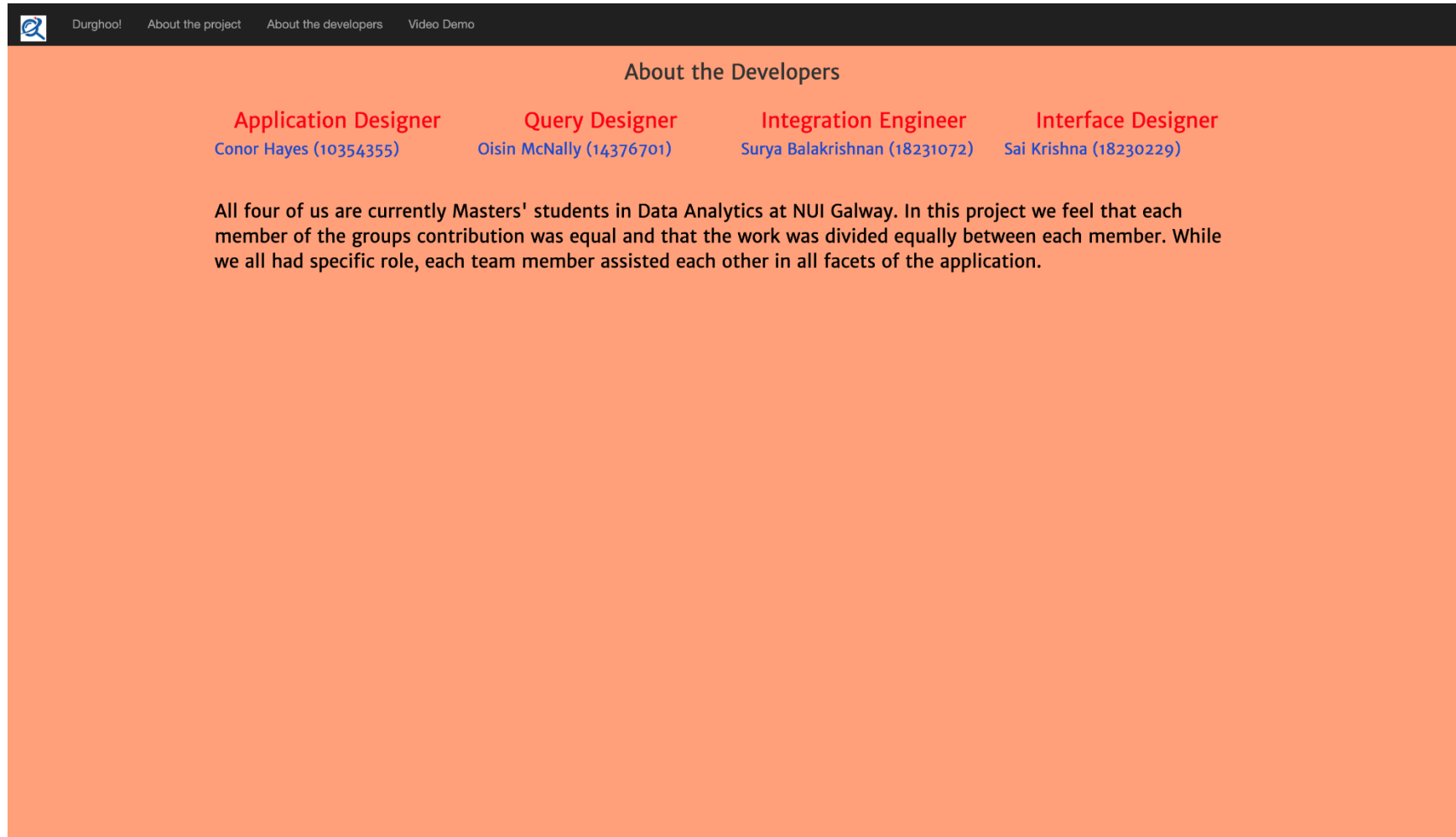
About the Drughoo! initiative



The goal of this project is to leverage the power of linked and open data to create a solution for information retrieval by making it accessible through a user friendly user interface. What better way to accomplish the task rather than designing a search engine to provide information using a combination of two databases. The domain of choice is the medical drug industry as details on a drug is not easily available and accessible. Bio2RDF is a biological life-science database that provides a database of drug names while an abstract (short description) about the drug can be extracted from DBPedia. The search engine should perform similar to a Google search where each time a character a drug name is typed into the search box, and the search button be pressed to send a query the application looks for the drug information into DBPedia and retrieves a description about the drug and some additional information if it is available.

This project is part of the assignment for the linked data module DER 5101, Developing a complete linked data application.

4. About Developers



The screenshot shows a web application interface. At the top, there is a dark navigation bar with a logo on the left and four links: 'Durghool', 'About the project', 'About the developers', and 'Video Demo'. The main content area has an orange background. It features a title 'About the Developers' centered at the top. Below the title, there are four columns, each representing a developer's role and name with a contact number in parentheses. The roles are 'Application Designer', 'Query Designer', 'Integration Engineer', and 'Interface Designer'. The names are 'Conor Hayes', 'Oisin McNally', 'Surya Balakrishnan', and 'Sai Krishna'. At the bottom of the orange section, there is a paragraph of text explaining that all four are Masters' students in Data Analytics at NUI Galway and that their contributions were equal.

Application Designer
Conor Hayes (10354355)

Query Designer
Oisin McNally (14376701)

Integration Engineer
Surya Balakrishnan (18231072)

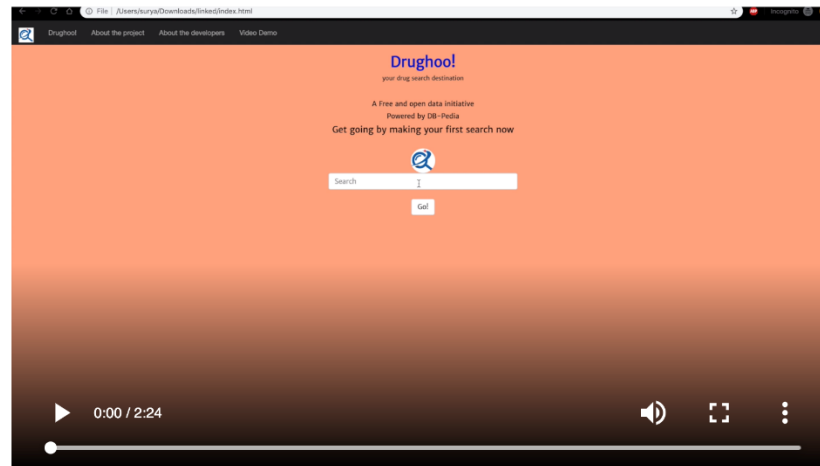
Interface Designer
Sai Krishna (18230229)

All four of us are currently Masters' students in Data Analytics at NUI Galway. In this project we feel that each member of the groups contribution was equal and that the work was divided equally between each member. While we all had specific role, each team member assisted each other in all facets of the application.

5. Video Demo



Video Demo



A hands on video demo of the project