The YouTube Data API provides access to YouTube. In this assignment you will write an MVC web application which uses the YouTube data api to provide a user interface for search and video details. Please use ASP.Net MVC in this project, but outside of that feel free to use any frameworks or libraries you wish to accomplish this task. If you choose to use code generators or a boilerplate project as a starting point, please make sure we can tell which parts of the project are yours and which are generated or from the boilerplate. Descriptions in the readme and a range of commits helps with this.

*I started by a standard Visual Studio create new mvc web project, and added my controller action links and menu items to Home\Index. In order to do IoC testing of the controller, it must be moved to its own project.*

*Addins I utilized:*

*JQuery 1.12.4,*

*Javascript Data Tables:* [*https://datatables.net/*](https://datatables.net/)

*Newtonsoft Json 9.0.1*

*Entity Framework 6.1.3*

Your application should consist of the following views/features:

**Search**

         Search by keyword

*My interpretation: Search YouTube in the same manner as we would on the YouTube web site, in the search box; in other words enter the words to search on and click the button.*

        Sort by date, rating and relevance

*Interpretation: “date” = date the video was posted. “rating” = from you tube data model. “relevance” = exactly what order the videos come up in the search results. Compare my output to what the search would show on youtube.com; that’s what order the records come up.*

*I added a link to “View”, which will just open the video in youtube.com on a new tab in the browser. The link “Fave it” will add that video to your favorites. I pass a json string as the only parameter to Home\Adder, which will persist it to the database. I realize a hacker could do malicious query string alterations by viewing, with Fiddler, what I pass.*

***Wish list for next version:*** *Refactor the <a href=”home\adder\?parm=jsonstring”> to a true Ajax post click handler. I would have to keep track of indexes on the array, or make a unique #id for each array item, so my jquery click selector would bring in the proper data, and build a $.ajax call. Once I do a real ajax call, I would just send my json string, and home\adder would be a [HttpPost], and its param would be an instance of my view model, and no need for json deserialize.*

**Video Details**

         Video player

*I use a fixed iframe to show the video. You may visit the page Home\Details or Home\Details\string where [string] is the unique you tube video id that all you tube videos use to identify. If you do not specify a string, this page will default to video “H2KkiRbDZyc” which is the 10 hour nyan cat video. Wish list for next version: research how to do a native html5 viewer.*

         Meta information like comments, likes, dislikes, channel

*The information shown here will be what I persist to the database at the time the video is added to favorites. Therefore, if more likes / dislikes / comments are added to the outside world of you tube after you favorite the video, the latest information will not show on this page.*

*A video can have thousands of user comments. In the interest of demonstrating the functionality of the whole system, and not digging too deep into individual details, I plan to only display the first comment. Making an open-ended (1..n) list of all comments is doable as a next sprint item: a new one to many relationship in the database and the view model.*

*The look and feel of the meta data is identical to the Favorites page: I use the same Html.Partial, and javascript datatables with a size of “1” to display the data. A next sprint item could refactor this display of one record to have the exact same styling as the data tables, without bringing in the whole functionality [sorting, searching, paging] which are all redundant for a one record table.*

         Ability to add/remove from favorites (see below)

*In the search results window, I added a link “Fave it” which will add that video to your favorites.*

*In the “Favorites” page, I added a link “Remove” which will remove that video from your favorites.*

**Favorite Videos**

         The list of videos the user has favorited

*The “View” link will bring up the selected record on the “Details” web page. The “Remove” link will delete that favorite from the list of saved favorites.*

* Implement your own data storage system

*Use enclosed script Create\_youtube\_db.sql to create a sql 2014 db. I could have set the compatibility level to 2008 or lower to accommodate the target system, but believe sql 2014 is conservatively low enough.*

*Because all you tube videos use a string of random characters like “H2KkiRbDZyc” to uniquely identify videos, I use that as the primary key. Because the whole system is very simple [one table], just standard default non clustered primary key index is the only “optimization” to perform.*

*The connection string in the system says server = . which means same server as the web server. This is not best practice: have the database server on the application server. In this case, as a demo app, it is acceptable.*

Try not to get bogged down in the look and feel of the application - we’re more interested in how you organize the project and which tools (patterns, libraries, frameworks, dependency management, etc.) you choose to use.

When you’re finished, please put your project in a repository on either Github or Bitbucket and send us a link. We’ll then do a pair programming session where we’ll have some questions for you about your code and possibly make some additions to it.

*Known issues / next sprint items to fix:*

*I could not get bundling to work. I had to drag jquery and datatables onto each individual .cshtml file as an include to make it work. I know properly adding the css and js files into the bundle config.cs will make it transparent, and efficient.*

*Some add to favorites abort. I believe it is because of some columns are not populated, and are undefined, even though I check all columns like so: if (Title === undefined) and populate them with default values.*

*Doing search twice on the search page causes a warning popup, and the subsequent searches do nothing. Workaround: leave the search page, come back, and all is well.*

*More youtube api work to get the ancillary data not provided in first sprint: comments, likes, dislikes, etc. First sprint currently shows title, channel title, date published.*

*Change all javascript files to the minified js before deploying to production!*