NATIONAL CRIMINALS DATABASE

SOURCE CODE

1. https://github.com/aukgit/National-Criminals-Database
2. All the completed tasks: https://github.com/aukgit/National-Criminals-Database/issues?q=is%3Aissue+is%3Aclosed
3. 10 Errors : https://github.com/aukgit/National-Criminals-Database/tree/2e744d8071a1edee51a5e1204adf1a49df13531a
4. Additional Implementations : https://github.com/aukgit/National-Criminals-Database/tree/582c6b127286a516182e235be525b75b788919a3

TO RUN THE PROJECT

1. By opening the "Source\NCD.sln" project and running the NCD project as startup should work just fine.
2. Emails are also configured in the system, however if required to reconfigure then
    a. Open "Source\NCD\Web.config" and then change 'MailHostingServer',' MailingPort' ,' MailSender', 'MailSenderEncryptedPassword',' EncryptedPassPhrase',' enableSSL';
    b. To get the encrypted password, please set "PasswordEncrypt.csproj" or "PasswordEncrypt" project as startup project.
    c. Or we can also run the executable file "Source\PasswordEncrypt\bin\Debug\PasswordEncrypt.exe" and give the passphrase 'NDC-Sample-Application 1.0' and give the sender email address's password and then it will save the encrypted password in a txt file located at "Source\PasswordEncrypt\bin\Debug\encrypted.txt".
    d. Finally copy paste that text in the 'MailSenderEncryptedPassword' field to have configured mailbox.
    e. When the 'MailSenderEncryptedPassword' field is retrieved then please change the startup project as "NCD".

EXISTING ERRORS

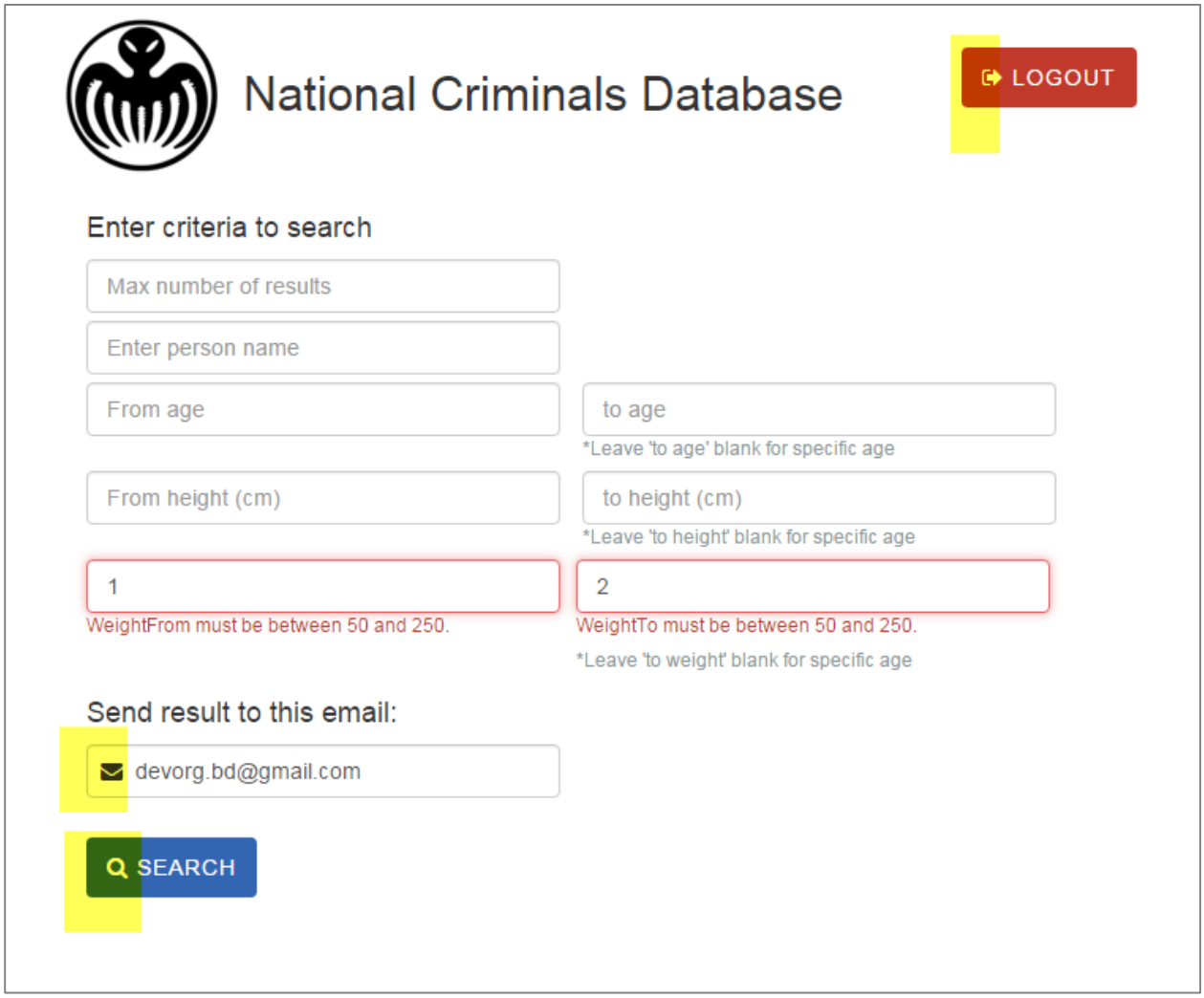
1. All are fixed from the given list https://github.com/aukgit/National-Criminals-Database/issues?q=is%3Aissue+is%3Aclosed.

ADDITIONAL ERRORS

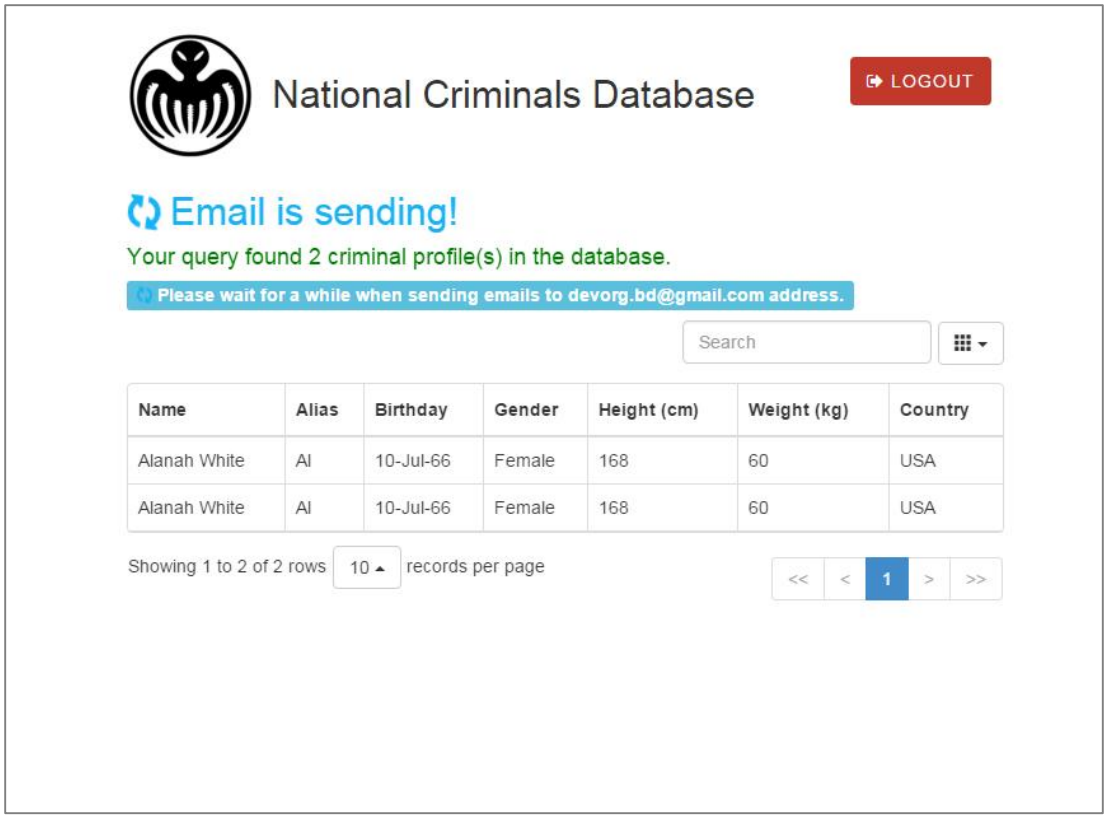
1. Last two fields 'WeightFrom' and 'WeightTo' didn't have inline validation messages. Which are also fixed.

ENHANCEMENTS

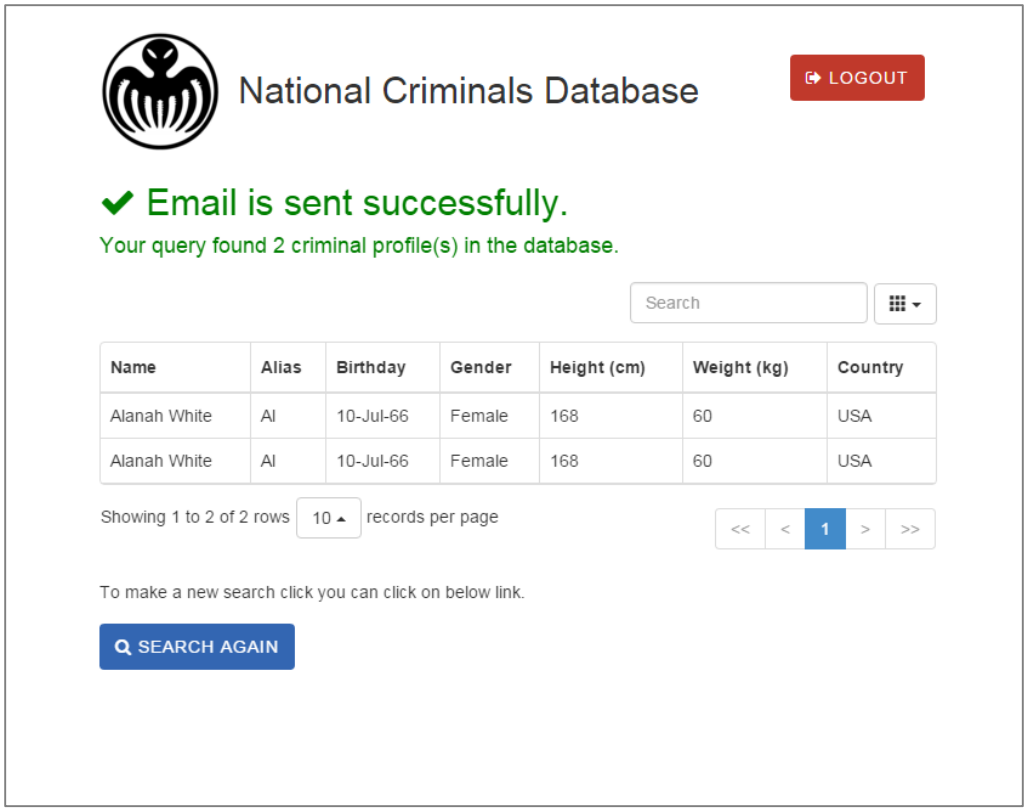
1. Enhancements are marked as 'enhancement' https://github.com/aukgit/National-Criminals-Database/issues?q=is%3Aissue+is%3Aclosed.
2. Font-Awesome.css library added , and most of buttons now contains icons from fonts.



3. Bootstrap css included.
4. Animate.css library included for css based animations.
5. Getting query results from database is fast, however convert those into pdf takes a lot of time. As a result it would be better to separate those tasks. Based on this concept, now when a query is made results will be displayed in the confirm page.



6. And then via JavaScript MVC framework written by Md. Alim Ul Karim and ajax a sending email request will be sent to the sever and then above screen is shown. In case server failed, those blue writings will became red and notify the user to try again sending the email to the user.
7. When the results are sent successfully to the email it will show a screen-shot like below:



8. Based on the given results in this page, clients can also search the results from the search bar in the screen. It will do a quick client side searching using jquery.(Bootstrap table plugin is used to implement this)
9. Client side paginations are also available here.
10. Client can click on the search again to move back to the search page again. (this is an old feature);

ADDITIONAL FRAMWORKS

1. Foolproof framework is used and modified to implement the Age compare validation. (foolproof.codeplex.com). Source code is also included inside the NDC sln . Project is 'Foolproof'. Histories can be found at github links that which parts were modified.
2. Own JavaScript Framework (similar to AngularJS) :
   a) JavaScript Framework Code: http://bit.ly/1UO1MoL
   b) JavaScript Framework how it is implemented : http://bit.ly/1KdWSHD | http://bit.ly/1KdXoqq
3. DevMvcComponent nuget package is used for Mailing setup correctly https://github.com/aukgit/DevMvcComponent

IMPORTANT CODING PARTS

```
  [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Search(SearchViewModel model) {
            if (ModelState.IsValid) {
                var searchRequest = new SearchRequest {
                    Email = model.Email,
                    MaxNumberResults = model.MaxNumberResults.HasValue ?
model.MaxNumberResults.Value : 0,
                    Name = model.Name,
                    AgeFrom = model.AgeFrom,
                    AgeTo = model.AgeTo,
                    HeightTo = model.HeightTo,
                    HeightFrom = model.HeightFrom,
                    WeightFrom = model.WeightFrom,
                    WeightTo = model.WeightTo
                };

                var criminals =
SearchService.SearchCriminal(searchRequest);
                /**
                 *  On a different, I could have used Any() on IEnumerable
instead of changing the data type to IList.
                 *  Since the query already ran in the database and there
is no new queries to add in future from C# then
                 *  IList would be a better choice.
                 *
                 *  However, if there is any future chance of query the
data from C# then IEnumerable<> would be a better choice.
                 *  Again, it depends on requirements.

                 *  for IEnumerable, Any() is faster and better
                 *  List Count is faster.
                 *  I know it is better to not change the method
signature ,
                 *  however since we already executed the query it is
better to have in memory object.
                 *  */
                if (criminals.Count > 0) {
                    var token = Guid.NewGuid();
                    string cacheToken = token.ToString();
                    var criminalRecordsViewModel = new
CriminalRecordsViewModel() {
                        Criminals = criminals,
                        Token = token,
                        Email = searchRequest.Email
                    };
```

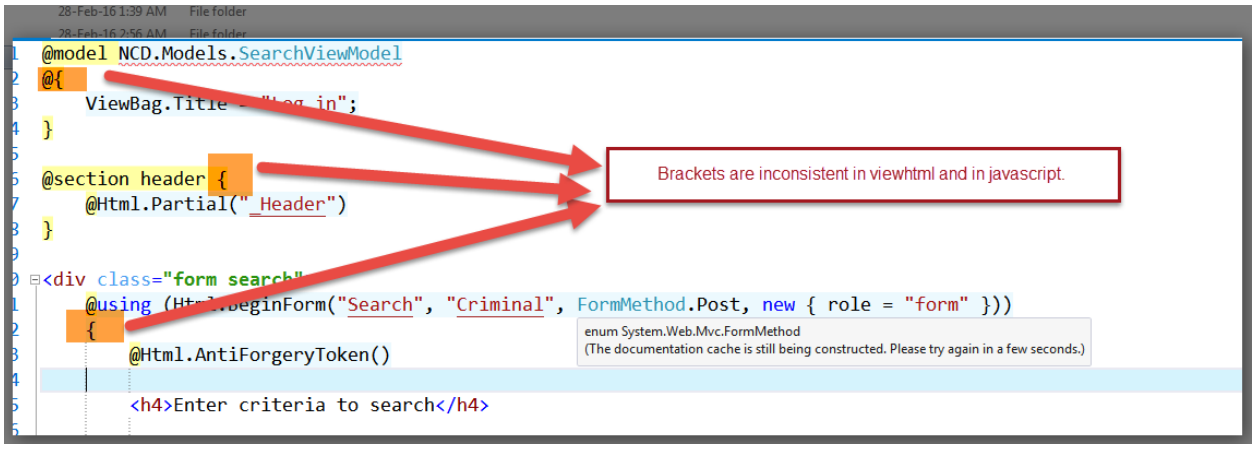
```
                    HttpContext.Cache[cacheToken] =
criminalRecordsViewModel;
                    return View("Confirmation", criminalRecordsViewModel);
                } else {
                    ModelState.AddModelError("", "Sorry ! No results found
with these parameters.");
                    return View("Index", model);
                }
            }
            ModelState.AddModelError("", "Sorry ! Invalid query
parameters.");
            return View("Index", model);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public JsonResult SendEmail(Guid? token) {
            if (token.HasValue) {
                var cacheToken = token.Value.ToString();
                var criminalRecords = HttpContext.Cache[cacheToken] as
CriminalRecordsViewModel;
                if (criminalRecords != null) {
                    //send email to that given address.
                    EmailService.Send(criminalRecords.Email,
criminalRecords.Criminals);
                    var result = new {
                        found = true,
                        message = "Email is sent successfully."
                    };
                    HttpContext.Cache.Remove(cacheToken);
                    // remove cache , don't need to keep it anymore.
                    return Json(result);
                }
            }
            var notFoundResult = new {
                found = false,
                message = "Email is not sent."
            };
            GC.Collect();
            return Json(notFoundResult);
        }

    }
```

NOTES

1. I have changed the brackets from left to right. Because there were lack of inconsistency in the project :



2. Changed method signature from IEnumerable to IList explained in the code above.

If any problem, please get in touch with me +8801833002021 or email devorg.bd@gmail.com