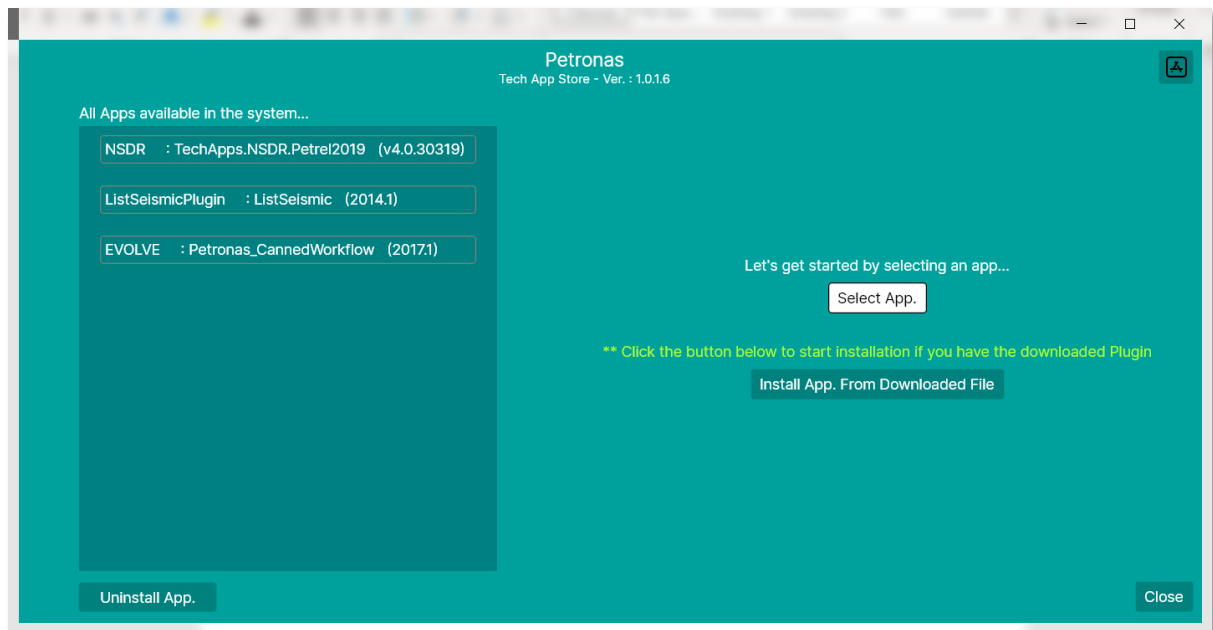


## TechAppLauncher

Application Name	TechAppLauncher
Application Framework	.Net 5.0
UI Framework	Avalonia
IDE	Visual Studio 2019
Repository	<a href="https://github.com/azzulhisham/Petronas_TechAppLauncher.git">https://github.com/azzulhisham/Petronas_TechAppLauncher.git</a>
Application Url	N/A – Standalone Desktop Application
Host	Software Center

### Overview

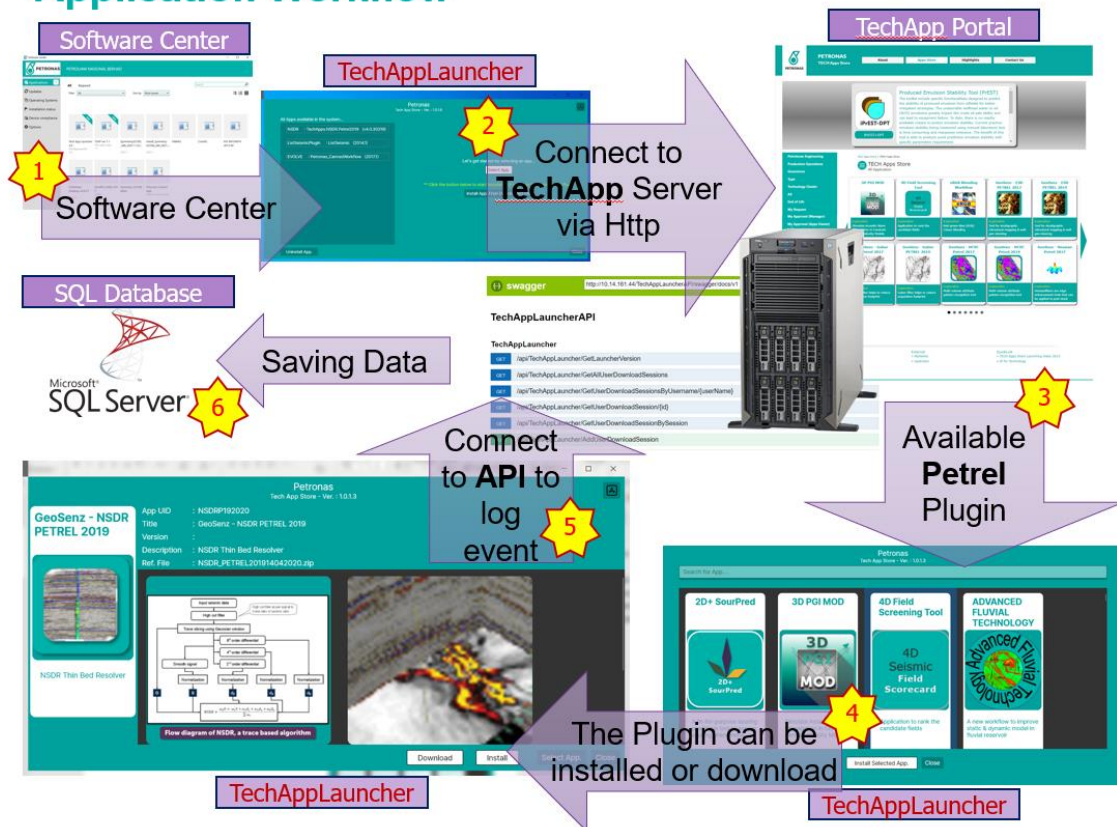


**TechAppLauncher** is a standalone desktop application that engaging the **Avalonia UI framework**. The Application is hosted at the **Software Center** in order to distribute to the client.

Click the link [Basics - Avalonia \(avaloniaui.net\)](https://avaloniaui.net) in order to learn more about Avalonia. You can follow the tutorial provides by the home page of Avalonia in order to understand how the application works in details.

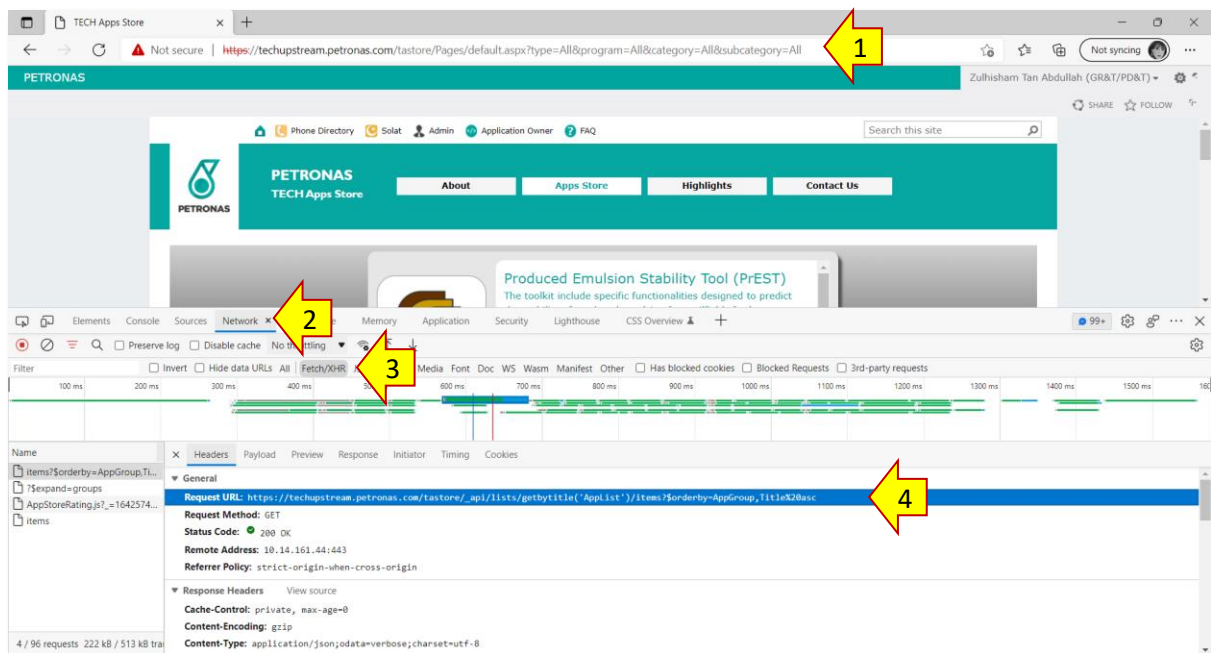
The entire flow of the application is describes in the next page.

## Application Workflow



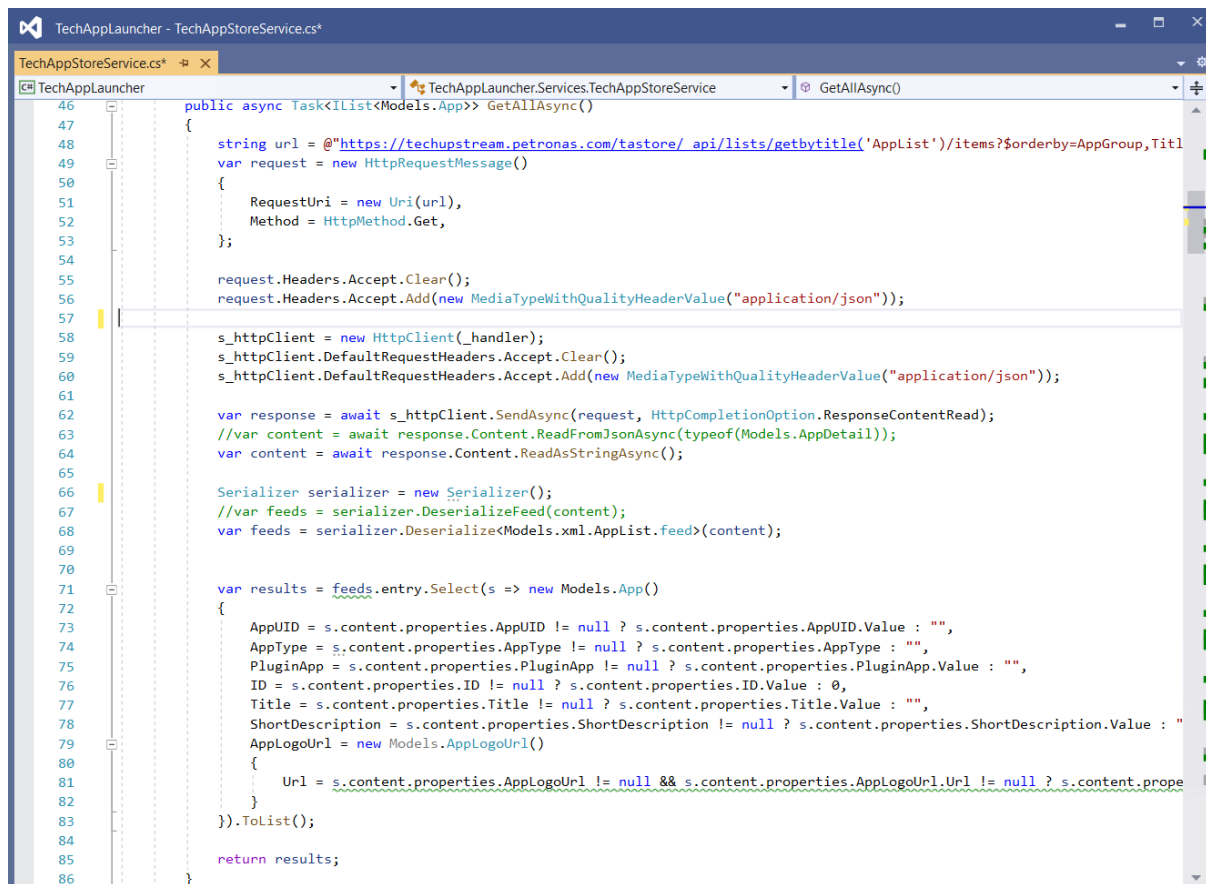
1. The application is hosted in the **Software Center**, hence user need to be installed the application from the **Software Center**.
2. The application is connected to the **TechApp Store Portal** to retrieve all the available app in the **TechApp store**.
3. The application list all the available application which is typically identical to the **TechApp Store portal**.
4. The application provides the **download** feature as well as **install** feature to the client.
5. Whenever user successfully download or install a TechApp's application, it will then connect to its **API**.
6. The application updates necessary **download session information** through its API.

## Using Browser to track important endpoint



In order to used these data in C# code, first select all the text which response from the server in the browser, and then switch Visual Studio 2019, add a new class file to the desire folder that you wish, make sure the new added class file is being selected. Then, in the menu, select 'Edit' >> 'Paste

**Special' >> 'Paste XML as Classes'.** Since the data return from the server is in **XML** format, we need to engage the **'Paste XML as Classes'** to help us convert the data format into C# entity. If you are pro in XML then you may skip this step. The generate C# class is actually serves as an temporary object to keep the XML data.



```

46 public async Task<IList<Models.App>> GetAllAsync()
47 {
48     string url = @"https://techupstream.petronas.com/tastore/_api/lists/getbytitle('AppList')/items?$orderby=AppGroup,Titl
49     var request = new HttpRequestMessage()
50     {
51         RequestUri = new Uri(url),
52         Method = HttpMethod.Get,
53     };
54
55     request.Headers.Accept.Clear();
56     request.Headers.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
57
58     s_httpClient = new HttpClient(_handler);
59     s_httpClient.DefaultRequestHeaders.Accept.Clear();
60     s_httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
61
62     var response = await s_httpClient.SendAsync(request, HttpCompletionOption.ResponseContentRead);
63     //var content = await response.Content.ReadFromJsonAsync(typeof(Models.AppDetail));
64     var content = await response.Content.ReadAsStringAsync();
65
66     Serializer serializer = new Serializer();
67     //var feeds = serializer.DeserializeFeed(content);
68     var feeds = serializer.Deserialize<Models.xml.AppList.feed>(content);
69
70
71     var results = feeds.entry.Select(s => new Models.App()
72     {
73         AppUID = s.content.properties.AppUID != null ? s.content.properties.AppUID.Value : "",
74         AppType = s.content.properties.AppType != null ? s.content.properties.AppType : "",
75         PluginApp = s.content.properties.PluginApp != null ? s.content.properties.PluginApp.Value : "",
76         ID = s.content.properties.ID != null ? s.content.properties.ID.Value : 0,
77         Title = s.content.properties.Title != null ? s.content.properties.Title.Value : "",
78         ShortDescription = s.content.properties.ShortDescription != null ? s.content.properties.ShortDescription.Value : "
79         AppLogoUrl = new Models.AppLogoUrl()
80         {
81             Url = s.content.properties.AppLogoUrl != null && s.content.properties.AppLogoUrl.Url != null ? s.content.prope
82         }
83     }).ToList();
84
85     return results;
86 }

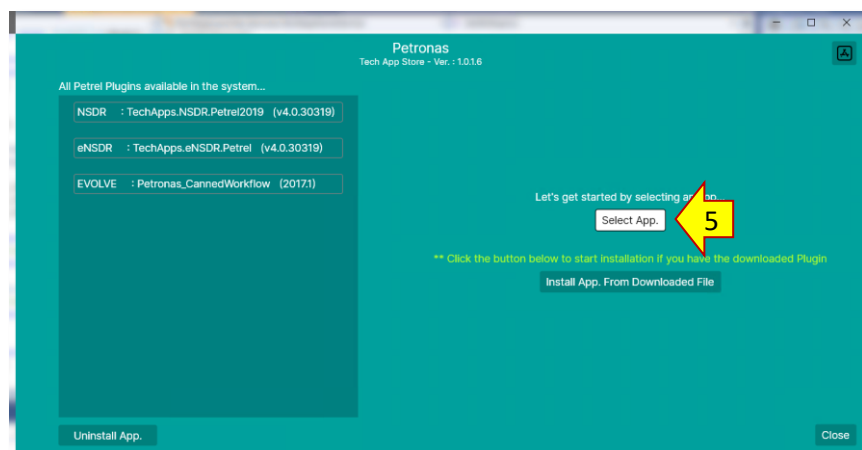
```

If we look t the code, line 48 to line 62 is actually send out the http request to the server. The data came from the server is then read as a string (which is XML) and keep in variable **'content'**, this is in line 64.

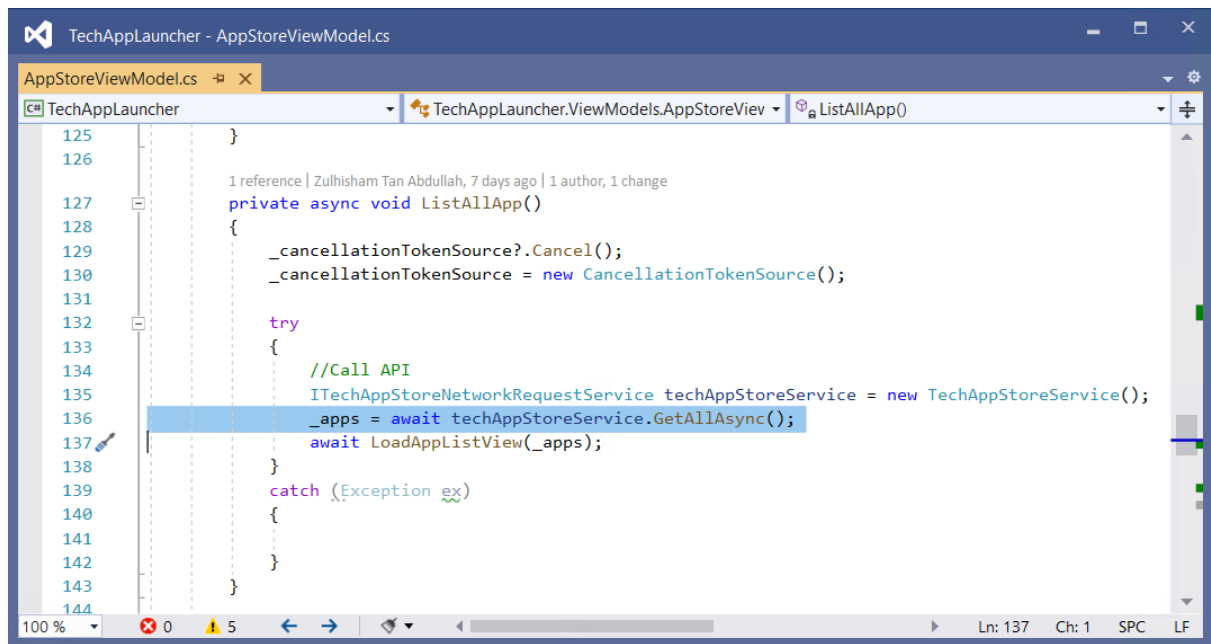
In line 68, the content in variable **'content'** is then deserialize into C# object **'feeds'**. Line 71 to line 83, it picks up those necessary data as **'result'** object and send back to the caller as a list in line 85.

The function is call whenever user click the **'Select App.'** button<sup>[5]</sup>.

The screenshot below shows the code where the function is being called (line 136).





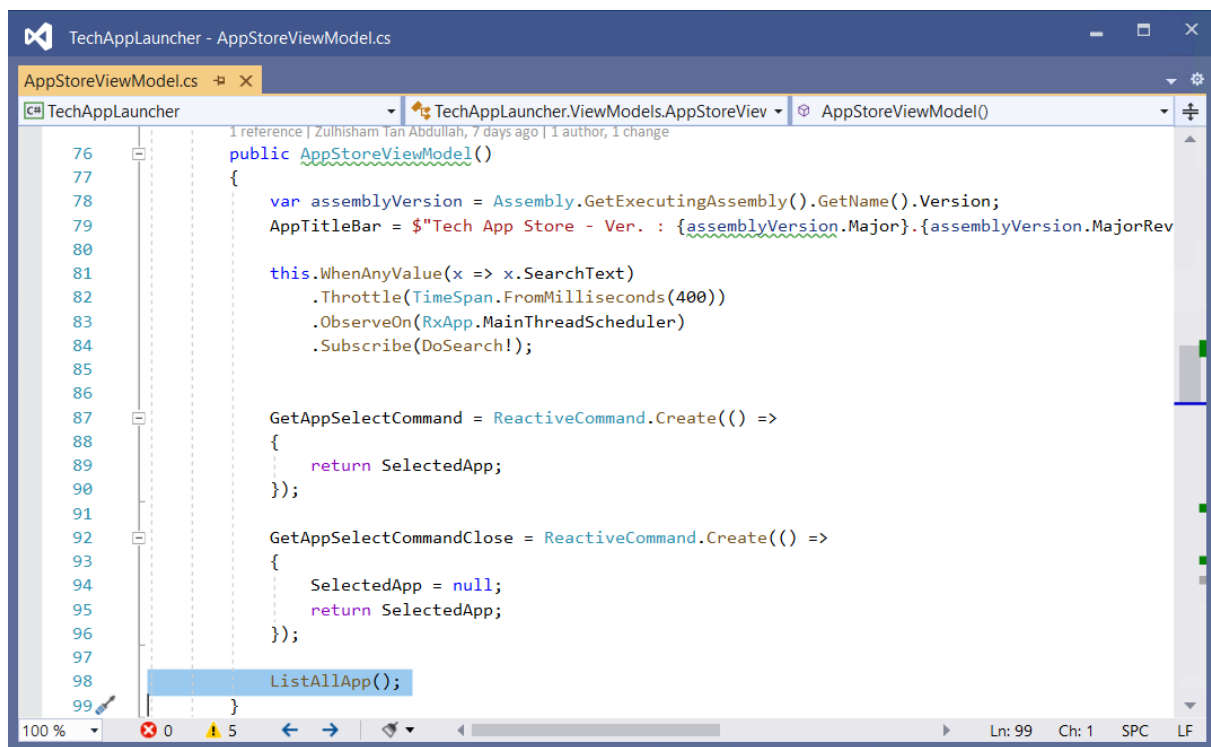


The screenshot shows the Visual Studio IDE with the file 'AppStoreViewModel.cs' open. The 'ListAllApp()' method is selected, and the code is as follows:

```
125     }
126
127     1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
128     private async void ListAllApp()
129     {
130         _cancellationTokenSource?.Cancel();
131         _cancellationTokenSource = new CancellationTokenSource();
132
133         try
134         {
135             //Call API
136             ITechAppStoreNetworkRequestService techAppStoreService = new TechAppStoreService();
137             _apps = await techAppStoreService.GetAllAsync();
138             await LoadAppListView(_apps);
139         }
140         catch (Exception ex)
141         {
142         }
143     }
144
```

The status bar at the bottom indicates 'Ln: 137 Ch: 1 SPC LF'.

The '**ListAllApp()**' function is actually being call from the class's constructor.



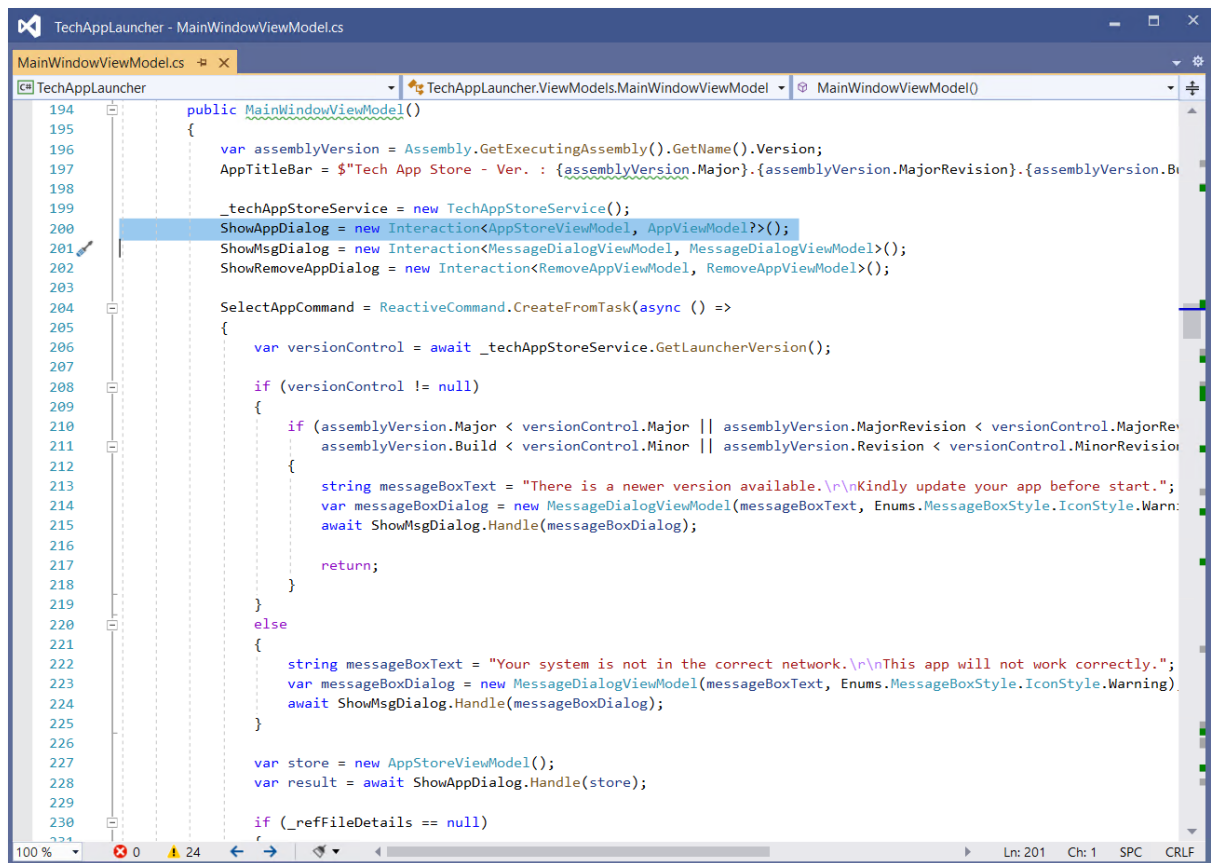
The screenshot shows the Visual Studio IDE with the file 'AppStoreViewModel.cs' open. The 'AppStoreViewModel()' constructor is selected, and the code is as follows:

```
76     1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
77     public AppStoreViewModel()
78     {
79         var assemblyVersion = Assembly.GetExecutingAssembly().GetName().Version;
80         AppTitleBar = $"Tech App Store - Ver. : {assemblyVersion.Major}.{assemblyVersion.Minor}";
81
82         this.WhenAnyValue(x => x.SearchText)
83             .Throttle(TimeSpan.FromMilliseconds(400))
84             .ObserveOn(RxApp.MainThreadScheduler)
85             .Subscribe(DoSearch!);
86
87         GetAppSelectCommand = ReactiveCommand.Create(() =>
88         {
89             return SelectedApp;
90         });
91
92         GetAppSelectCommandClose = ReactiveCommand.Create(() =>
93         {
94             SelectedApp = null;
95             return SelectedApp;
96         });
97
98         ListAllApp();
99     }

```

The status bar at the bottom indicates 'Ln: 99 Ch: 1 SPC LF'.

The '**AppStoreViewModel.cs**' is a view model for '**AppStoreView**' interface.

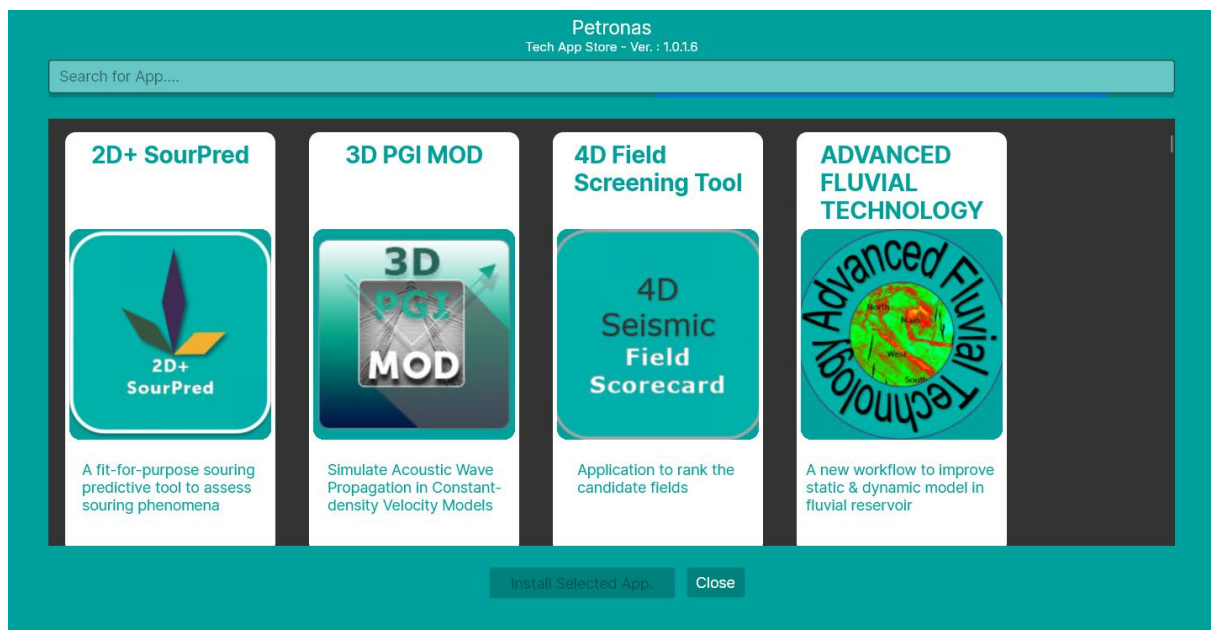


```
194 public MainWindowViewModel()
195 {
196     var assemblyVersion = Assembly.GetExecutingAssembly().GetName().Version;
197     AppTitleBar = $"Tech App Store - Ver. : {assemblyVersion.Major}.{assemblyVersion.Minor}.{assemblyVersion.Build}";
198
199     _techAppStoreService = new TechAppStoreService();
200     ShowAppDialog = new Interaction<AppStoreViewModel, AppViewModel?>();
201     ShowMsgDialog = new Interaction<MessageDialogViewModel, MessageDialogViewModel>();
202     ShowRemoveAppDialog = new Interaction<RemoveAppViewModel, RemoveAppViewModel>();
203
204     SelectAppCommand = ReactiveCommand.CreateFromTask(async () =>
205     {
206         var versionControl = await _techAppStoreService.GetLauncherVersion();
207
208         if (versionControl != null)
209         {
210             if (assemblyVersion.Major < versionControl.Major || assemblyVersion.MajorRevision < versionControl.MajorRevision ||
211                 assemblyVersion.Build < versionControl.Minor || assemblyVersion.Revision < versionControl.MinorRevision)
212             {
213                 string messageBoxText = "There is a newer version available.\r\nKindly update your app before start.";
214                 var messageBoxDialog = new MessageDialogViewModel(messageBoxText, Enums.MessageBoxStyle.IconStyle.Warning);
215                 await ShowMsgDialog.Handle(messageBoxDialog);
216
217                 return;
218             }
219         }
220         else
221         {
222             string messageBoxText = "Your system is not in the correct network.\r\nThis app will not work correctly.";
223             var messageBoxDialog = new MessageDialogViewModel(messageBoxText, Enums.MessageBoxStyle.IconStyle.Warning);
224             await ShowMsgDialog.Handle(messageBoxDialog);
225         }
226
227         var store = new AppStoreViewModel();
228         var result = await ShowAppDialog.Handle(store);
229
230         if (_refFileDetails == null)
```

Now, here is the magic happened in **Avalonia**. From the screenshot above, line 204, this is where the command is being fired when the 'Select App.' Button is pressed. It shows the dialogue of **AppStoreView**. Whenever the user select an app to install, the **AppStoreView** will returns a **result** to the main window which is in line 228 as a **AppViewModel** defined in line 200. The AppViewModel contain all the information about the selected app.

```
TechAppLauncher - AppViewModel.cs
AppViewModel.cs
TechAppLauncher
TechAppLauncher.ViewModels.AppViewModel
_app

17 1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
18 public AppViewModel(Models.App app)
19 {
20     _app = app;
21 }
22
23 1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
24 public string Description => _app.ShortDescription;
25
26 1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
27 public string Title => _app.Title;
28
29 0 references | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
30 public string AppGroup => _app.AppGroup;
31
32 2 references | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
33 public int AppId => _app.ID;
34
35 4 references | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
36 public string AppUID => _app.AppUID;
37 1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
38 public string AppType => _app.AppType;
39 1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
40 public string PluginApp => _app.PluginApp;
41
42 2 references | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
43 public double? AppVersion => _app.AppVersion;
44
45 private Bitmap? _appImg;
46
47 1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
48 public Bitmap? AppImg
49 {
50     get => _appImg;
51     private set => this.RaiseAndSetIfChanged(ref _appImg, value);
52 }
53
54 1 reference | Zulhisham Tan Abdullah, 7 days ago | 1 author, 1 change
55 public async Task LoadAppImage()
```

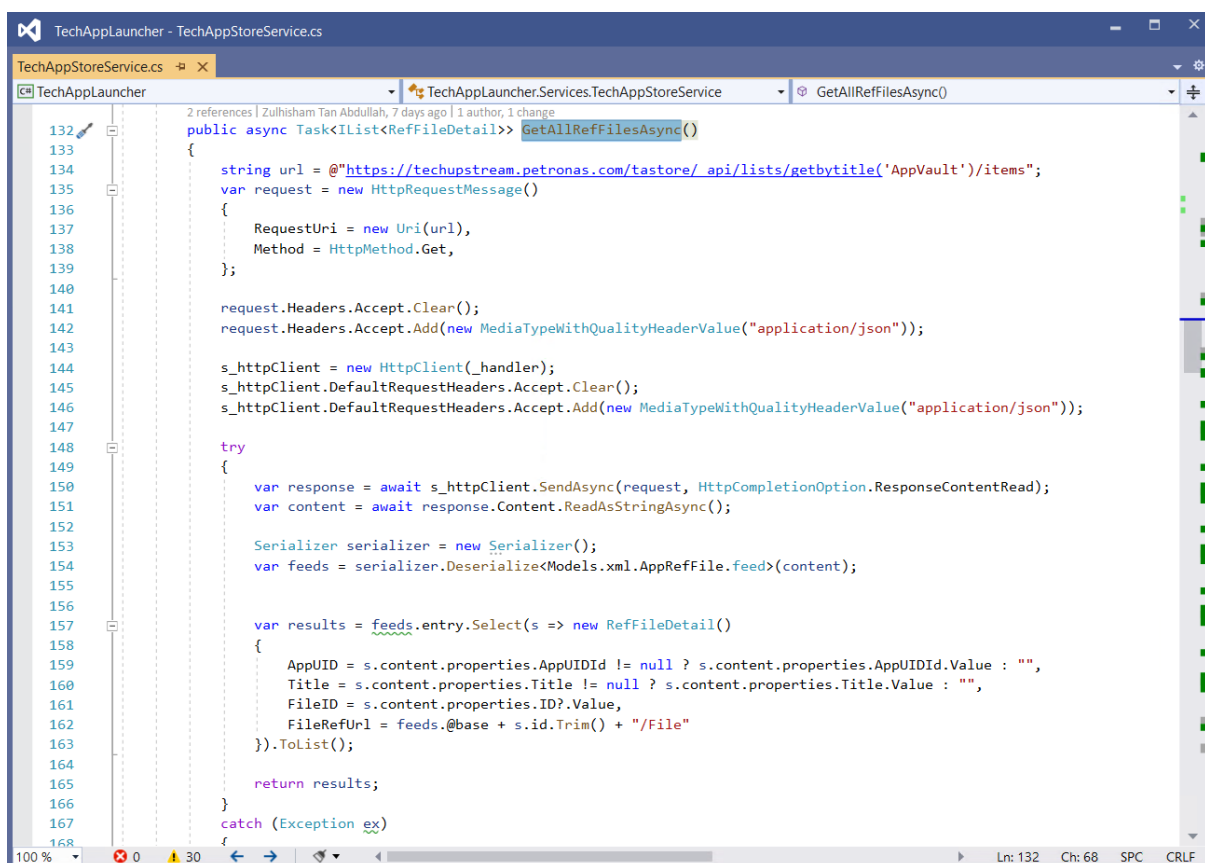


While preparing the interaction user-interface as shown above, the TechAppLauncher make another http call to the Server (line 232 in the following screenshot).



```
227 var store = new AppStoreViewModel();
228 var result = await ShowAppDialog.Handle(store);
229
230 if (_refFileDetails == null)
231 {
232     _refFileDetails = await _techAppStoreService.GetAllRefFilesAsync();
233 }
234
235 bool isLaunchable = false;
236 this.IsLaunchable = isLaunchable;
237 this.IsDownloadable = false;
238
239 Apps.Clear();
240
241 if (result != null && _refFileDetails != null)
242 {
243     Apps.Add(result);
244
245     var refFileDetailSelect = _refFileDetails.Where(n => n.AppUID == result.AppId.ToString()).FirstOrDefault();
246     var refFileUrl = refFileDetailSelect != null ? refFileDetailSelect.FileRefUrl : null;
247 }
```

This is to get all the zip file that attached to the specific application based on **AppUID**. Notice that it is picking up all the necessary data (line 159 to line 162) such as AppUID, File URL, Title of the app and etc...



```
132 public async Task<IList<RefFileDetail>> GetAllRefFilesAsync()
133 {
134     string url = @"https://techupstream.petronas.com/tastore/_api/lists/getbytitle('AppVault')/items";
135     var request = new HttpRequestMessage()
136     {
137         RequestUri = new Uri(url),
138         Method = HttpMethod.Get,
139     };
140
141     request.Headers.Accept.Clear();
142     request.Headers.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
143
144     s_httpClient = new HttpClient(_handler);
145     s_httpClient.DefaultRequestHeaders.Accept.Clear();
146     s_httpClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
147
148     try
149     {
150         var response = await s_httpClient.SendAsync(request, HttpCompletionOption.ResponseContentRead);
151         var content = await response.Content.ReadAsStringAsync();
152
153         Serializer serializer = new Serializer();
154         var feeds = serializer.Deserialize<Models.xml1.AppRefFile.feed>(content);
155
156         var results = feeds.entry.Select(s => new RefFileDetail()
157         {
158             AppUID = s.content.properties.AppUIDId != null ? s.content.properties.AppUIDId.Value : "",
159             Title = s.content.properties.Title != null ? s.content.properties.Title.Value : "",
160             FileID = s.content.properties.ID?.Value,
161             FileRefUrl = feeds.@base + s.id.Trim() + "/File"
162         }).ToList();
163
164         return results;
165     }
166     catch (Exception ex)
167     {
168     }
```

Hence, based of the file URL, the TechAppLauncher knows where to download the attachment files.

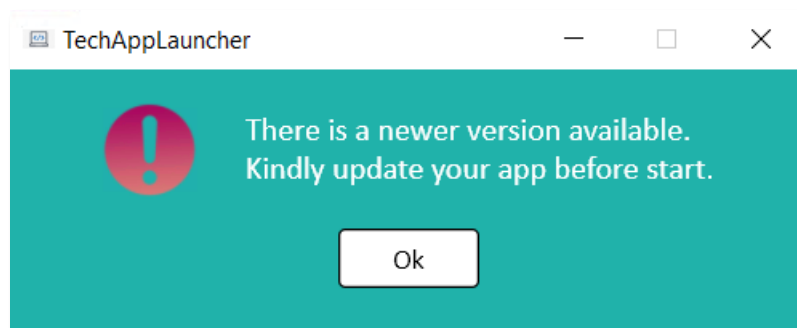


## TechAppLauncherAPI

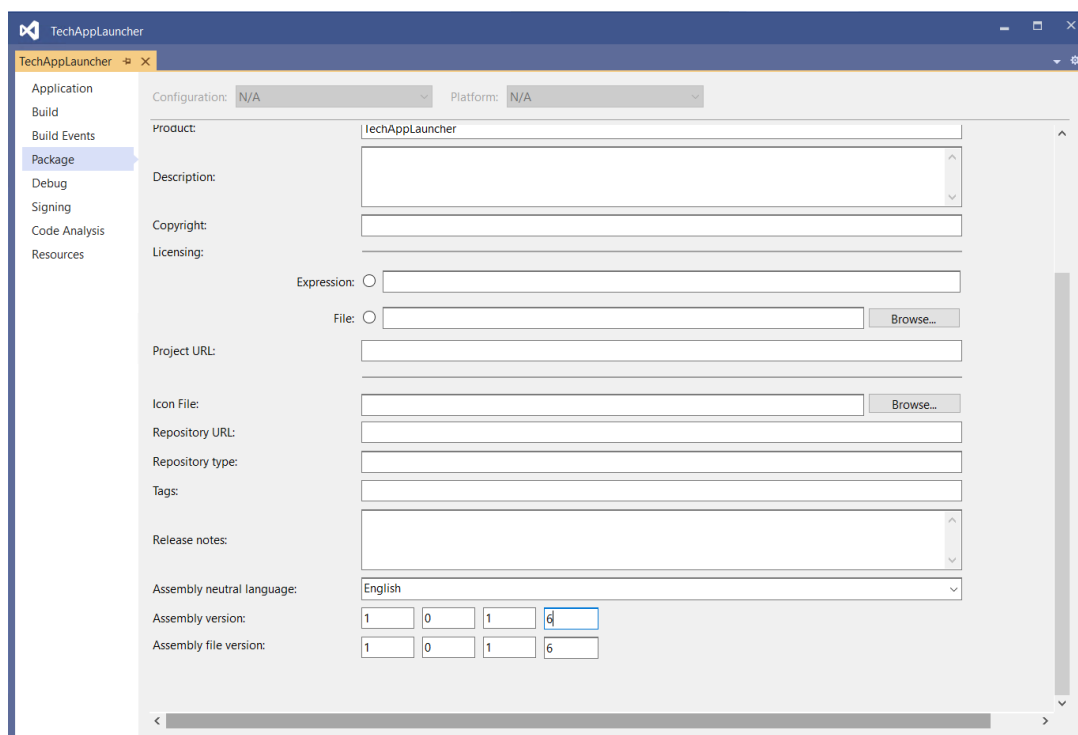
There are few important endpoints that the **TechAppLauncher** connected to:

1. /api/TechAppLauncher/GetLauncherVersion
2. /api/TechAppLauncher/GetlauncherAppConfig
3. /api/TechAppLauncher/GetUserDownloadSessionsByUsername
4. /api/TechAppLauncher/AddUserDownloadSession
5. /api/TechAppLauncher/GetAppDistributionReferenceDetailByAppUID

1. **GetLauncherVersion** – The launcher uses this information to determine whether the client is using the latest version of the application, otherwise it will prompt the user with the message as shown below.



The launcher version number is set at it's project property as shown below. Hence, it is recommended to increase the number every time there is a new update deploy to the server.



You can use the following endpoints to update the version number.

- i. `/api/TechAppLauncher/UpdateAppVerMajorNumber`
- ii. `/api/TechAppLauncher/UpdateAppVerMajorRevisionNumber`
- iii. `/api/TechAppLauncher/UpdateAppVerMinorNumber`
- iv. `/api/TechAppLauncher/UpdateAppVerMinorRevisionNumber`

Let's try update the Launcher's minor revision version number. Open your browser and go to <http://10.14.161.44/TechAppLauncherAPI/swagger/ui/index#/TechAppLauncher>. Make sure your VPN is connected if you are not in the Petronas network. Then look for the endpoint **'/api/TechAppLauncher/UpdateAppVerMinorRevisionNumber'**.

**PATCH** `/api/TechAppLauncher/UpdateAppVerMinorRevisionNumber`

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
<code>minorRevisionVersionNumber</code>	6		body	integer

Parameter content type:

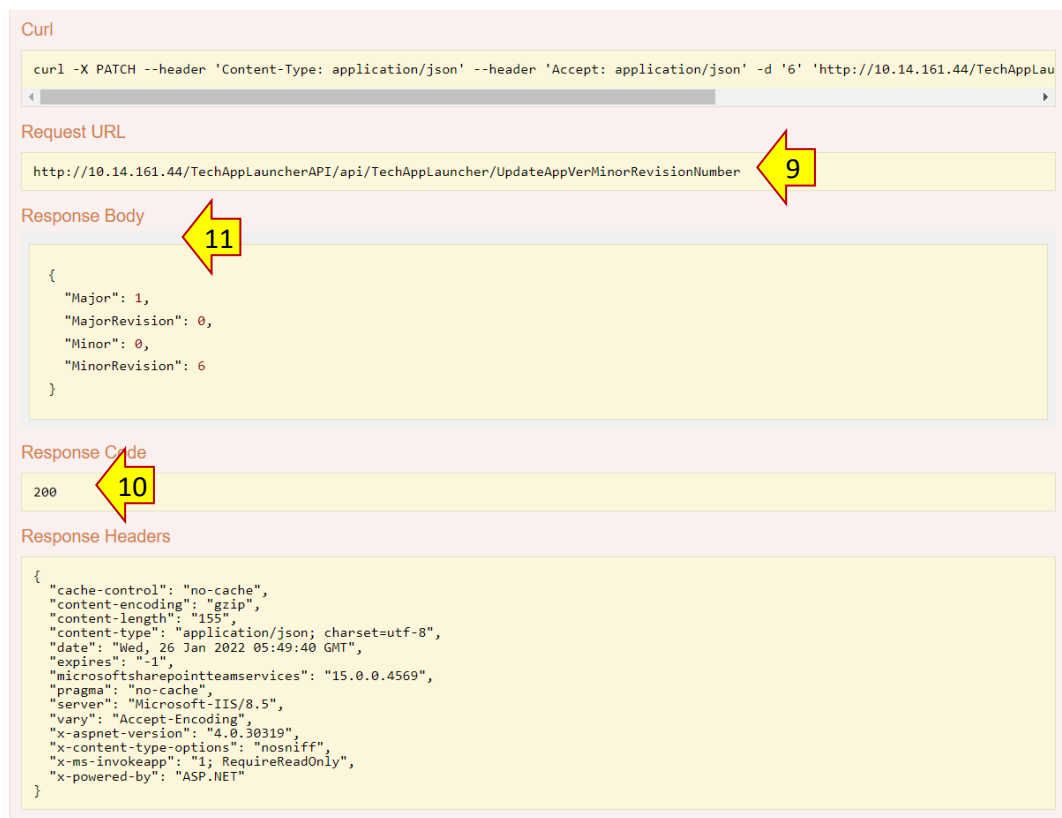
**Response Messages**

HTTP Status Code	Reason	Response Model	Headers
200	Returns the specific record when success. Otherwise returns null.		
400	Invalid Data!		

On the swagger page, the endpoint required a parameter **'minorRevisionVersionNumber'** which is an integer data type<sup>[6]</sup>. Since it is an integer, enter a desire version number<sup>[7]</sup> without any quotation mark. When done, click on the **'Try it out'** button<sup>[8]</sup>.

In the HTTP response as shown in the screenshot at the following screen, the swagger page shown the fully qualified URL<sup>[9]</sup> for the request. The response status code shows that is **200**<sup>[10]</sup> which is **'OK'**. The message body shown the entire updated version number<sup>[11]</sup>.

Note that if the request is fails, it will return **HTTP status code 400** which is a **bad request**. In this case, you need to check the parameter that you gave is meet the data type requirement. Otherwise, the failure may due to the server itself. The possible courses may due to the failure to connect to the database server.



2. **GetlauncherAppConfig** – The launcher uses this endpoint to retrieve the **TechApp Store** server credentials.

Since the server credentials might need to update in a period of time, you can update the credential information with the following endpoints.

- i. `/api/TechAppLauncher/UpdateAppStoreServerDomainName`
- ii. `/api/TechAppLauncher/UpdateAppStoreServerUserName`
- iii. `/api/TechAppLauncher/UpdateAppStoreServerPassword`

Let's try update the **App Store's Server Password**. Open your browser and go to <http://10.14.161.44/TechAppLauncherAPI/swagger/ui/index#/TechAppLauncher>. As usual, you must make sure your VPN is connected if you are not in the Petronas network. Then look for the endpoint `'/api/TechAppLauncher/UpdateAppStoreServerPassword'`.

On the swagger page, the endpoint required a parameter **'password'** which is a string data type<sup>[12]</sup>. Since it is a string data type, enter a desire password<sup>[13]</sup> **with a quotation mark** as shown in the screenshot in following page. When done, click on the **'Try it out'** button<sup>[14]</sup>.

**PATCH** /api/TechAppLauncher/UpdateAppStoreServerPassword

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
password	gfEmW&C19pQ@dF		body	string

Parameter content type: application/json

**Response Messages**

HTTP Status Code	Reason	Response Model	Headers
200	Returns the specific record when success. Otherwise returns null.		
400	Invalid Data!		

Try it out!

In the HTTP response as shown in the screenshot below, the swagger page shown the fully qualified URL<sup>[15]</sup> for the request. The response status code shows that is **200**<sup>[16]</sup> which is 'OK'. The message body shown the entire updated Application Configuration<sup>[17]</sup>.

Note that if the request is fails, it will return **HTTP status code 400** which is a **bad request**. In this case, you need to check the parameter that you gave is meet the data type requirement. Otherwise, the failure may due to the server itself. The possible courses may due to the failure to connect to the database server.

**Request URL**

http://10.14.161.44/TechAppLauncherAPI/api/TechAppLauncher/UpdateAppStoreServerPassword

**Response Body**

```
{
  "Id": 1,
  "LauncherVerMajor": 1,
  "LauncherVerMajorRev": 0,
  "LauncherVerMinor": 0,
  "LauncherVerMinorRev": 6,
  "AppStoreServerDomain": "petronas",
  "AppStoreServerUser": "techus.admin",
  "AppStoreServerPwd": "gfEmW&C19pQ@dF",
  "LauncherInfo": ""
}
```

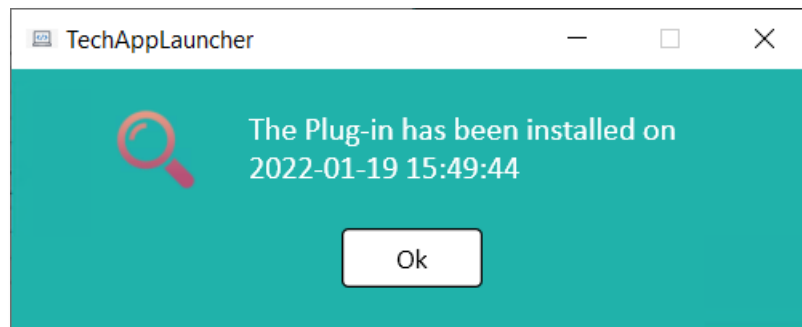
**Response Code**

200

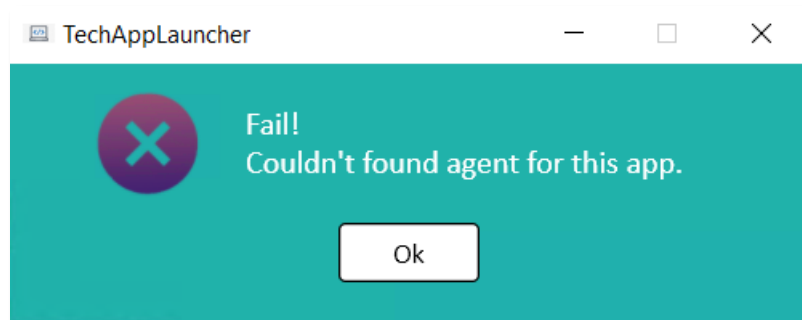
**Response Headers**

```
{
  "cache-control": "no-cache",
  "content-encoding": "gzip",
  "content-length": "246",
  "content-type": "application/json; charset=utf-8",
  "date": "Wed, 26 Jan 2022 06:41:38 GMT",
  "expires": "-1",
  "microsoftsharepointteamservices": "15.0.0.4569",
  "pragma": "no-cache",
  "server": "Microsoft-IIS/8.5",
  "vary": "Accept-Encoding",
  "x-aspnet-version": "4.0.30319",
  "x-content-type-options": "nosniff",
  "x-ms-invokeapp": "1; RequireReadOnly",
  "x-powered-by": "ASP.NET"
}
```

3. **GetUserDownloadSessionsByUsername** – The launcher uses this endpoint to determine whether a specific plugin had already installed at the client workstation. It will prompt the message as shown below if that specific plugin had already installed.



4. **AddUserDownloadSession** – The launcher uses this endpoint to insert new record of User Download Session into the database.
5. **GetAppDistributionReferenceDetailByAppUID** – The launcher uses this endpoint to find for the installation agent for a specific application such as **Software Center**. The launcher will prompt an error message if you try to install a standalone application but there is no information has been setup in the database.



You can use the following endpoint to add and to delete a specific agent.

- i. `/api/TechAppLauncher/AddAppDistributionReferenceDetail`
- ii. `/api/TechAppLauncher/DeleteAppDistributionReferenceDetail`

You can also update the details of a specific agent with the following endpoints.

- i. `/api/TechAppLauncher/UpdateAppDistributionReferenceDetailByAppUID`
- ii. `/api/TechAppLauncher/UpdateAppDistributionReferenceDetailById`

Now, let's try to add an application distribution reference detail. Open your browser and go to <http://10.14.161.44/TechAppLauncherAPI/swagger/ui/index#/TechAppLauncher>. As usual, you must make sure your VPN is connected if you are not in the Petronas network. Then look for the endpoint '`/api/TechAppLauncher/AddAppDistributionReferenceDetail`'.



POST /api/TechAppLauncher/AddAppDistributionReferenceDetail

Parameters

Parameter	Value	Description	Parameter Type	Data Type
appDistRefDet	<pre> {   "LinkId": "ScopeId_03A3D380-BC8F-4E85-8A80-23F2A423C8CF/Application_c81c689b-6695-4ed0-846f-74c99a88c453",   "AgentName": "softwarecenter",   "Description": "PRSB &amp; Ohio University" } </pre>		body	Model

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Returns the specific record when success. Otherwise returns null.		
400	Invalid Data!		

Try it out!

On the swagger page, the endpoint required a parameter 'appDistRefDet' which is an object data type<sup>[18]</sup>. Since it is an object data type, you need to enter a json format data object<sup>[19]</sup> as shown in the code below as an example (refer to the screenshot as well). You can ignore the 'Id' entity since it will be generated by the database server. Whereas, you need to the TechApp Store portal for the AppUID. It is unique, which means you couldn't have it duplicated in the database.

```

{
  "AppUID": "SuperCorp2014",
  "LinkId": "ScopeId_03A3D380-BC8F-4E85-8A80-23F2A423C8CF/Application_c81c689b-6695-4ed0-846f-74c99a88c453",
  "AgentName": "softwarecenter",
  "Description": "PRSB & Ohio University"
}

```

← → ↻ 🔒 Not secure | https://techupstream.petronas.com/tastore/Pages/ApplicationListingAdmin.aspx?nplview=hash7d531fd1-448b-46a4-8658-46782447861c=CascDelWarn... Zulhisham Tan Abdullah (GR&T/PD&T) ✖ ⚙ Not syncing

🏠 Phone Directory 🗂 Solat 👤 Admin 🔄 Application Owner 📖 FAQ

**PETRONAS**  
TECH Apps Store

About Apps Store Highlights Contact Us

Tech Apps Store > TECH Apps Store > Application Listing

Application Listing

🔍 new item or edit this list

AppTitle	AppUID	Version	AppType	AppGroup	Developedby	DevelopedDate	UploadedDate	Approval Status	Rating (0-5)	SortOrder	Created By	Inst
SWC WaveClass 2013	waveformClassification-2013		Plugin	Exploration	Exploration Technology			Approved	☆☆☆☆☆ 0		Administrator	Req.
Relative Amplitude Change 2013	Relative_Amplitude_Change-2013	1	Plugin	Exploration	Exploration Technology			Approved	☆☆☆☆☆ 0		Administrator	Req.
eRGB Blending Workflow	eRGBBlending2015		Others	Exploration	Exploration Technology			Approved	☆☆☆☆☆ 0		Administrator	Not
SuperCorp V3	SuperCorp2014	3	Stand Alone	Fluids	PRSB & Ohio University		8/22/2015	Approved	★★★★☆ 3	1	Administrator	Rest
Laplacian Edge Enhancer 2013	LaplacianFilter2013	1	Plugin	Exploration	Exploration Technology		8/21/2015	Approved	☆☆☆☆☆ 0	1	Administrator	Req.

When done, click on the 'Try it out' button<sup>[14]</sup>.

In the HTTP response as shown in the following screenshot, the swagger page shown the fully qualified URL<sup>[21]</sup> for the request. The response status code shows that is **200**<sup>[22]</sup> which is 'OK'. The message body shown the entire new application distribution reference detail<sup>[23]</sup> which has been successfully added into the database. Note that the payload included the 'Id' property and the entity is given a number by the database server.

Note that if the request is fails, it will return **HTTP status code 400** which is a **bad request**. In this case, you need to check the parameter that you gave is meet with the data type requirement. Otherwise, the failure may due to the server itself. The most possible courses may due to the failure to connect to the database server.

**Request URL**

http://10.14.161.44/TechAppLauncherAPI/api/TechAppLauncher/AddAppDistributionReferenceDetail **21**

**Response Body**

**23**

```
{
  "Id": 14,
  "AppUID": "SuperCorp2014",
  "LinkID": "ScopeId_03A3D380-BC8F-4E85-8A80-23F2A423C8CF/Application_c81c689b-6695-4ed0-846f-74c99a88c453",
  "AgentName": "softwarecenter",
  "Description": "PRSB & Ohio University"
}
```

**Response Code**

**22**

200

**Response Headers**

```
{
  "cache-control": "no-cache",
  "content-encoding": "gzip",
  "content-length": "308",
  "content-type": "application/json; charset=utf-8",
  "date": "Wed, 26 Jan 2022 07:16:56 GMT",
  "expires": "-1",
  "microsoftsharepointteamservices": "15.0.0.4569",
  "pragma": "no-cache",
  "server": "Microsoft-IIS/8.5",
  "vary": "Accept-Encoding",
  "x-aspnet-version": "4.0.30319",
  "x-content-type-options": "nosniff",
  "x-ms-invokeapp": "1; RequireReadOnly",
  "x-powered-by": "ASP.NET"
}
```