**Honeywell Assignment: Video Player**

Steps to run the project:

1.Open the solution in VS 2022.

2. Set both the project (UploadVideoAPI and VideoPlayer) as a startup project.

Right click on the solution ->Configure startup projects->Multiple startup projects->Choose Action as ‘Start’->Apply->Ok

3.Click on start.

**What is working:**

\* The web application is to provide all of its view(s) from one single URL - Home/Index. The default path should direct to Home/Index, so the path can be ommitted and the site can be accessed simply from 'https://localhost:{portNum}/'

\* The web applications front-end should provide two discrete actions that are mutually exclusive - Uploading file(s) - from an upload form view, and viewing the contents of the catalogue available in the server media folder - from a catalogue view.

\* The upload form view should allow the user to browse their clients file system and choose one or more MP4 files. Once files have been selected, they can simply request the files be uploaded to the servers media folder by pressing an upload button.

\* The catalogue view should present all the MP4 files in the servers media folder in a table containing the filename and file size of each file in the catalogue.

\* When the user clicks on an item in the catalogue view table, the video should commence playing back in a video player on the same page. If the user selects another video from the table, it closes the previous one and starts playing back the latest selected one. Only one video can ever been seen playing back at a time.

\* The web application should also host an ASP.NET Core Web API (please do not use minimal APIs for this). This Web API should be used by the upload form for uploading content to the servers media folder.

\* The upload API should support uploading one or more MP4 files at a time. If a file with the same name is uploaded more than once, you can overwrite the pre-exsting file with the new one.

\* Only files with an MP4 extension are allowed to be uploaded into the servers media folder.

\* After the upload form is used to successfully upload content, the user should be shown the updated catalogue view.

\* If an error occurs whilst uploading files to the catalogue, the view should stay on the upload form view and inform the user that an error has occurred.

\* The request to upload content and access content for playback must work as if it were deployed on a remote web server, with relative references to the media from the origin host of the deployed web application. You can assume though that the permissions to the media folder would just work if deployed on a remote server (even though there would be extra configuration required for a real deployment).

Build:

\* The expectation is that we can open the provided solution in Visual Studio 2022, with the .NET 6 SDK & runtime, and optionally Typescript installed - and just run your web application locally using the likes of Kestral (eg. dotnet run) or IIS Express in Visual Studio. Other than NuGet restore, or a tsc, there shouldn't be any need for anything other than the ASP.NET Core application to compile.

**What is not Working:**

\* The upload API should accept uploads of up to 200 megabytes, no more. This limit should be enforced for the upload API only. Uploads greater than 200MB should return an appropriate error code. Any other endpoints exposed by the web application should continue to enforce the default ASP.NET upload limit (which is much less than 200MB).

Note: Due to some time constraint I am not able to resolve this point(200MB file) .

But the application is working for small Files. I have attached the sample video files in the “SampleVideos.zip” folder in the git repository as well as attached in the email.