**Technical Questions**

**1. What would you add to your solution if you had more time?**

The following functionalities can be added to the dashboard that will help the user and improve user experience:

* Upload multiple files at a time (Done)
* Sort files that are uploaded in the view list block based on tag/labels, uploaded date
* Select multiple files and perform delete operation
* Search files based on name, tag/labels, uploaded date
* Archive files that may not be of use now but might be needed in the future
* Rename the file name during upload
* Rename the file name, modify the tag/labels after upload
* Create user login and registration page
* Alert and logging system

**2. How would you track down a performance issue in production with the application you created?**

We can start with checking the *CPU and Memory utilization* in the server to identify which App is causing performance issue. It is recommended that the *App Pools are must be isolated* for each application. This is help narrow down to the app creating the issue. Also, as a temporary workaround, we will be able to reboot the issue causing App Pool.

To track down a performance issue in production, the application must be written to have *multiple levels of logging* system. This means the application should be capable of logging custom messages, success, information, warning, and error. It is not wise to log everything when the application is running fine. So, it is best to have configuration set to turn ON the required logging level when any performance issue occurs.

Once you can track what is happening in the application with the help logs, a *detailed code review can be performed*. We can check for following issues while performing the review:

* Missing closure of database connections
* Unnecessary bulk data fetching queries
* Inefficient coding like not disposing objects that are no longer needed
* Slow web service calls
* Large XML processing
* Unnecessary loops
* More cache usage than required
* Load due to third-party integration

**3. Why did you choose the language/framework/libraries you did to create the application? What was the most useful feature of the language selected?**

Language/framework/libraries:

* Front End: Angular 6, RxJS, HTML5, CSS3, Bootstrap
* Back End: C#, ASP.NET Core WebAPI (MVC)
* Database: SQL Server, ASP.NET Entity Framework Core (Code-First), AutoMapper

Tools

* Microsoft Visual Studio
* Microsoft VS code
* Microsoft SQL Server Express

I chose the following tech-stack for the listed reasons:

**Angular**

* Supports cross- platforms like Windows, iOS, Android, Linux
* Two-way data binding helps in reduction of development time as the model and view are in continuous sync
* Component based model with the use of TypeScript leads to clean code and provides good scope for scalability
* RxJS for efficient asynchronous programming

**ASP.NET Core WebAPI (MVC)**

* Easy code maintenance due to less and structured code
* WebAPI interacts using application/json which makes it compatible to work with modern front-end frameworks like Angular, React
* Supports multiple platforms like Windows, iOS, Android, Linux. The same C# code can be reused to create an iOS or Android app
* ASP.NET Core framework has higher performance and its complier optimizes the code when new enhancements are added
* There is option for future Cloud integration

**ASP.NET Entity Framework Core (Code-First)**

* No need to write database queries. Instead create the domain models and let Entity-Framework generate database and tables.
* No need to write persistence framework (the ADO.NET connections between application and database to store data)

**AutoMapper**

* No manual mapping of models in business and data access layer required