**Coding Exam Overview (S3ntinel)**

**\*\*ASSUMPTIONS\*\***

* **The application will be deployed in a Windows Server**
* **The remote Linux servers have ssh enabled and has ssh key pair setup to able to connect without user or pass.**
* **The Windows Server that hosts the application has windows authentication access to the remote Windows servers.**
* **All ssh key pairs setups (e.g. `. pem` files) are in `C:\.ssh`**
* **All monitoring configurations are in `C:\MonitoringConfigs\monitoring\_configs.json`(i.e. server name, hostname, services, etc.)**
* **Used `ex.Message` for the purpose of descriptive logging but can be changed to ex to get full stack tracing.**

**\*\*HOW TO SETUP AND TEST SOLUTION\*\***

1. **Copy `S3ntinel.Application/Configs/monitoring\_configs.json` to `C:/MonitoringConfigs/monitoring\_configs.json`.**
   1. **NOTE: This can be setup in other drives but code will need to be adjusted in `ServerCheckService.cs`**
2. **Setup ssh keys and put them into `C:/.ssh` folder.**
   1. **NOTE: This can be setup in other drives but configuration in `monitoring\_configs.json` needs to be modified.**
3. **Enable SSH configurations for remote servers.**
4. **Allow user that runs the application for remote access to Windows servers.**

**\*\*SOLUTION STRUCTURE\*\***

* **S3ntinel.Application**
  + **Configs**
    - Temporary placeholder for `monitoring\_configs.json`
  + **Interfaces**
    - **IServerCheckService.cs** – Interface template for Server Check Service.
  + **Services**
    - **ServerCheckService.cs** – Main Class
* **S3ntinel.Domain (Core)**
  + **Interfaces**
    - **IServerMonitor.cs** – Interface template for Linux and Windows Server monitoring.
  + Models
    - **ServerConfig.cs** – Class that will hold the monitoring configurations.
* **S3ntinel.Infrastructure**
  + **LinuxServerChecker.cs** – Linux server monitoring class.
  + **WindowsServerChecker.cs** – Windows server monitoring class.
* **S3ntinel.Test**
  + **UnitTests1.cs** – Unit Tests for `**WindowsServerChecker.cs**` and `**LinuxServerChecker.cs**`.
* **S3ntinel.WebAPI (Presentation)**
  + **Controllers**
    - **ServerMonitoringController.cs** – Main controller.
  + **Program.cs** – Dependency injections
  + **Alerts**
    - **monitoring\_logs-YYYYMMDD.txt** – logging for info, Alerts, Warnings and Errors. To be moved to another path for ease of checking.

**\*\*LOGIC BEHIND IMPLEMENTATIONS\*\***

* Usage Clean/Onion Architecture for separation of layers (Core, Infrastructure, Application and Presentation).
* Use of Interfaces
  + For adherence of checkers to the required methods.
  + For interchangeability of classes.
  + For scalability; ease of adding classes i.e. other implementations of the same methods.
  + Abstracting methods for underlying class implementations.
* Use of configuration file `monitoring\_configs.json`
  + For ease of implementing known server configurations.
* The design also allows the classes to be used in either web or console settings.

**\*\*PREFERRED MODIFICATIONS\*\***

* Have a batch/app that extracts disk space and services information in the servers and have that application send the files on the unifying server instead of remotely accessing the servers for the information. This will significantly speed up the application.
* Use of Asynchronous methods instead of normal methods.
* Customize logging by creating a class for it.
* Putting file paths/queries in configuration file/database for security.
* Secure folders that has configs and ssh keys.
* Having the logging information in separate files e.g. Server1-YYYMMDD.txt, Server2-YYYYMMDD.txt, etc.
* More dynamic automation testing.

**\*\*\*TESTS DONE\*\*\***

* Manual testing for log information:
  + Manual Testing 1

A screenshot of a computer

Description automatically generated

* + Manual Testing 2:

A screenshot of a computer

Description automatically generated

* Automated testing for method inside `WindowsServerChecker.cs`, `LinuxServerChecker.cs` classes using xUnit:

**Failed Testing** - since disk space on my local Windows Server is under the threshold:

A screenshot of a computer

Description automatically generated

**Successful Testing**

A screenshot of a computer

Description automatically generated