Requirements Intake

Congratulations on making it to this stage of the evaluation.  You are obviously very talented as very few people make it to this stage.  As we’ve stated earlier, the companies we represent receive 1000s of resumes for any given role and it is through these difficult assignments where you can differentiate yourself and be noticed.  After completion of this final ‘real scenario’ assignment - there will be a quick technical interview on your delivery then you are ready to be hired.

The project is scoped to be simple and reasonable in size to enable you to demonstrate your enterprise - class skills. Though this is a fictitious example, this scenario is very similar to what you may encounter in your job.

**Overall Objective**

Create the architecture and design of a medical journals publication and subscription system. Implement the system's services and applications.

**Prerequisites**

The following prerequisites should be respected.

* Use one major technology stack for development out of either .NET or Java along with any compatible open technologies.
* Use any database and development environment tools according to the technology stack chosen.
* Do not use any proprietary technologies or tools that are not available for evaluation.

**Functional Requirements**

The system allows publishing and subscribing to medical journals in a secure way. The system implements the following specifications.

* For publishers
* A web portal to upload and manage a list of medical journals
* To upload journals in PDF format
* For public users
* A web portal to find and subscribe to journals of their interest
* A desktop client
* To list and read the subscribed journals
* That does not allow copying the journal’s content in any common way
* Not from web requests/responses
* Not from their file system
* Not from the client user interface (copy to clipboard, screenshot, etc)

Assume any functional details required to achieve the above requirements based on logic and your experience, but follow the KISS principle.

**Other Technical and Non-functional Requirements**

The following list of technical specifications should be adhered to

* Assume missing/unclear requirements to fill in the gaps in the specifications.
* Choose a mix of server and client side technologies for an efficient design.
* Secure the applications and services.
* Use only software enforced protection that does not require any special hardware.

**What we will evaluate**

* Efficacy of your submission: fundamentally how well your solution is able to achieve the assignment objective and solve the stated problem.
* Code quality
* Code modularity
* Application organization across files and within each file - please ensure you follow the framework standards
* Code documentation - balancing between self documenting code and comments
* Unit and integration testing
* Exception handling where available and expected in the frameworks you’re using
* For any technology used, the correct usage of that technology based on consecrated best practices
* Design
* Clarity and completeness of the readme and design documents
* Fitness of solution to problem
* Efficiency of communication flows between frontend and backend, if applicable
* Functional completeness
* Scoring ratio matrix (out of 10), all of these are individually mandatory so don’t skip any:
* Design quality = 2
* Code quality = 3
* Docs and demo quality = 2
* Specifications compliance = 3

**What to deliver**

Demonstration Video

Record the video demonstration of your work using a screencast tool like [screen-o-matic](http://www.screencast-o-matic.com/)(or any other tool you prefer) commenting on the execution of all components. Save the video to your local machine and include it with the delivery package.

Database script

Create manual steps and SQL script files to create the database (if required), its schema, stored procedures, or any seed data you have used for testing. If using code first migrations to manage the database (if required), then also do not forget to mention the steps in the readme.

Readme Document

Create a txt file with the following information

* Instructions to install and configure prerequisites or dependencies, if any
* Instructions to create and initialize the database (if required)
* Assumptions you have made - it is good to explain your thought process and the assumptions you have made
* Requirements that you have not covered in your submission, if any
* Instructions to configure and prepare the source code to build and run properly
* Issues you have faced while completing the assignment, if any
* Constructive feedback for improving the assignment

Design Document

Create a design document containing the following

* High level requirement analysis
* High level presentation of the data model
* Architecture diagrams describing the composition and working of the system, explaining the component interaction and process, control and data flows.
* Explain the breakdown of the system into components with technical implementation details of each component along with the design patterns involved and with reasons that justify your choices.
* Use both visual elements (diagrams) and text descriptions to maximize the amount of information conveyed while keeping the document as compact as possible

Source Code

You should deliver all the implemented source code including any dependencies. For the dependencies that could not be included due to size, the readme file should have proper instructions on how to download and install them.

**What to submit**

Please read this section carefully.

Failing to follow these directions will disqualify you from consideration.

Create and submit an archive named *<your\_name>\_SEM\_Journals\_<Tech>.zip* containing the following:  
(For example JohnDoug\_SEM\_*Journals\_*NET.zip, or MaryDoug\_SEM\_*Journals*\_Java.zip)

* *<your\_name>\_SEM\_Journals\_<Tech>.zip*
* *<your\_name>\_SEM\_Journals\_<Tech>.zip \Readme.txt*
* *<your\_name>\_SEM\_Journals\_<Tech>.zip \Design.doc (or pdf etc)*
* *<your\_name>\_SEM\_Journals\_<Tech>.zip \Demo\    < this folder contains the screencast video recording*
* *<your\_name>\_SEM\_Journals\_<Tech>.zip \Source\    < this folder contains the complete source code*

ATTENTION! YOUR APPLICATION WILL BE REJECTED IF IT:

- Does not compile

- Does not contain unit tests

- Unit tests are failing

**Design**

Should contain following **projects**:

* Site - from requirement
* Desktop Application - requirement
* WebService - for Desktop application
* Core - generic part for Site and WebService, data access, entities
* Tests - by requirement

**Data**

I choose code first approach. Would create a DB project.

**Data initialization steps**:

* export DB project and copy its connection string
* paste the connection strings to Web.config of Core, Site and WcfService
* type update-database to package manager console in VS2015 for SEM.Core project selected in nuget console.
* Modify connection string to suffix-test and add it to test project.
* run update-database for SEM.Tests project

I started from custom integrations but choosen to use automatic code-first integrations to rapid pre-production development phase.

**Core**

**Data model** - As the project should be as simple as possible, I choose **MVC** with **EntityFramework** as ORM.

Entities:

* Journal (Id, AuthorUserId, Title, FileBytes)
* Subscription (Id, SubscriberId, JournalId)
* ASP .NET authentication and roles entities.
* User
* +Subscriptions custom field
* User roles: Publisher - grants right to access Publishing page.

**Site**

Should inherit **EntityFramework** **Data model** from Core project.- As the project should be as simple as possible, I choose **MVC** with **EntityFramework** as ORM.

Controllers:

* JournalsController - allows to add publications for publishers
* SubscriptionsController - allows to subscribe for users

**WCF Service**

Need to be used with Desktop Application. It could check authority of application with App certificate. Authentication system should be providen.

[OperationContract]

byte[] GetJournal(int value); // Download to view

[OperationContract]

ICollection<Subscription> GetMySubscriptions(); // IDs

**Desktop Application**

By security requirements, Desktop Application should use **secure connection to WCF** service.

PDF viewer choosen to be non proprietary by requirement. Looks like best fit would be **opensource** project [**PDF.js**](https://mozilla.github.io/pdf.js/)by Mozilla. It should be published to NuGet repository because it not there yet.

For most simplicity and bonus portability **Universal Windows** application type choosen.

Design simple as possible. Screens:

* Login
* List of subscriptions
* PDF

**Tests**

Static analysis of the project contracts should be used to ensure all development decisions are met through continuous development process.

Coverage targets:

* Core contracts
* Web Site controllers
* WCF service API