Only file with type '.zip, application/zip' is acceptable. Make sure the size of your delivery file does not exceed 100 MB

## **Instructions**

Please read through your assignment very carefully before beginning your solution. Try to complete as much of the solution as possible in the given timeframe. To upload your solution, click the "SELECT FILE" button above. After selecting your solution, click "UPLOAD". You will be able to review your submission after uploading it.

If you have any questions, submit a Support Request. However, your questions should pertain to the test-taking procedure rather than the content of the assignment. If you had any problems and need an extension, please feel free to Request an Extension. We will determine the extension period based on the reason of the request.

## **Assignment**

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.

Please read the complete specifications and make notes important for you. This is a comprehensive test to check many skills and so details are important.

# **Overall objective**

Create the architecture and design of a banking system. Implement the system's services and applications.

# **Prerequisites**

The following prerequisites must be respected.

1. You can code in C#.NET using windows based development tools and related technologies.
2. Use any database and development tools according to the technologies chosen.
3. Use Visual Studio any version (free trial can be downloaded). Use any nuget packages required to use APIs/frameworks.
4. Do not use any proprietary technologies or tools that are not available for free or for trial.

# **Functional requirements**

Your company works for a multinational bank, that operates branches in many countries. You need to architect and design a banking system for this client. The following set of functional specifications are given by various stakeholders that you need to take care of.

1. Need a customer banking portal with the following facilities
   1. Intra-bank account-to-account money transfers. These transfers could be made between bank accounts across countries in different currencies. The bank charges 2% for currency exchange. The bank does not maintain currency rates and instead wants to use an online API (use any you can find).
   2. The customer receives a guaranteed, but may be delayed, email alert for every transaction in his accounts.
   3. The portal supports authentication from any one external provider like Facebook, Twitter, LinkedIn and Google.
2. Need an application that could be installed on the bank's ATM machines (windows desktop based).
   1. Shall support basic operations like balance check, withdrawal, PIN change, etc.
3. Need a bill payment system
   1. Using this system external websites are able to integrate online payment facility for this bank.

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

# **Technical and non-functional requirements**

The following list of technical specifications must be adhered to

1. Choose the latest popular technologies for which you won't need specially skilled development team.
2. Use of the following patterns is recommended
   1. Back-end REST API on a MVC framework, along with an ORM
   2. Front-end MVVM framework, HTML5 and CSS3 based responsive framework
   3. EDA and SOA mixed architecture
3. Some coverage of proofing automated tests.
4. Secure the applications and services.
5. Deployment possibilities
   1. Load balanced services and applications on-premises or on cloud.
   2. Database load balancing and fail safe configurations.

# **What we will evaluate**

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

# **What to deliver**

## **Demonstration video**

Record the video demonstration of your work using a screen-cast 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. Also do not forget to mention the steps to run the scripts in the readme.

## **Readme document**

Create a text file with the following information

1. Instructions to install and configure prerequisites or dependencies, if any
2. Instructions to create and initialize the database (if required)
3. Requirements that you have not covered in your submission, if any
4. Instructions to configure and prepare the source code to build and run properly
5. Issues you have faced while completing the assignment, if any
6. Constructive feedback for improving the assignment

## **Design document**

Create a design document containing the following info using any design templates that you are used to

1. The solution and system architecture involving the system with interfacing and deployment scenarios
2. High level requirements analysis and the assumptions you have made
3. High level presentation of the data or domain model
4. If necessary, architecture diagrams describing the composition and working of the system, explaining the component interaction and process, control and data flows.
5. Technical implementation summary along with the design patterns involved and with reasons that justify your choices.
6. Use both visual elements (diagrams) and text descriptions to maximize the amount of information conveyed while keeping the document as compact as possible (about 10 pages)

## **Source code**

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

# **How to submit**

Please read this section carefully.

Failing to follow these directions will fail your submission.

Create and submit an archive named *<your\_name>\_CA\_Net\_Banking.zip* containing the following:

* *<your\_name>\_CA\_Net\_Banking.zip*
* *<your\_name>\_CA\_Net\_Banking.zip \Readme.txt*
* *<your\_name>\_CA\_Net\_Banking.zip \Design.doc (or pdf etc)*
* *<your\_name>\_CA\_Net\_Banking.zip \Demo\ < this folder contains the screen-cast video recording*
* *<your\_name>\_CA\_Net\_Banking.zip \Source\ < this folder contains the complete source code*

Check that the size of the archive is less than 80MB. If not, reduce the size of the demo video by removing similar frames and remove the 3rd party binary dependencies.

If your submission does not contain either of the following, it will fail

1. Design document with architecture
2. Readme document with code setup/build/run instructions
3. Video demo showing implemented functions
4. Automated unit/integration tests