Installation Instructions:

SERVER

In order to start the server you should configure the server-settings
section in the app.config file. The server application is a console application with a WCF self-hosted service.

The server has to be run either with administrator privileges or permissions should be added to the current user by running these commands in Powershell as an Administrator:

```
netsh http add urlacl url=http://+:8081/IConversionService user=<LocalUser>
netsh http add urlacl url=http://+:8081/IConversionService/PdfService user=<LocalUser>
```

The easiest way is to run it as an Administrator(either the executable or Visual Studio).

The link for the service can of course be changed, then also it has to be updated in the client's app.config.

The project requires .NET 4.5, Entity Framework 5 and Visual Studio 2012. The machine should have an SQL server installed, preferably an SQL Express default instance, so that auto generation of the database by Entity Framework runs smoothly. Beware, only EF5 supports enums in tables so any previous version will break the app. To use EF5 .net 4.5 is needed.

CLIENT

In order to run the client you should configure the client-settings from app.config in the client project.

The client app does not require any special privileges; it has to be allowed to pass through firewall to connect to the WCF service on the server side.

The application can handle general image files. It was tested with png, jpg and tiff; other image formats should be supported by the 3rd party library. Currently, the converter scales all submitted files to the size of the page, this can be improved if needed.

Validation and QA:

The PdfConversion.Architecture project presents the high level design of the application. It contains a components diagram to present the components of the application. It also has a layers diagram that presents the architecture of the server.

Tests were run on each component in isolation using the Fakes framework for stubs. The test coverage of the application is 94%. Detailed information on coverage and metrics can be found in Visual Studio.

Integration tests were manually run on the application. Some UI testing was done, recorded and automated. Most of the UI testing was done manually and not recorded.

Stress tests were run with a total of 500 files dropped at once for conversion and 5 clients opened simultaneously and the application performed correctly. The profiling revealed that the most time was spent in the conversion routines (as expected).

Most of the modules were tested and the application was tested and it seems stable. All automated tests pass and the manual testing proved the application is stable.

Requirements:

All requirements were implemented.

Users are able to submit files and check the file status and server status.

Users are able to view files with the locally installed PDF reader. Multiple instances can be run on different machines (app.config must be configured on each in order to point to the appropriate shared folder and wcf endpoint).

The Server monitors files on a shared folder, picks them up and converts them in parallel. The server is able to convert Image formats (was tested with jpg, tiff, png) to pdfs. The server provides an API to see the server and file status. Several new statuses were added(missing – in case somebody asks about a file that was not submitted, erroneous – some wrong format of file was submitted and could not be converted).

Delivered:

A zip archive was delivered containing source files and all project and configuration files.

Visual Studio 2012 should build all projects fine, if not contact me at Alexandru.Gyori@studentpartner.com

The binaries were not included for size considerations.