# Document Generation using Aspose with Microservice Architecture

**Project Team**

| Sl. No. | Reg. No. | Student Name |
|---|---|---|
| 1 | 18ETCS002130 | Suyash Jaiswal |

**Supervisors: Naveeta Kharb**

**August – 2018**

**B. Tech. in Computer Science and Engineering**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES**

**Bengaluru -560 054**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

# *Certificate*

*This is to certify that the Project titled "* Document Generation using Aspose with Microservice Architecture *" is a bonafide work carried out in the Department of Computer Science and Engineering by <Suyash Jaiswal > bearing Reg. No. 18ETCS002130, respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.*

**August – 2018**

**Naveeta Kharb**
**<Supervisor name>**

**<HoD CSE Name>**
**Professor and Head – Dept. of CSE**

**<Dean FET Name>**
**Professor and Dean-FET**

*ON*

*Document Generation using Aspose with Microservice Architecture*

The project work is submitted in partial fulfilment of academic requirements for the award of B. Tech. Degree in the Department of Computer Science and Engineering of the Faculty of Engineering and Technology of Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of the text and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of University regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

| Sl. No. | Reg. No. | Student Name | Signature |
|---|---|---|---|
| 1 | 18ETCS002130 | Suyash Jaiswal | |

**Date : XX August 2018**

# Acknowledgements

It is with extreme pleasure and pride that we present our B-Tech. dissertation titled "Document Generation using Aspose with Microservice Architecture". We would like to express our sincere thanks and gratitude to the following people, who stood by us throughout, helping us with much required inputs, guidance, knowledge and supported us.

We take great pleasure to express our sincere thanks and gratitude to academic project guide < Naveeta Kharb>**.** Asst. Professor Department of CSE, for her support, guidance and suggestions throughout the project which is leading this project for the completion.

We express our sincere thanks to**,** , our respected Dean and to **,** Head of Department of Computer Science and Engineering, for their kind cooperation and support toward out dissertation, and to the management of Ramaiah University of Applied Science for their continued support. We are thankful to the staff members of the Computer Science and Engineering, RUAS for giving us good support and suggestion.

Lastly, we would like to thanks our parents and friends for their continued support, encouragement and motivation and God for paving our way of success in this object.

# Abstract

A C# based code will perform all the necessary functions by taking in the desired input. ASP.NET Core 6 has been used to write all the ASPOSE code within it. Once all the inputs are given by the user and the program is run, it will automatically perform the functionalities like creation, document type change, encryption and decryption using the microservices. CQRS pattern has been put to use in this project.

# Table of Contents

# List of Tables



# List of Figures

# 1. Introduction

In this chapter, the project is introduced with its theme followed by its purpose. Then a literature survey is documented stating the most popular existing applications under the domain of the current project and their key features. In the end, the current project and its output will be compared with the existing application and in the end, key features of the current project will be evaluated.

## 1.1 Introduction

**Aspose for .NET** is a modern and professional PDF API used to create, read-write, modify, and make other operations with PDF files without any external dependencies in a .NET application. You can use Aspose for .NET in any programming language for .NET or .NET Core. This component is written in managed C# and it allows developers to add PDF creation and manipulation functionality to their Microsoft .NET applications (WinForms, WPF, ASP.NET and .NET Compact Framework). Using this library, you can implement rich capabilities for creating PDF files from scratch, or completely process existing PDF documents without installing Adobe Acrobat.

### 1.2 Literature Survey

Below is the tabulated study of Papers available on this domain. The results are as follows.

| Sl No. | Site name | Type and year of publication | Findings in research | Conclusions derived via authors | Limitation | Conclusion |
|---|---|---|---|---|---|---|
| 1 | Ilovepdf | Web app, 2018 | Implemented using Javascript, HTML and CSS. | This application is Used by browser. | The Technology Isn't cost Effective. | New Framework for others |

**Table 1 Literature Survey**

### 1.3 Conclusion

In this chapter, different document editor lists available in the market were discussed extensively with their features so that the proposed application can have a good comparison to what is available in the market. Also, technical papers with respect to document-editor application were surveyed to get the knowledge of past work done in this domain. This will help in getting a clear idea of the innovation associated with this project.

# 2. Background Theory

In this chapter, all theories related to the proposed project including technical aspects and resources are explained. This section includes the discussion of the topics on which the proposed project is built, and the entire theory will be summarized via a summary.

## 2.1 Background Theory:

This section gives complete knowledge and understanding of different technologies and frameworks that were required in this project for its completion. The background Theory is listed below:

### 2.1.1 Microservices

Microservice architecture, or simply microservices, is a distinctive method of developing software systems that tries to focus on building single-function modules with well-defined interfaces and operations. The trend has grown popular in recent years as Enterprises look to become more Agile and move towards a DevOps and continuous testing.

Microservices have many benefits for Agile and DevOps teams. Netflix, eBay, Amazon, Twitter, PayPal, and other tech stars have all evolved from monolithic to microservices architecture. Unlike microservices, a monolith application is built as a single, autonomous unit. This makes changes to the application slow as it affects the entire system. A modification made to a small section of code might require building and deploying an entirely new version of software. Scaling specific functions of an application, also means it must scale the entire application.

Microservices solve these challenges of monolithic systems by being as modular as possible. In the simplest form, they help build an application as a suite of small services, each running in its own process and are independently deployable. These services may be written in different programming languages and may use different data storage techniques. While this results in the development of systems that are scalable and flexible, it needs a dynamic makeover. Microservices are often connected via APIs and can leverage many of the same tools and solutions that have grown in the RESTful and web service ecosystem. Testing these APIs can help validate the flow of data and information throughout your microservice deployment.

### 2.1.2 MVC

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

**Model**

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

**View**

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

**Controller**

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

### 2.1.3 CQRS

CQRS is one of the important patterns when querying between microservices. We can use CQRS design pattern to avoid complex queries to get rid of inefficient joins. CQRS stands for Command and Query Responsibility Segregation. Basically, this pattern separates read and update operations for a database.

Normally, in monolithic applications, most of time we have 1 database and this database should respond both query and update operations. That means a database is both working for complex join queries, and also perform CRUD operations. But if the application goes more complex this query and crud operations will be also is going to be un-manageable situation.

In example of reading database, if your application required some query that needs to **join more than 10 table**, this will lock the database due to latency of **query computation**. Also, if we give example of writing database, when performing crud operations we would need to make complex validations and process long business logics, so this will **cause** to **lock database** operations.

So, reading and writing database has different approaches that we can define different strategy to handle that operation. In order to that CQRS offers to use “separation of concerns” principles and separate reading database and the writing database with 2 database. By this way we can even use different database for reading and writing database types like using no-sql for reading and using relational database for crud operations.

Another consideration is we should understand our application use case behaviors, if our application is mostly reading use cases and not writing so much, we can say our application is read-incentive application. So we should design our architecture as per our reading requirements with focusing reading databases.

So, we can say that **CQRS separates** reads and writes into **different databases**, Commands performs update data, Queries performs read data.

Co**mmands** should be actions with task-based operations like “add item into shopping cart” or “checkout order”. So, commands can be handled with message broker systems that provide to process commands in async way.

**Queries** is never modifying the database. Queries always return the JSON data with DTO objects. By this way, we can isolate the Commands and Queries.

### 2.1.4 Aspose

**Aspose for .NET** is a modern and professional PDF API used to create, read-write, modify, and make other operations with PDF files without any external dependencies in a .NET application. You can use Aspose for .NET in any programming language for .NET or .NET Core. This component is written in managed C# and it allows developers to add PDF creation and manipulation functionality to their Microsoft .NET applications (WinForms, WPF, ASP.NET and .NET Compact Framework). Using this library, you can implement rich capabilities for creating PDF files from scratch, or completely process existing PDF documents without installing Adobe Acrobat.
The Aspose.PDF for .NET supports a wide variety of functions such as:

- document compression
- table creation and manipulation
- support for graph objects
- extensive hyperlink functionality
- extended security controls
- custom font handling
- integration with data sources
- add or remove bookmarks
- create a table of contents
- add, update, delete attachments and annotations
- import or export PDF form data

- add, replace or remove text and images
- split, concatenate, extract or inset pages
- transform pages to image
- print PDF documents and much more.

### 2.2 Background of Existing Application

For an individual, the best to-do application is the one that works for him. In the present market, there are many online document editors that can be visited from the web or installedin a person's desktop. Most of the applications (including applications in this particular

domain) are free whereas few of them are paid. Different document editors have different ways of execution along with overlapping features. This is the reason why sometimes it becomes difficult to select the best of them among many in the present market. Hence, a few of the popular applications are mentioned below with their key features listed. This is done in order to know the existing application in this domain along with their features so that the proposed project can have adequate references and the work done in this project can imply something new and innovative which can further contribute to the advancement of this domain.

- Google docs:
    - Easy to maintain
- Visme
    - Customizable templates
- IlovePDF
    - PDF Conversion

## 2.3 Conclusion

In this Chapter, background knowledge for completion of this project is explained elaborately. Also, in each mentioned resource and technology, the reason for its use is also expressed.

# 3. Aim and Objectives

This chapter focuses on the defined title and Aim of the project correctly and clearly. Later this chapter also includes the required objectives that needed to be fulfilled to complete this project. Functional Requirements are well documented in this section since it is required to design different diagrams leading to a complete view of theproject. This is followed by method and methodologies that tabulates the procedure thatwill be followed to complete the objectives. This section then ends with a summary.

## 3.1 Title

Document Generation using Aspose with Microservice Architecture.

## 3.2 Aim

To provide a better format for creation and usage of microservices to improve performance and scalability, enabling technologists to effectively identify and resolve availability and observability issues.

## 3.3 Objectives

The objectives of the proposed Project are listed below:

- To perform literature survey.
- To collect the required dataset for the system.
- To analyze the algorithms used in the implementation.
- To test and validate the built system for various scenarios.
- To document the result by unifying all the results and outcomes.

### 3.4 Functional Requirements

The functional requirements for this project are mentioned below:

FR 1. The user must be able to create a document as per inputs.

FR 2. The user must be able to set privileges to the document.

FR 3. The user must be able to encrypt the document.

FR 4. The user must be able to decrypt the document.

FR 5. The user must be able to change the document format.

### 3.5 Method and Methodology

**Table 2 Methods and Methodology**

| Obj No. | Statement of the Objective | Method/ Methodology |
|---|---|---|
| 1 | To perform literature survey. | 1.1 Literature review on document generation system will be carried out by referring reviewed journals, books, manuals, related documents, by interviewing users of the existing applications.<br>1.2 Based on the survey, the characteristicsof a document generator and related methodologies will be documented. |

| | | |
|---|---|---|
| 2 | To collect the required dataset for the system. | 2.1 Based on the characteristics identified in the survey, a generic list of requirementsfor a document generator will be generated. |
| | | 2.2 The functional and non-functional requirements of the code-editor will be derived after conducting a risk analysis on the feasibility of development effort with respect to time, manpower, cost and technology.<br>2.3 An analysis model of the application will be developed using sequence diagram.<br>2.4 A Software Requirement Specification (SRS) document will be created. |
| 3 | To analyze the algorithms used in the implementation. | 3.1 Based on the SRS document, a high-level design specification will be created using sequence diagram.<br>3.2 Best suiting algorithm is found |
| 4 | To test and validate the built system for various scenarios. | 4.1 Testing and validation using POSTMAN is done to check for various inputs. |
| 5 | To document the result by unifying all the results and outcomes. | 5.1 Develop a scientific project report as per the template specified. |

### 3.6 Conclusion

This chapter focused on defined title and Aim of the project correctly. Later this chapter included required objectives that were required to be fulfilled to complete this project. Functional Requirements is documented successfully in this section followed by method and methodologies which is successfully tabulated to knowthe steps used in completing the objectives including resources used.

# 4. Problem Solving

In this section, the actual dissection of project is shown and each module is built piece by piece in order to complete the project. In design section, the application has been built in accordance with functional requirements based on which diagrams like Sequence Diagram is drawn. In implementation section, snips of important code is displayed with their explanation given below and the result is analyzed resulting in status of test condition.

### 4.1 Design

Design is necessary when it comes to development since it acts as a blueprint for entire process from requirement making to finished (final product). Hence, in this section designs specific to this project like sequence diagram is attached.

## 4.1 **Implementation**

```
// initialize document object
Document document = new Document();
// add a page
Page page = document.Pages.Add();
// add text to the new page
page.Paragraphs.Add(new Aspose.Pdf.Text.TextFragment("Hello World!"));
// save PDF document
document.Save(dir + "output.pdf");
```

**Fig 4.1.1 Implementation of creation of a PDF document**

```
// load the file to be converted
var pfile = new Aspose.Pdf.Document(dir + "template.pdf");
// save in different formats
pfile.Save(dir + "output.docx", Aspose.Pdf.SaveFormat.DocX);
pfile.Save(dir + "output.pptx", Aspose.Pdf.SaveFormat.Pptx);
pfile.Save(dir + "output.html", Aspose.Pdf.SaveFormat.Html);
```

**Fig 4.1.2 Implementation of changing doc type from PDF document**

```
namespace Aspose.Pdf.Examples.CSharp.AsposePDFFacades.SecuritySignatures
{
    0 references | 0 changes | 0 authors, 0 changes
    public class SetPrivilegesOnFile
    {
        0 references | 0 changes | 0 authors, 0 changes
        public static void Run()
        {
            // ExStart:SetPrivilegesOnFile
            // The path to the documents directory.
            string dataDir = "C:\\Users\\sujaiswal\\Downloads\\Document1.pdf";

            // Create DocumentPrivileges object
            DocumentPrivilege privilege = DocumentPrivilege.ForbidAll;
            privilege.ChangeAllowLevel = 1;
            privilege.AllowPrint = true;
            privilege.AllowCopy = true;

            // Create PdfFileSecurity object
            PdfFileSecurity fileSecurity = new PdfFileSecurity();
            fileSecurity.BindPdf(dataDir + "input.pdf");

            // Set document privileges
            fileSecurity.SetPrivilege(privilege);
            fileSecurity.Save(dataDir + "SetPrivilegesOnFile_out.pdf");
            // ExEnd:SetPrivilegesOnFile
```

**Fig 4.1.3 Implementation of setting privileges in a PDF document**

```
// Open document
Document document = new Document("Encrypt.pdf");
// Encrypt PDF
document.Encrypt("user_password", "owner_password", 0 /*permissions*/, CryptoAlgorithm.RC4x128);
// Save updated PDF
document.Save("Encrypted PDF.pdf");
```

**Fig 4.1.4 Implementation of encryption in a PDF document**

```
// Open document by specifying its user's or owner's password
Document document = new Document("Encrypted PDF.pdf", "user_password");
// Decrypt PDF
document.Decrypt();
// Save updated PDF
document.Save("Decrypted PDF.pdf");
```

**Fig 4.1.5 Implementation of decryption in a PDF document**

4.2 **Testing**

All functional requirements are tested and are tabulated below:

**Table 3 Test Cases**

| Sl No. | FR No. | Expected | Obtained | Result |
|---|---|---|---|---|
| 1 | FR 1 | The user must be able to write the code. | The user was successfully able to write the code. | PASS |
| 2 | FR 2 | The user must be able to delete a written code. | The user was successfully able to delete the written code. | PASS |
| 3 | FR 3 | The user must be able to edit the code. | The user was successfully able to edit the code. | PASS |
| 4 | FR 4 | The user must be able to set their favorite language. | The user was successfully able to set their favorite language. | PASS |
| 5 | FR 5 | The user must be able to compile the code. | The user was successfully able to compile the code. | PASS |
| 6 | FR 6 | The user must be able to see the result. | The user is successfully able to see the result. | PASS |

# 5. Results


**Fig 5.1 Result of encryption in a PDF document**

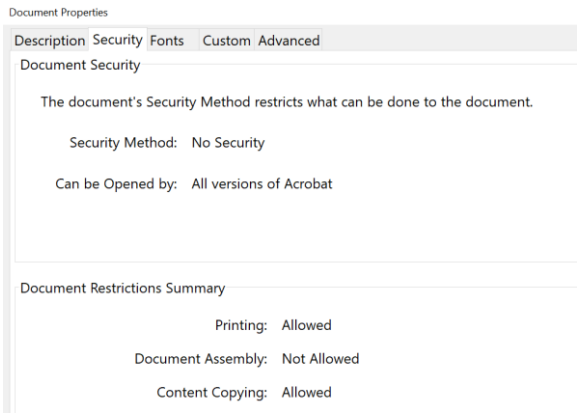


**Fig 5.2 Result of setting privileges in a PDF document**

# 6. Project Costing

This chapter deals with the costing of this project which gives an overall estimation of expenses that was required to complete this project. This Covers expenses of testing devices, Platform and Hardware cost, Human Resource Cost, and a grand total of the Entire cost.

## 6.1 **Project Cost Estimation**

The cost of the project is summarized in a tabular form displayed below:

**Table 4 Cost estimation table**

| Serial Number | Resources and Work |
|---|---|
| 1 | Hardware cost |
| 2 | Software Licensing |
| 3 | Cloud Infrastructure Cost |
| 4 | Man hours |

## 6.2 **Summary**

Since this project didn't have any physical model, hence no expenses were made for the physical model. However, effort on making the software via parallel learning of new technology raised the Human Resource cost.

# 7. Conclusions and Suggestions for Future Work

In this chapter, a conclusion has been given for the entire project along with a conclusion to each section present in the report. All are explained with the status of completion of each section mentioned clearly. This section ends with Suggestions and scope of future work which directs this project towards new openings of technology where the same project can be extended to meet the requirement of customers from time to time.

## 7.1 Conclusion

This project started with a Literature survey done via gathering information from different IEEE papers, patented documents, and reputed Websites. Also, popular applications were listed against theirfeatures making it clearer to compare and contrast applications with each other along with applications proposed in this project. Background Theory of all resources including technology and architectures worked upon and engines applied were extensively elaborated so that these theories can be applied effectively and the reason for their use/ application can be well understood. Later, all objectives were listed after declaring the title and Aim of the project, and methods and mythologies, to complete the bulleted objectives, were well tabulated. From objectives, Functional Requirements were extracted and were well segregated for sequential completion of theproject. Later, Sequence diagram was created giving a complete view of the project. Implementation of the project was displayed via displaying code written in C# and Aspose interacting with each other to create the application. Testing was done for all functionalities and was found to be working successfully. Later in the result section, all screenshots of applications in different states were taken to demonstrate the product of this project.

Each screenshot was explained with its importance as a view for the application. Performance analysis was done to give a numerical value to the performance of the application clearly stating the advancement in application proposed in this project compared to other applications present in the market. Later project cost estimation was done to know the financial asset required to rebuild this project. The entire project was concluded with s suitable conclusion and its scope in near future.

### 7.2 Suggestion for future work

Innovation is a never-ending process, hence bringing innovation and extension of this project is always possible. This project, for now, is limited to functionalities like creation and updation but it can be extended too.

## References

1. Aspose official documentation
   Set Privileges, Encrypt and Decrypt PDF|Aspose.PDF for .NET
2. Aspose-pdf/Aspose.PDF-for-.NET: Aspose.PDF for .NET examples, plugins and showcase projects
3. www.microservices.io
4. Conga-docgen resources