YCompany- Electronic Claims Processing

DAR Document



Nagarro Software Pvt. Ltd.

Pankaj Mahur

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| Version | Date | Author/Contributor | Comments |
| 1.0 | 29/12/2020 | Pankaj Mahur | First draft created |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Introduction 4](#_Toc62144930)

[1.1 Objective and scope of document 4](#_Toc62144931)

[2 Requirements at a Glance 5](#_Toc62144932)

[3 Available tools 6](#_Toc62144933)

[3.1 Application architecture 6](#_Toc62144934)

[3.1.1 Monolithic applications 6](#_Toc62144935)

[3.1.2 Microservices 6](#_Toc62144936)

[3.2 Frontend Frameworks 6](#_Toc62144937)

[3.2.1 React 6](#_Toc62144938)

[3.2.2 Vue.js 7](#_Toc62144939)

[3.2.3 Angular 7](#_Toc62144940)

[4 Comparison Analysis 8](#_Toc62144941)

[4.1 Point Matrix 9](#_Toc62144942)

[4.2 <Comparison 1> 9](#_Toc62144943)

[4.3 <Comparison 2> 9](#_Toc62144944)

[5 Recommendation 8](#_Toc62144945)

[6 Assumptions 9](#_Toc62144946)

[7 Risks 10](#_Toc62144947)

[8 Appendix 11](#_Toc62144948)

[8.1 References 11](#_Toc62144949)

# Introduction

This document has a detailed evaluation of popular and recommended tools and technologies that can be used in designing the Electronic Claim Processing System. The document has a detailed explanation of tools comparison, the pricing model for the various technologies available, risk, and assumptions too.

## Objective and scope of document

This document describes in detail the various tool & technology options available for the development of an Electronic Claim Processing System. It will begin with presenting all the options available in different areas of software development, bringing out a detailed analysis & comparison between different tools & frameworks around various factors. In the end, recommendations are presented.

# Requirements at a Glance

The eClaims system will essentially provide the following:

* A web application or Mobile App – this web-based application and mobile application will be available to authorized users such as customer case managers, auditors, reporting, etc.
  + Customer Portal
  + 3rd Party Provider Portal
  + Internal Portal
* Alert and notification system – this subsystem will be analysing the time to resolve any incident.
* Reporting
* Document Management

# Available tools

## Application architecture

### Monolithic applications

If all the functionalities of a project exists in a single codebase, then that application is known as monolithic application. We all must have designed a monolithic application in our lives in which we were given a problem statement and were asked to design a system with various functionalities. We design our application in various layers like presentation, service and persistence and then deploy that codebase as single jar/war file. This is nothing but a monolithic application where “mono” represents the single codebase containing all the required functionalities.

#### Features

* Simple to develop relative to microservices where skilled developers are required to identify and develop the services.
* Easier to deploy as only a single jar/war file is deployed.
* Relatively easier and simple to develop in comparison to microservices architecture.
* The problems of network latency and security are relatively less in comparison to microservices architecture.

Presentation Layer

Database

Service Layer

Data Layer

#### Pricing

NA

### Microservices

Microservices are designed to combat the problems associated with monoliths by going in the complete opposite direction. Microservices are an architectural style where an application is broken into a series of modules, each associated with a specific business objective.

Because microservices are relatively new, there is no universal definition that lays out what exactly a microservices architecture should look like.

However, all microservices share some commonalities:

* All services are independently deployable.
* They communicate using a network of lightweight technology-agnostic protocols.
* Services should be small in scope and focus on a single business goal.
* Each service is failure-resistant and fault-tolerant.

#### Features

* Software built as microservices can be broken down into multiple component services, so that each of these services can be deployed and then redeployed independently without compromising the integrity of an application. That means that microservice architecture gives developers the freedom to independently develop and deploy services.
* Better fault isolation: if one microservice fails, the others will continue to work.
* Code for different services can be written in different languages.
* Easy integration and automatic deployment; using open-source [continuous integration](https://apiumhub.com/tech-blog-barcelona/benefits-of-continuous-integration/) tools such as [Jenkins](https://apiumhub.com/tech-blog-barcelona/best-jenkins-plugins/), etc.
* The microservice architecture enables continuous delivery.
* Easy to understand since they represent a small piece of functionality, and easy to modify for developers, thus they can help a new team member become productive quickly.
* The code is organized around business capabilities.
* Scalability and reusability, as well as efficiency. Easy to scale and integrate with third-party services.
* Components can be spread across multiple servers or even multiple data centers.
* Work very well with containers, such as Docker.
* Complement cloud activities.
* Microservices simplify security monitoring because the various parts of an app are isolated. A security problem could happen in one section without affecting other areas of the project.
* Increase the autonomy of individual development teams within an organization, as ideas can be implemented and deployed without having to coordinate with a wider IT delivery function.

One simple way of understanding could be:

Presentation Layer

Microservice 3

Microservice 2

Microservice 1

Database

Data Layer

Service Layer

Service Layer

Data Layer

Database

Service Layer

Data Layer

Database

#### Pricing

NA

## Frontend Frameworks

T*aking into consideration the standpoint of the software development life cycle, the right choice among top front-end frameworks will be the fundamental step of your future success. The market has a wide diversity due to the vast set of challenges that architect, coders and developers solve every day. Thus, the number of new front-end frameworks considerably increases. More and more different frameworks appear, and in this article, we would like to discuss some of them.*

### React

Created as an open-source project and still utilized by Facebook, [React](https://reactjs.org/) is a popular JS framework that focuses on user experience. Unlike certain other frameworks, React is very portable. Regardless of what underlying technologies you are using, you will be able to take advantage of React.

#### Features

* Reusability of components makes it easy to collaborate and reuse them in other parts of the application.
* Consistent and seamless performance with the use of virtual DOM.
* Small learning curve
* It can be used for the development of both web and mobile apps.
* JSX - JavaScript Syntax Extension, By using JSX, we can write HTML structures in the same file that contains JavaScript code
* Due to its popularity, there is a huge amount of free help from peer developers available online.

#### Pricing

React is an open-source framework.

### Vue.js

Vue.js is the web framework for building user interfaces. It is an independent tool that creates web interfaces and does not require the additional extension. Vue.js was created by Evan You and initially released in February 2014. Probably you might not find the long list of big companies using Vue.js but it is popular among developers and the framework popularity considerably increases.

#### Features

* Tiny size
* Component Based Architecture (CBA),
* Virtual DOM rendering
* Reactive two-way data binding
* Easy to learn.

#### Pricing

Vue.js is an open-source framework.

### Angular

Google's flagship JS framework, Angular, has been in development for quite some time. While it is not the easiest framework to learn, the steep learning curve could end up being well worth the time.

It is great for projects requiring a revolving team because the way it encapsulates components makes it modular and easy for newer developers to understand. Developers taking advantage of Angular have the unparalleled ability to make apps look like Google's apps that use the same technology. This is accomplished through the usage of the Material Design framework.

You cannot have a list of the best front end development frameworks without Angular. Angular is the only framework that is based on TypeScript on this list.

#### Features

* TypeScript
* Dependency Injection
* Directives
* Efficient Two-Way Data Binding
* MVC Architecture
* Smart unit testing frameworks like Jasmine and Karma
* A vast community for learning and support.

#### Pricing

Angular is an open-source framework.

# Comparison Analysis

<Comparison brief>

## Point Matrix

<point assignment as per features>

|  |  |
| --- | --- |
| Feature | Points |
|  |  |
|  |  |
|  |  |
|  |  |

## <Comparison 1>

<comparison table>

|  |  |  |
| --- | --- | --- |
| Feature | <Tool 1> | <Tool 2> |
| A | **** | **** |
| B | **** | **** |
| C | **** | **** |

## <Comparison 2>

<comparison table>

|  |  |  |
| --- | --- | --- |
| Feature | <Tool 1> | <Tool 2> |
| A | **** | **** |
| B | **** | **** |
| C | **** | **** |

# Recommendation

<Your recommendation>

# Assumptions

<Assumption made during this analysis>



# Risks

<Detail out any risks involved>

# Appendix

## References

<Any reference>