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 6](#_Toc62729506)

[1.1 Objective and scope of document 6](#_Toc62729507)

[2 Requirements at a Glance 7](#_Toc62729508)

[3 Available tools 8](#_Toc62729509)

[3.1 Application architecture 8](#_Toc62729510)

[3.1.1 Monolithic applications 8](#_Toc62729511)

[3.1.2 Microservices 9](#_Toc62729512)

[3.2 Frontend Frameworks 11](#_Toc62729513)

[3.2.1 React 11](#_Toc62729514)

[3.2.2 Vue.js 12](#_Toc62729515)

[3.2.3 Angular 12](#_Toc62729516)

[3.3 Server-side programming technology 13](#_Toc62729517)

[3.4 Database server 13](#_Toc62729518)

[3.4.1 Microsoft SQL Server 13](#_Toc62729519)

[3.4.2 ElephantSQL 14](#_Toc62729520)

[3.4.3 Amazon Relational Database Service (RDS) 14](#_Toc62729521)

[3.5 ORM frameworks 15](#_Toc62729522)

[3.5.1 Entity Framework Core 15](#_Toc62729523)

[3.5.2 N-Hibernate. 15](#_Toc62729524)

[3.6 File server 16](#_Toc62729525)

[3.7 SignalR 16](#_Toc62729526)

[3.8 Backend service 16](#_Toc62729527)

[3.8.1 Windows Communication Foundation 16](#_Toc62729528)

[3.8.2 Web API 17](#_Toc62729529)

[3.9 Source code management 17](#_Toc62729530)

[3.9.1 Git 17](#_Toc62729531)

[3.9.2 Bitbucket 18](#_Toc62729532)

[3.10 Continuous Integration/Continuous Delivery and/or Continuous Deployment 18](#_Toc62729533)

[3.10.1 Jenkins 18](#_Toc62729534)

[3.10.2 Bamboo 19](#_Toc62729535)

[4 Comparison Analysis 19](#_Toc62729536)

[4.1 Monolithic vs Microservice Architecture 19](#_Toc62729537)

[4.1.1 Point Matrix 19](#_Toc62729538)

[4.1.2 Comparison Table 20](#_Toc62729539)

[4.2 Angular vs ReactJS vs Vue 20](#_Toc62729540)

[4.2.1 Point Matrix 20](#_Toc62729541)

[4.2.2 Comparison table 20](#_Toc62729542)

[4.3 Web API core vs WCF 21](#_Toc62729543)

[4.3.1 Point Matrix 21](#_Toc62729544)

[4.3.2 Comparison table 21](#_Toc62729545)

[4.4 Entity framework vs NHibernate 22](#_Toc62729546)

[4.4.1 Point Matrix 22](#_Toc62729547)

[4.4.2 Comparison table 22](#_Toc62729548)

[4.5 Bitbucket vs Git 22](#_Toc62729549)

[4.5.1 Point Matrix 22](#_Toc62729550)

[4.5.2 Comparison table 23](#_Toc62729551)

[4.6 Jenkins vs Bamboo 23](#_Toc62729552)

[4.6.1 Point Matrix 23](#_Toc62729553)

[4.6.2 Comparison table 23](#_Toc62729554)

[5 Recommendation 24](#_Toc62729555)

[5.1 Application architecture 24](#_Toc62729556)

[5.2 Docker 24](#_Toc62729557)

[5.3 Frontend framework 24](#_Toc62729558)

[5.4 Server-side programming technology 24](#_Toc62729559)

[5.5 Database server 24](#_Toc62729560)

[5.6 ORM framework 24](#_Toc62729561)

[5.7 Backend service 24](#_Toc62729562)

[5.8 Source code management 25](#_Toc62729563)

[5.9 CI/CD 25](#_Toc62729564)

[6 Assumptions 26](#_Toc62729565)

[7 Risks 27](#_Toc62729566)

[8 Appendix 28](#_Toc62729567)

[8.1 References 28](#_Toc62729568)

# 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.

## Server-side programming technology

Server-side programming can be explained as: It is the general name for the kind of program that runs directly on the server. Or we can say that server-side programming must deal with dynamic content. Server-side Uses.

* It processes the data from front end frameworks.
* Send back the requested data.
* Interaction with servers/storages
* Interaction with 3rd party application like email servers, short messaging server etc.
* Interaction with databases
* CRUD operation over the database

There are several languages that can be used for server-side programming:

* PHP
* NodeJS
* C# .NET
* Java

All above mentioned programming languages are well established and used throughout. We will be opting C# .NET due to technology working background and rich experience in this technology.

## Database server

Whether a small business or large corporation, digital transformation has enabled organizations to generate data at every touchpoint. From customer transaction data to marketing automation, organizations need to store data in an organized manner while ensuring its security.

### Microsoft SQL Server

Microsoft SQL Server is one of the most effective and sought-after database software. This relational DBMS server is perfect for storing and retrieving multiple requests at once.

#### Features

* Excellent Performance: Turns raw data into actionable insights that can be delivered on any platform or device.
* Improved Query Processing: It now leverages adaptive query processing features to enhance query performance in SQL database and SQL server.
* Easier to set up a brand-new database server from the start.
* Leverage Developer Network Feature to develop advanced queries.
* Handles complex queries and integrate with other programs.
* Advanced querying and easy management of distributed databases

#### Pricing

The “Express” and “Developer” versions are available free of cost. The paid plans start at $931 and range up to $14,256 (one-time cost).

### ElephantSQL

ElephantSQL is a PostgreSQL database hosting service. ElephantSQL will manage administrative tasks of PostgreSQL, such as installation, upgrades to latest stable version and backup handling.

ElephantSQL is also integrated to several cloud application platforms (also known as PaaS). With a click of a button your database is provisioned in the same data center as your application is hosted and is ready to be used immediately.

#### Features

* Fully Managed High Availability PostgreSQL.
* ElephantSQL is hosted by PostgreSQL experts, with several years of DBA PostgreSQL experience. Provides 24/7 support to thousands of customers.
* Automated backups are performed every day, which is stored in a cloud file storage so that they are always accessible. Also use point-in-time-recovery to restore your database.

#### Pricing

Pricing model varies from free to $2798 per month.

### Amazon Relational Database Service (RDS)

Amazon RDS is a stable and capable relational Database-as-a-Service (DBaaS) that is at par with other competent offerings from Amazon Web Services (AWS). It is a good option for experienced data scientists, users, and digital-native companies.

#### Features

* Data Backup & Security: Offers dedicated secure connection and backs up data automatically through its inbuilt feature.
* Flexible: It offers the option to scale computing resources or increase the storage capacity of the database through a single API call
* Scalable: You can scale up to more than the capacity of the single database deployment for read-heavy database workloads
* Easy to Use: Amazon RDS offers access to the abilities of a more familiar MySQL, Oracle, or Microsoft SQL Server database engine.

#### Pricing

Amazon RDS is a ‘free to try’ tool. Users can pay for only those resources that they use. You can leverage On-Demand or Reserved Instance to pay for its services. It might look overwhelming to pay for resources used, but you can easily estimate your monthly bill through its AWS Simple Monthly Calculator.

## ORM frameworks

Object-relational mapping (ORM, O/RM, and O/R mapping) in computer software is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. There are both free and commercial packages available that perform object-relational mapping, although some programmers opt to create their own ORM tools.

### Entity Framework Core

Entity Framework Core is an ORM made by Microsoft. It allows performing CRUD operations without having to write SQL queries. It supports Code First, Database First, Stored Procedure, Transaction, etc.

#### Features

* Support for LINQ, LINQ (Language Integrated Query) is uniform query syntax in C# and VB.NET to retrieve data from different sources and formats.
* Client-side Query Execution.
* Cross-platform
* Change Tracking
* Concurrency
* Migrations

#### Pricing

Entity Framework core is an open source ORM.

### N-Hibernate.

NHibernate is an actively developed, fully featured, open-source object-relational mapper for the .NET framework. It is used in thousands of successful projects. It is built on top of ADO.NET and the current version is NHibernate 4.0.4

#### Features

* Persist the object in the relation database and can do all CRUD operations in it.
* NHibernate does all the database work.
* Eliminates the necessity to write SQL statements or create a stored query.
* We can have all of our data access logic contained within our application.
* Data Cartography not only will our queries be effective, but they will also be validated by the compiler. Therefore, if our underlying table structure changes, the compiler will alert us that we need to change our queries!
* For all Create, Retrieve, Update, and Delete (CRUD) there is no need to write a SQL query.

#### Pricing

N-Hibernate is an open-source ORM framework.

## File server

A file server is a central server in a computer network that provides file systems or at least parts of a file system to connected clients. File servers therefore offer users a central storage place for files on internal data media, which is accessible to all authorized clients. Here, the server administrator defines strict rules regarding which users have which access rights: For instance, the configuration or file authorizations of the respective file system enable the admin to set which files can be seen and opened by a certain user or user group, and whether data can only be viewed or also added, edited, or deleted.

Application will store the uploaded files for future use and there are following ways to store the file.

1. Store File locally on hosted server or separate file server
2. Store File in Cloud storage

## SignalR

SignalR is a free and open-source software library for Microsoft ASP.NET that allows server code to send asynchronous notifications to client-side web applications. The library includes server-side and client-side JavaScript components. SignalR is an open-source software library.

## Backend service

A backend service can be of any type. It can be a data service, web service, .NET service, security service, or using a third-party API. Or even extending a feature with your own code. Thus, you can consider external data as a service. Based on the underlying data, you can categorize backend services as Web Services, Database Services, .NET Services and Security Services.

### Windows Communication Foundation

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data.

#### Feature

* Service Orientation
* Interoperability
* Multiple Message Patterns
* Reliable and Queued Messages
* AJAX and REST Support

#### Pricing

Microsoft visual studio developer license required as it is shipped with .NET framework.

### Web API

The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc. It works more or less the same way as MVC web application except that it sends data as a response instead of html view. It is like a webservice or WCF service, but the exception is that it only supports HTTP protocol.

#### Feature

* Web API is an ideal platform for building RESTful services.
* Web API maps HTTP verbs to method names.
* Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
* Web API framework includes HTTP Client to communicate with Web API server.

#### Pricing

We will target .NET Core web API and will require Microsoft visual studio developer license as it is shipped with .NET framework.

## Source code management

Source code management (SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors. SCM is also synonymous with Version control.

As software projects grow in lines of code and contributor head count, the costs of communication overhead and management complexity also grow. SCM is a critical tool to alleviate the organizational strain of growing development costs. Some of the well-known source code management tools are as follows-

* TFS
* SVN
* GIT
* BITBUCKET

We will discuss some open source SCM tools.

### Git

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning-fast performance.

#### Feature

* Tracks history
* Free and open source
* Supports non-linear development.
* Creates backups.
* Scalable
* Supports collaboration.
* Branching is easier.
* Distributed development

### Bitbucket

Bitbucket is a Git repository management solution designed for professional teams. It gives you a central place to manage git repositories, collaborate on your source code and guide you through the development flow.

#### Feature

* Access control to restrict access to your source code.
* Workflow control to enforce a project or team workflow.
* Pull requests with in-line commenting for collaboration on code review.
* Jira integration for full development traceability.
* Full Rest API to build features custom to your workflow if they are not already available from our Marketplace.

#### Pricing

3 pricing models – Free, Standard, Premium

## Continuous Integration/Continuous Delivery and/or Continuous Deployment

CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment. CI/CD is a solution to the problems integrating new code can cause for development and operations teams (AKA "integration hell").

Specifically, CI/CD introduces ongoing automation and continuous monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment. Taken together, these connected practices are often referred to as a "CI/CD pipeline" and are supported by development and operations teams working together in an agile way with DevOps.

### Jenkins

Jenkins is an open-source automation server in which the central build and continuous integration process take place. It is a self-contained Java-based program with packages for Windows, macOS, and other Unix-like operating systems. With hundreds of plugins available, Jenkins supports building, deploying, and automating for software development projects.

#### Feature

* Easy installation and upgrade on various OSs
* Simple and user-friendly interface
* Extensible with huge community-contributed plugin resource
* Easy environment configuration in the user interface
* Supports distributed builds with master-slave architecture.
* Build schedules based on expressions.
* Supports shells and Windows command execution in pre-build steps.
* Supports notification on the build status.

#### Pricing

Jenkins is an open-source tool.

### Bamboo

Bamboo is a continuous integration server that automates the management of software application releases, thus creating a continuous delivery pipeline. Bamboo covers building and functional testing, assigning versions, tagging releases, deploying, and activating new versions on production.

#### Feature

* Supports up to 100 remote build agents.
* Run batches of tests in parallel and get feedback quickly.
* Creates images and pushes into a registry.
* Per-environment permissions that allow developers and testers to deploy to their environments on-demand while the production stays locked down
* Detects new branches in Git, Mercurial, SVN Repos and applies the main line’s CI scheme to them automatically.
* Triggers build based on the changes detected in the repository. Pushes notifications from Bitbucket, a set schedule, the completion of another build or any combination thereof.

#### Pricing

Bamboo pricing tiers are based on agents or “build slaves” rather than users. The more agents, the more processes it can run concurrently – either in the same build or different builds.

# Comparison Analysis

## Monolithic vs Microservice Architecture

### Point Matrix

|  |  |
| --- | --- |
| Feature | Points |
| Complexity | -5 |
| System distribution | -5 |
| CI/CD | 5 |
| Testing | 5 |
| Reliability | 5 |
| Simplicity | 5 |
| Expertise | -5 |
| Scalability | 5 |
| Codebase | 5 |
| Technology Adoption | 5 |

### Comparison Table

|  |  |  |
| --- | --- | --- |
| Feature | Monolithic | Microservices |
| Complexity | **** | **** |
| System distribution | **** | **** |
| CI/CD | **** | **** |
| Testing | **** | **** |
| Reliability | **** | **** |
| Simplicity | **** | **** |
| Expertise | **** | **** |
| Scalability | **** | **** |
| Codebase | **** | **** |
| Technology Adoption | **** | **** |

## Angular vs ReactJS vs Vue

### Point Matrix

|  |  |
| --- | --- |
| Feature | Points |
| UI/DOM Manipulation | 5 |
| State Management | 5 |
| Two-way Data Binding | 5 |
| Routing | 5 |
| Form Validation & Handling | 5 |
| Http Client | 5 |
| Knowledge sharing and support | 5 |

### Comparison table

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Angular | ReactJS | Vue |
| UI/DOM Manipulation | **** | **** | **** |
| State Management | **** | **** | **** |
| Two-way Data Binding | **** | **** | **** |
| Routing | **** | **** | **** |
| Form Validation & Handling | **** | **** | **** |
| Http Client | **** | **** | **** |
| Knowledge sharing and support | **** | **** | **** |

## Web API core vs WCF

### Point Matrix

|  |  |
| --- | --- |
| Feature | Points |
| Cross-Platform Features | 5 |
| Lightweight | 5 |
| Self-Hosting | 5 |
| Action Filters | 5 |
| Rich-client web applications | 5 |
| Open Source | 5 |
| Protocol support | 5 |
| UTF support | 5 |
| Configurable transports | 5 |

### Comparison table

|  |  |  |
| --- | --- | --- |
| Feature | Web API core | WCF |
| Cross-Platform Features | **** | **** |
| Lightweight | **** | **** |
| Self-Hosting | **** | **** |
| Action Filters | **** | **** |
| Rich-client web applications | **** | **** |
| Open Source | **** | **** |
| Protocol support | **** | **** |
| UTF support | **** | **** |
| Configurable transports | **** | **** |

## Entity framework vs NHibernate

### Point Matrix

|  |  |
| --- | --- |
| Feature | Points |
| Open Source | 5 |
| Custom Eager Loading | 5 |
| LINQ Support | 5 |
| Code first Mapping | 5 |
| Wider database provider support | 5 |
| Seeding of Initial database | 5 |
| Support of Asynchronous Operations | 5 |
| Connection Resiliency |  |

### Comparison table

|  |  |  |
| --- | --- | --- |
| Feature | Entity Framework 6 | Nhibernate 4 |
| Open Source | **** | **** |
| Custom Eager Loading | **** | **** |
| Linq Support | **** | **** |
| Code first Mapping | **** | **** |
| Wider database provider support | **** | **** |
| Seeding of Initial database | **** | **** |
| Support of Asynchoronours Operations | **** | **** |
| Connection Resiliency | **** | **** |

## Bitbucket vs Git

### Point Matrix

|  |  |
| --- | --- |
| Feature | Points |
| Navigation | 5 |
| Project Analysis | 5 |
| Flexible | 5 |
| Pricing | 5 |
| Community | 5 |
| JIRA integration | 5 |
| Integrated CI/CD | 5 |

### Comparison table

|  |  |  |
| --- | --- | --- |
| Feature | Bitbucket | Git |
| Navigation | **** | **** |
| Project Analysis | **** | **** |
| Flexible | **** | **** |
| Pricing | **** | **** |
| Community | **** | **** |
| JIRA integration | **** | **** |
| Integrated CI/CD | **** | **** |

## Jenkins vs Bamboo

### Point Matrix

|  |  |
| --- | --- |
| Feature | Points |
| Open source | 5 |
| Plugin Support | 5 |
| Automatic Platform | 5 |
| JIRA Support | 5 |
| Ease of usage | 5 |
| Repository Support | 5 |
| Wide Online Support | 5 |
| Handle multiple languages | 5 |
| Documentation | 5 |
| Parallel build in multiple branches | 5 |

### Comparison table

|  |  |  |
| --- | --- | --- |
| Feature | Jenkins | Bamboo |
| Open source | **** | **** |
| Plugin Support | **** | **** But cost associated |
| Automatic Platform | **** | **** |
| JIRA Support | **** | **** |
| Ease of usage | **** | **** |
| Repository Support | **** | **** |
| Wide Online Support | **** | **** |
| Handle multiple languages | **** | **** |
| Documentation | **** | **** |
| Parallel build in multiple branches | **** | **** |

### 

# Recommendation

## Application architecture

**Microservices based**- Microservices is preferred architecture for the application for achieving all nonfunctional requirements.

## Docker

**Docker** for packaging applications in “containers,” so they are portable for any system running the Linux operating system (OS) or Windows OS.

## Frontend framework

**Angular** framework makes it easy to develop web applications. Combining dependency injection, declarative templates, two-way binding, end-to-end tooling, and integrated best practices, it solves almost all the challenges when creating a web app.

## Server-side programming technology

**ASP.Net Core** to build enterprise web applications on, mainly because of its flexibility that comes from it being cross-platform. It starts all the way from the tooling available to be able to develop ASP.Net Core applications using Visual Studio, or Visual Studio Code on either Windows or Mac operating systems, even on Linux.

ASP.Net Core has such a vibrant community that it is always allowed to give their input. The fact that it is open source actually paves way for faster improvements and applicability across industries. Apart from the development environment, when ASP.Net Core applications are ready to be deployed into production, you can do so internally in your organization, or just about any other worthwhile cloud hosting service provider including Azure and AWS.

## Database server

**.**NET professionals are often familiar with related programming languages like C# and VB.Net, so using **Microsoft SQL** can reduce staff training expenditure and personnel changes.

## ORM framework

**Entity framework core** - .NET ecosystem contains a few actively maintained ORMs, ability to generate schemas or get you out of writing SQL, EF supports all of this and can mean that you do not need to write a single bit of SQL in your application. The queries that EF generates are very good and even quite readable, if you do need to drop to SQL to debug. When you need to get an application off the ground quickly, EF provides a low-friction path for data access.

## Backend service

**ASP .NET Core API**- using a web API framework, one can easily create services that can run on various entities. Hence, web API makes it easier for developers to build an ASP.NET application that is compatible with any browser and almost any device. With Web API, you get access to entire features of HTTP like URIs, request/response headers, content formatting, caching, etc., and hence, as compared to WCF rest services (which require defining extra config settings for different devices), it is much easier to develop ASP.NET web applications using RESTful web services through Web APIs.

## Source code management

**Bitbucket**- Bitbucket based on git and provide some cool feature like Pull Request, Code Review Comments, easy to merge. Easy to integrate with Jenkins by using web hooks.

## CI/CD

**Jenkins**- Jenkins is an open-source serve, it gives you a common way to monitor a CI/CD pipeline (with many users watching the same important environment). Nowadays Jenkins is the core for many projects that needed end to end automated application deployments. In short, it is being used as an orchestrator for deploying applications.

# Assumptions

* Container-based microservices applications and deployment.
* Cloud SQL for SQL Server.
* Development team will be of .NET background and will work on ASP .NET Core.
* Team is expected to be aware of SQL server concepts.
* Team is expected to work on mentioned tools with minimal training if any tool/tools if required.
* Expecting deployment and hosting will be done on Microsoft Azure cloud service, Azure Kubernetes service.

# Risks

Resourcing for the tool, framework selected is important and if not done then things need to be revisited to accommodate the changes.

# Appendix

## References

1. http://techbrij.com/database-change-notifications-asp-net-signalr-sqldependency
2. https://www.aspsnippets.com/Articles/Automated-Email-Notifications-using-SQL-Server-Job-Schedular.aspx
3. <https://msdn.microsoft.com/en-us/library/ms178359(v=vs.110).aspx#orm>
4. <https://www.devbridge.com/articles/entity-framework-6-vs-nhibernate-4/>
5. https://xhalent.wordpress.com/2011/02/01/nhibernate-vs-entityframework-experience-from-the-real-world/#comment-158
6. <https://www.libertycenterone.com/blog/oracle-vs-mysql-vs-sql-server-vs-postgresql-which-dbms-is-the-best-choice-for-you/>
7. https://codeship.com/continuous-integration-essentials