HealthStats Enhancement Architecture Review

Prepared for

Ministry of Health

Design

8 January 2019 | Version 2.0

DOCUMENT CONTROL

Authors

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Title | Date |
| **Prepared:** | Michael Youssef | Technology Consultant | 8/1/2019 |

RELEASE INFORMATION

|  |  |  |  |
| --- | --- | --- | --- |
| Name | [Status] HealthStats Enhancement Architecture Review | | |
| Version | Date Released | Pages Affected | Remarks |
| 0.1 | 8/1/2019 | All | Creation of Document for internal review |
| 1.0 | 10/1/2018 | All | Review and ready for release |

AUTHORISED RELEASE

|  |  |  |  |
| --- | --- | --- | --- |
| Authorised | Position | Signature | Date |

CONTACT DETAILS

|  |  |  |  |
| --- | --- | --- | --- |
| Technology Consultant, MOQdigital | | Practice Manager, MOQdigital | |
| Michael Youssef | | Piers Matthews | |
| **Telephone:** | + 614 15885554 | **Telephone:** | +614 16701432 |
| **Email:** | [myoussef@moqdigital.com.au](mailto:myoussef@moqdigital.com.au) | **Email:** | p[mtthews@moqdigital.com.au](mailto:mtthews@moqdigital.com.au) |

Definitions

| **Term** | **Definition** |
| --- | --- |
| CI/CD | Continuous integration/ continuous delivery |
| EBU | Epidemiology and Biostatistics Unit |
| LOB | Line of business |
| MOQ | MOQdigital |
| MOH | Ministry of Health |
| VSTS | Visual Studio Team Services |
| NFR | Non-functional Requirement |
| BAU | Business as Usual |
| DMZ | Demilitarised Zone |
| PaaS | Platform as a Service |
| SPA | Single Page Application |

Table of contents

[1. Introduction 6](#_Toc536193645)

[1.1. Purpose of document 6](#_Toc536193646)

[1.2. Background 6](#_Toc536193647)

[1.3. Scope 6](#_Toc536193648)

[1.4. ReferenceS 6](#_Toc536193649)

[1.5. decisions 6](#_Toc536193650)

[2. Requirements model 7](#_Toc536193651)

[2.1. Architecture objectives 7](#_Toc536193652)

[2.2. Solution design drivers 7](#_Toc536193653)

[2.3. Non-functional requirements 8](#_Toc536193654)

[2.3.1. Authentication and Authorisation Management 8](#_Toc536193655)

[2.3.1.1. Authentication Options 8](#_Toc536193656)

[2.3.2. Performance 8](#_Toc536193657)

[2.3.3. Browser Compatibility 8](#_Toc536193658)

[2.3.4. Availability 9](#_Toc536193659)

[2.3.5. Backup and Recovery 9](#_Toc536193660)

[2.4. Assumptions 9](#_Toc536193661)

[2.5. Constraints 9](#_Toc536193662)

[3. Technical application context 11](#_Toc536193663)

[3.1. solution components 11](#_Toc536193664)

[3.2. actors 14](#_Toc536193665)

[3.3. external systems 15](#_Toc536193666)

[3.3.1. Internal application(s) integration (SAPHaRI) 15](#_Toc536193667)

[3.3.2. External application(s) integration (PoolParty) 15](#_Toc536193668)

[4. Technical architecture 16](#_Toc536193669)

[4.1. technology stack 16](#_Toc536193670)

[4.2. user authorisation 16](#_Toc536193671)

[4.3. RESTfull api’s 17](#_Toc536193672)

[4.4. failover protection 17](#_Toc536193673)

[5. architecture 18](#_Toc536193674)

[5.1. infrastructure 18](#_Toc536193675)

[5.2. Data architecture 18](#_Toc536193676)

[5.3. solution architecture 19](#_Toc536193677)

[6. Architectural RevIew 20](#_Toc536193678)

[6.1. High Level Architecture 20](#_Toc536193679)

[6.2. Cloud TransfoRmation 20](#_Toc536193680)

[6.3. Components Review 22](#_Toc536193681)

[6.3.1. Business Logic and Server Processing 22](#_Toc536193682)

[6.3.2. User Experience 23](#_Toc536193683)

[6.3.2.1. User Interface 23](#_Toc536193684)

[6.3.2.2. Graphing Visualisation: 23](#_Toc536193685)

[6.3.2.3. Printing Output: 24](#_Toc536193686)

[6.3.3. Data Management & Processing 24](#_Toc536193687)

[6.3.3.1. Data Storage Platform: 24](#_Toc536193688)

[6.3.3.2. Data Transformation 24](#_Toc536193689)

[6.3.3.3. Versioning 24](#_Toc536193690)

[6.3.4. Content Management 25](#_Toc536193691)

[6.3.5. Workflow Management 25](#_Toc536193692)

[6.3.6. Logging 25](#_Toc536193693)

[6.4. Environment & CI/CD 26](#_Toc536193694)

[6.5. sUMMARY 26](#_Toc536193695)

[6.5.1. APIS 26](#_Toc536193696)

[6.5.2. Security 27](#_Toc536193697)

[7. Project Execution Plan 28](#_Toc536193698)

[7.1. Agile / Recommended Approach 28](#_Toc536193699)

[7.2. Team Model 28](#_Toc536193700)

[8. Development Tools and processes 29](#_Toc536193701)

[8.1. system development 29](#_Toc536193702)

[8.2. source control 29](#_Toc536193703)

[8.3. continuous Integration / continuous delivery 30](#_Toc536193704)

[8.4. Environments 31](#_Toc536193705)

[8.5. deployment 31](#_Toc536193706)

[9. disaster recovery 32](#_Toc536193707)

[9.1. backup process 32](#_Toc536193708)

[9.2. recovery process 32](#_Toc536193709)

[APPENDIX A. 33](#_Toc536193710)

# Introduction

## Purpose of document

The purpose of this document is to review and evaluate the proposed architectures and detail the suitable solution for HealthStats Major Enhancements program.

Some sections of this document are derived from the baseline architecture document.

This document provides the framework of system components that will be used to build a solution. The architecture does not provide functionality but does provide the tools and techniques that will be used to deliver functionality. This also enables the provision of non-functional requirements for a solution such as user identification and access management, security, data backup, system accessibility, responsiveness etc.

## Background

This document covers the architectural review’s findings in terms of changes, requirements mapping and technical risks.

The following sections will describe in detail each architectural component, our understanding of it and recommendation.

Also, MOQ has added a project execution plan including the following:

* Recommended Approach
* Team

## Scope

This document describes the recommendations based on the baseline architecture for the new HealthStats system, confirms the technical strategy selected, proposes appropriate products, describe code management processes including the Continuous Integration / Continuous Deployment (CI/CD) pipeline. It also covers the basics of the solution’s backup and recovery processes.

## ReferenceS

MOQ has been requested to provide Ministry of Health with professional IT services to assist in providing architecture review for Health stats web application.

MOQ’s resources had several meetings with stake holders to get a clear understanding of the requirements, made a deep dive into the proposed solution architecture to validate it against the requirements.

## decisions

[See Components review section](#_Components_Review)

# Requirements model

## Architecture objectives

The Architecture described in this document must satisfy the following objectives:

1. It must support the **Solution Components** as specified in the [Technical Application Context](#_Technical_application_context).
2. It must help to satisfy the **Design Drivers**. (See Solution Design Drivers)
3. It must support the implied non-functional requirements (**NFRs**) on the new HealthStats system. E.g. Security and availability. (See Non-Functional Requirements.)
4. It must support an **API Driven Composite Application** by being extensible and allowing the integration of as-yet unknown packages into the overall HealthStats NSW solution. One such candidate package currently being considered is *PoolParty*.
5. It should provide a User Interface that is **Responsive** to the size and format of the user’s screen. Responsive applications will allow the layout of the content to be modified depending on the screen type.
6. It should specify **Skills** required to support the application that are available to Health NSW – preferably as in-house support staff.
7. It should be a system that eHealth can and will **support**.
8. It should only utilise **Strategic Products** that can be expected to be supported for at least 15 years.
9. It must provide a **Sound Investment**.
10. It should use an approach that enables integration with internal and external systems such as SAPHaRI and PoolParty.
11. It should allow **re-usability** so that components built for HealthStats can be leveraged by other applications.
12. It should be fault-tolerant, horizontally scalable, extendable to public APIs when required.
13. It should be platform agnostic e.g. can be lifted and shipped to Cloud when required.

## Solution design drivers

The design drivers for the new HealthStats system will have an impact on both the NFRs and the classes of components that will need to be developed. They are listed here to provide context for the requirements listed under those headings below.

| # | Design Driver | Description |
| --- | --- | --- |
| 1 | Improve the “**Look and Feel**” | To enhance navigation and visualisation of the website and statistical information. |
| 2 | **Improve Search** | To improve user ability to find HealthStats content in an effective and engaging manner. |
| 3 | **Data Timeliness** | To allow data to flow through the system as fast as possible but following all quality assurance processes and keeping privacy principles. |
| 4 | **System** **Maintenance Effort** | To reduce the time required from HealthStats team to maintain the system. |
| 5 | **Metadata availability** | To display most comprehensive view of the data meaning and its constraints to users. |
| 6 | **Scalability and Extensibility** | To consume resources effectively and be opened for various extensions. |

## Non-functional requirements

### Authentication and Authorisation Management

HealthStats develops and publishes articles for public consumption but these articles must be securely protected from public access until they have been approved for release. Authorized users should be able to create new content and to compile, review and approve unreleased articles.

HealthStats statistical content is in some cases developed from unit level medical records. The system must ensure that this information is not available to any users except those specifically authorised. Unreleased statistics may potentially be re-identifiable, and the system must ensure that all articles and content are not available to any users other than those who have been specifically been granted access.

A number of HealthStats roles have been identified in the conceptual solution (e.g. Manager HealthStats NSW, Release Approver, EBU Team Member) and each one of these roles requires certain system privileges. The system will need to support the granting (and revoking) of these roles from individuals and internal security groups. It is noted that granting the public with access to content and articles requires formal release by CEE executive (i.e. someone granted the Release Approver role).

#### Authentication Options

Internal users will be authenticated when they logon to the Health network. External users will generally be anonymous, but it is possible for external users to be identified and authenticated by an external authority. It is anticipated that external users will not be given system privileges beyond those given to anonymous users. All built-in Authorisation components will use ACL based (Access Control List) approach.

### Performance

As an aspiration pages should generally be fully rendered within 5 seconds of a request being sent by a user, but this metric is dependent on user hardware and network and so the technical architecture described here cannot be held responsible. An alternative metric for which the technical architecture can be held responsible is that the Application Gateway servers should provide a response to incoming requests within an average time of two seconds from receiving the request. This metric should apply to all requests regardless of which components are required to service them.

### Browser Compatibility

|  |  |
| --- | --- |
| **Browser** | **Minimum Version** |
| Chrome |  |
| Internet Explorer |  |
| Microsoft Edge |  |
| Firefox |  |
| Safari |  |

### Availability

NSW Health Staff and most other users will access the system during NSW business hours. Some users (including international users) may access the system any time night or day but generally these will not be priority users.

NSW Health Staff will access the system from workstations that are logged onto the NSW Health domain. These are considered *internal* users. Other users will access the system without being logged on to the NSW Health domain and are considered *External* users.

Availability during NSW business hours is a high priority for internal users and should be held above 95%. After hours availability should be maximised for the convenience of external users.

### Backup and Recovery

The face of HealthStats NSW is that of a reporting system, but it also manages the business of preparing the HealthStats statistics and for that reason is also an operational system. With that consideration in mind Backup and Recovery parameters should be consistent with a non-critical operational system.

There are no known safety risks that will arise if the system is not available for a period of two continuous weeks but for operational considerations any period of continuous downtime should be kept to a minimum.

Point-in-time recovery of operational components of the system is preferred so that no work of the HealthStats team will be lost in case of system failure. The most recent reports in the system should always be re-buildable and so recovery to the previous night on these components should be satisfactory.

## Assumptions

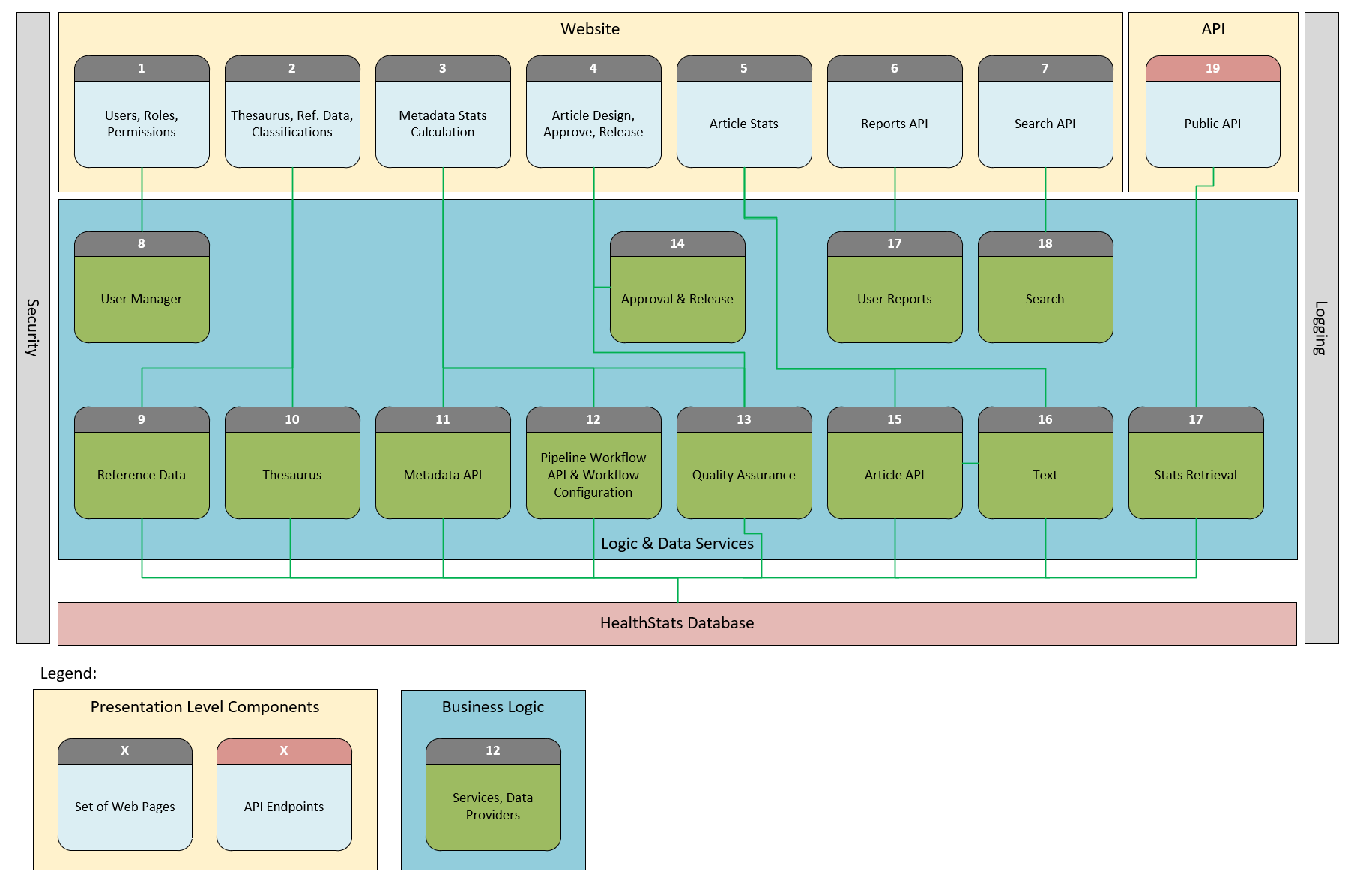
1. The system will be supported when in Business as Usual (BAU) mode with the following support structure:
   1. Level 1: The EBU team
   2. Level 2: Technical staff within PHIS and the State-wide Service Desk
   3. Level 3: eHealth technical staff
2. Policy will be in place that requires all potentially re-identifiable data to be hosted on internal servers. Note that this policy is expected to be challenged during the life-time of the new HealthStats system if not before it initially goes live. (*Note: The policy of NSW Health data needing to be stored on premises is being reconsidered within the eHealth policy areas – specifically Investment, Strategy and Architecture*).

## Constraints

The HealthStats technical architecture should be consistent with eHealth NSW architecture and the technical support capabilities they can provide. Currently this implies a Microsoft solution deployed internally, although it is expected that exceptions could be approved in special circumstances.

# Technical application context

## solution components



Note: *The webpages shown on the diagram are the “Website Layer”. The components shown in the Logic & Data Services layer (the light blue background) is where the APIs are provided. Applications that use the APIs will connect to these Logic & Data Services components directly*.

| # | Component | Description |
| --- | --- | --- |
| 1 | Users, Roles, Permissions | A set of web pages to map policies to roles and assign roles to users and/or user groups. The functionality is provided by User Manager. |
| 2 | Thesaurus, Reference Data Classifications | A set of web pages to display and manage terms such as classification schema items (i.e. “Male”, “Female”, “Adolescent”, “Heart Disease”, etc.) and the relationships between those terms |
| 3 | Metadata, Stats Calculation | A set of web pages to manage metadata, see the new data sets to be processed, configure and execute the stats recalculation job, analyse the results using QA reports and release recalculated stats. |
| 4 | Article Design, Approve, Release | A set of web pages to create, modify, review and approve the dynamic website content. Articles may contain stats diagrams, images, videos or text paragraphs. |
| 5 | Articles, Stats | Dynamic web page that can find and present the approved article, based on the URL path. Dynamic controls will be added to the page to redefine the stats presentation. |
| 6 | Custom User Reports | A set of web pages to manage the configuration and display the statistics. |
| 7 | Search, Static Pages | A set of static website pages, including the generic search control. Search logic will be provided by Search Manager. |
| 8 | User Manager | A middle-layer component to encapsulate the logic about managing lists of users, user groups, roles and policies. Will also contain a Policy Manager that will be used by security layer to provide authorization policies to the web site and API. |
| 9 | Reference Data Manager | A middle-layer component to provide create, read, update, delete (CRUD), methods for all stats classification schemas. |
| 10 | Thesaurus Manager | A middle-layer component to provide CRUD methods for the thesaurus management. Will also include business logic to automatically propose tags for the article created. |
| 11 | Metadata Manager | A middle-layer component to manage the data about what statistics and how should it be generated. |
| 12 | Stats Pipeline Manager | A middle-layer component which is responsible for stats recalculation process. It has access to Metadata Manager and data set sources to collect all data required and present a set of indicators and raw data to the Analyst, so he can analyse the changes and approve the recalculation process. The detailed design is described in Conceptual Solution, section 5 (Extract and Analyse). |
| 13 | Quality Assurance Manager | A middle-layer component to manage the QA tests configuration and QA tests. Three levels of QA tests are defined in system   * Data Source QA * Case Selection QA * Indicator Statistics QA |
| 14 | Approval and Release Manager | A middle-layer component to register all completed QA reports and metadata configurations as approved, push the approved statistics into the Stats Mart, initialize generation of new default articles for new indicators, initialize recompilation process for all article snapshots for the impacted stats and starts the process that requests their release. |
| 15 | Article Manager | A middle-layer component to manage the article lifecycle process. Can automatically create a new stats page and new release candidate and provides methods for managing article model and content. Please see Conceptual Solution, section 6 (Compile Articles) for details. |
| 16 | Text Manager | Some articles may share a dynamic text. This middle-layer component is responsible for the text CRUD operations and indexing for search. |
| 17 | Stats Retrieval Manager | A middle-layer component to select the statistics set, using the parameters provided. |
| 18 | User Reports Manager | A middle-layer component to manage user-created report configurations. |
| 19 | Search Manager | A middle-layer component responsible for taking a set of search terms, applying complex logic to determine the list of matching articles and ranking those articles in order of expected relevance. |
| 20 | Stats | A set of API endpoints to return a statistic set by parameters provided to external 3rd party systems. All API endpoints will consume the same authorization process as the main website. |

The features of this application are:

1. Internal and external users will access the system via a web browser. The web browser may be small or large format. Requests from external users will pass through a proxy in the DMZ before being forwarded to the HealthStats application or HealthStats API which will both be hosted behind the internal firewall.
2. Internal users are authenticated when they log on to the network. They will be recognised as internal users when they access the application and will be afforded system privileges dependent on the security groups to which they belong.
3. External users may be anonymous or may be identified and authenticated by an external authentication service (such as Service NSW).
4. Requests to the HealthStats app from external users will initially be submitted to the User Manager component where any claims owned by external users will be stored. They will be retrieved with each request and attached to the request.
5. A Policy Manager component will be used to assess requests against the claims each user (both internal and external) provides. Internal users with special privileges (most notably the EBU team) will be provided access to secure components while non-privileged users (including external users) will not be permitted to access those components at all.
6. Requests to retrieve HealthStats articles will be forwarded to the Article Manager for compilation of content. The Article Manager will create the article on-the-fly based on the request, the stored layout for the requested article and the user claims attached to the request. The Article Manager component will return the article to the HealthStats application which will pass it back to the user’s browser. If the article contains one or more statistical presentations the application will request the required data from the StatsRetrieval service and the data and the article will be returned to the User’s web browser together.
7. If the article contains a Statistical Presentation a script will use the supplied data to render the presentation on the screen. User controls associated with a Statistical Presentation may allow the user to modify the presentation using the data as supplied. Other user controls associated with the Statistical Presentation may also allow the user to drill-into or roll-up from the currently supplied data. E.g. if the data is available in the Statistics Mart the user may have a control allowing them to view the current statistics being displayed by perhaps ethnicity and/or gender. If the user selects one of these the script will make another call to the StatsRetrieval service to request the specified data and then render that data on the screen.
8. The StatsRetrieval service and the search service will be available via an API so they can be called from scripts and external applications as required. Claims will be associated with each request using the same mechanisms described previously to ensure that only resources for which the user is authorised are provided.
9. The Metadata Manager component will be used to specify what statistics are generated by the system and how they are generated. Included in this definition is metadata pertaining to the source data set, the indicator definitions, the statistical methods being executed and any constraints/filters being applied to the data.
10. The Statistics Pipeline Manager will use information from the Metadata Manager (including references to scripts provided by the EBU team) to build a calculation pipeline that will execute and return the next generation of the impacted statistics. It will be designed as a standalone stateful service to encapsulate the important functionality, distribute the resource load and persist the process result.

Once an Analyst user is comfortable with metadata updates, he activates the Statistics Pipeline Manager process. Its main responsibility is to collect all metadata associated with the current job, identify and configure all calculation stages required, execute the calculation process and store the result. The result will include a QA report as well. However, the calculated statistics will not be added to the Statistics Mart until the QA report will be approved by a Manager user.

1. As part of the pipeline the candidate statistics will be forwarded to the Quality Assurance Manager where they will be compared to previous versions of the same statistics so as to make an automatic determination as to whether the new statistics are likely right. The Quality Assurance Manager will allow an EBU user to comment on the new statistics and possibly request the Manager HealthStats NSW provide approval.
2. The Text Manager component will be used to specify text blocks that appear on Articles. The same Text blocks may appear on multiple articles. A similar component will be used to manage media assets such as pictures, audio and video.
3. The Reference Data Manager will be used to specify classification schemes such as Age Groups, Gender codes, Countries of birth etc.
4. The HealthStats database will keep a copy of all approved statistics and these will be available for including in articles presented to suitably privileged users. The database will also keep a copy of approved Article Layouts, Metadata, Reference Data, Text and Media Assets to enable Search.

## actors

The system will support the following actors:

| # | Actor | Description | Location |
| --- | --- | --- | --- |
| 1 | EBU Staff | Responsible for configuration and calculation areas. | Internal |
| 2 | Anonymous Users | Can browse the public area of the website and create the reports using the enhanced functionality. | External |
| 3 | Identified End Users | Can browse the public area of the website and create reports using the enhanced functionality. Can save reports for later retrieval. | Mainly external but can be internal |
| 4 | Manager HealthStats NSW | Coordinates content development and release | Internal |
| 5 | CEE Executive | Approves content for public release | Internal |
| 6 | Policy Area Staff | Reviews content and provides policy area approval (or rejection) | Internal although may be external in the future |

## external systems

The system must support an interface to SAPHaRI and Thesaurus and Search components.

It is expected that in the future it will also support an interface that will provide access to the NSW Open Data Portal.

### Internal application(s) integration (SAPHaRI)

To be updated later

### External application(s) integration (PoolParty)

To be updated later

# Technical architecture

This section presents the proposed architecture for the HealthStats application

## technology stack

* Asp.net Core 2.x Framework
* Entity Framework
* Angular 6 or higher
* Angular Material/Bootstrap
* Microsoft SQL Server
* Plot.ly
* F5 Load Balancer
* Solr

## user authorisation

The following Approach will be used for Authentication and Authorisation:

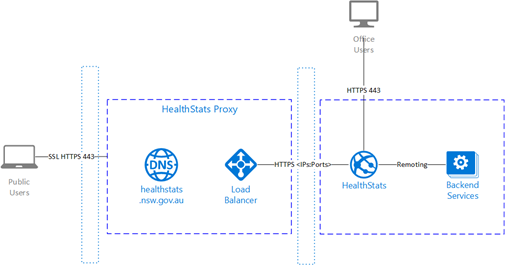
* Window Authentication for Internal users
* Policy-based authorisation
* Using JWT for public user claims.

The ASP.NET framework allows authorisation requirements to be declared as pre-compilation instructions. These instructions enforce authorisation rules that can be configured based on information associated with the user such as whether they are internal or external and what security group memberships they possess. This feature can be utilised using a [policy-based authorization](https://docs.microsoft.com/en-us/aspnet/core/security/authorization/policies?view=aspnetcore-2.1) pattern where access to system resources is guarded by one or more policies and where each policy is evaluated against user information.

The availability of user information for driving policy-based authorization is dependent on the type of User. The HealthStats NSW application should be available for 3 types of users: anonymous external, authenticated external and authenticated internal windows users. The internal and external users are authenticated differently, and the anonymous users are not authenticated at all.

A hybrid approach is required to service these three types of users. ASP.NET Core does support mixed authentication out of the box. Alternatively, the Ocelot package can be used to create custom ASP.NET Core application gateways that delegate the authentication process to external and internal users separately.

The diagram below reveals the security components of the system:



Public users come through the external firewall to the proxy application that ensures the user has a valid JWT token. For an anonymous user the application issues the “anonymous user” token. The request then gets redirected to HealthStats website through the internal firewall, and response returned to the user.

The internal office users come to the HealthStats website directly without any authentication token. In this case the application throws an authentication challenge request using windows authentication schema, like the other standard office web applications.

Therefore, when application wants to check the user permission, it relies on the policy-based permission provider, which consumes the user claims and returns the list of policies available to this user according to the business logic coded. This will give us enough flexibility to control user access from global controller/page level down to single control level.

## RESTfull api’s

ASP.Net Core is the recommended framework to use for API development. APIs will be grouped based on Business functions and will be built within the same project. Each project can be deployed independently and isolated from other projects. It will be built using .Net Core libraries so that it may be deployed to operating systems other than Microsoft Windows Server.

Dependency Injection built-in within the ASP.NET framework and common coding-patterns will be employed to build loosely coupled and unit-testable components.

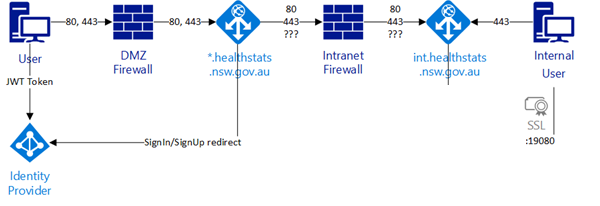
## failover protection

* Load balancing
* Health Monitoring

# architecture

## infrastructure

Based on the technologies selected in the previous section the diagram below indicates the major infrastructure components for the solution and how these components will communicate.



|  |  |  |
| --- | --- | --- |
| # | Element | Description |
| 1 | User | Any public user station, using the supported screen size and browser version. |
| 2 | DMZ Firewall | Security system that isolates the DMZ from internet. |
| 3 | \*.healthstats.nsw.gov.au | Custom gateway application that routes multiple subdomains to the appropriate ports. Contains the security certificate installed. Validates the security tokens provided and forces the challenge requests when required. |
| 4 | Identity Provider | The identity provider for all public users. |
| 5 | Intranet Firewall | Security system that isolates the Intranet from DMZ. |
| 6 | int.healthstats.nsw.gov.au | Intranet load balancer for internal and external traffic. |
| 7 | Internal User | Windows-authenticated staff user. |

## Data architecture

## solution architecture

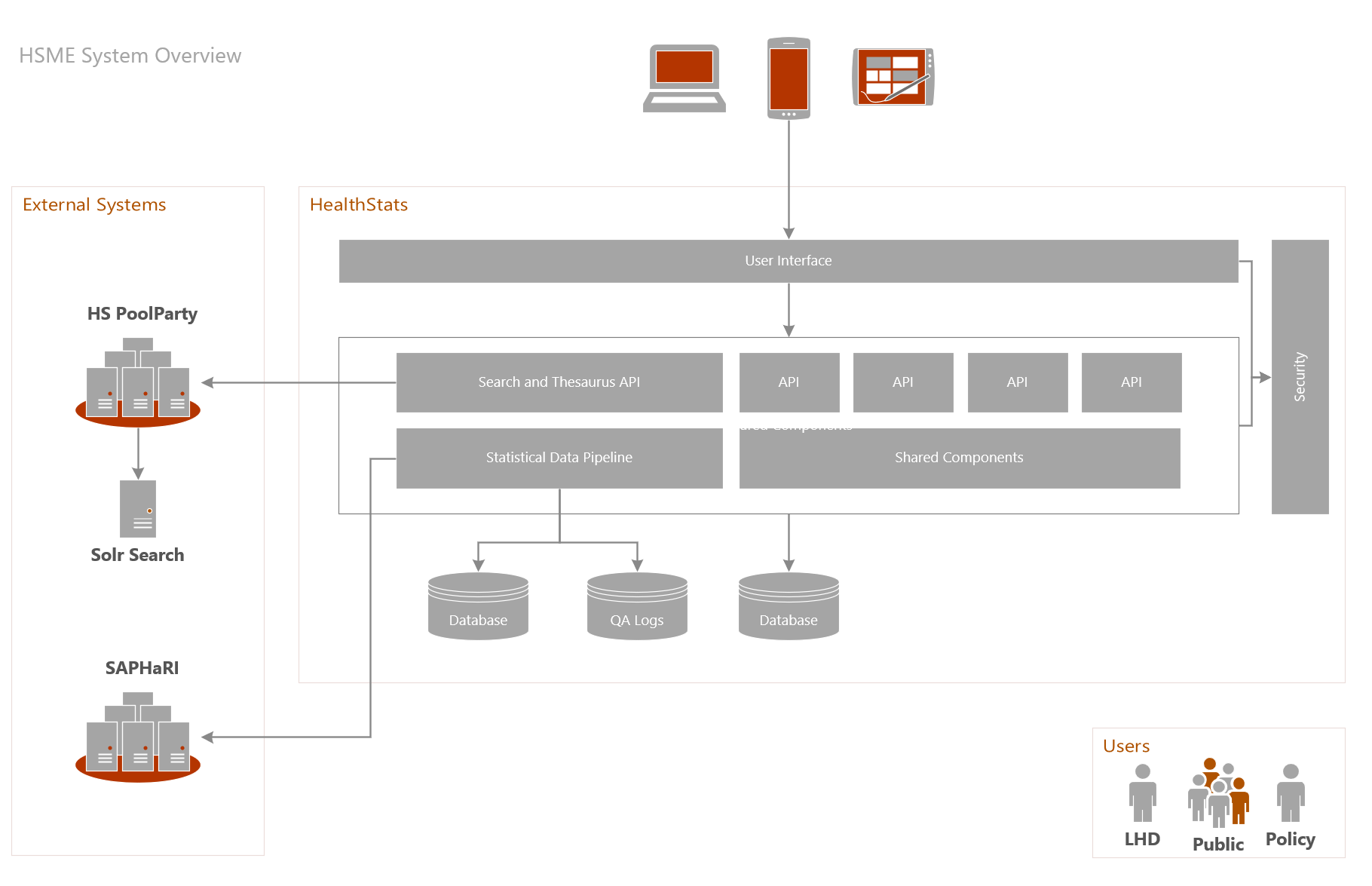
Please see [Solution Components](#_solution_components) for details of the solution architecture.

# Architectural RevIew

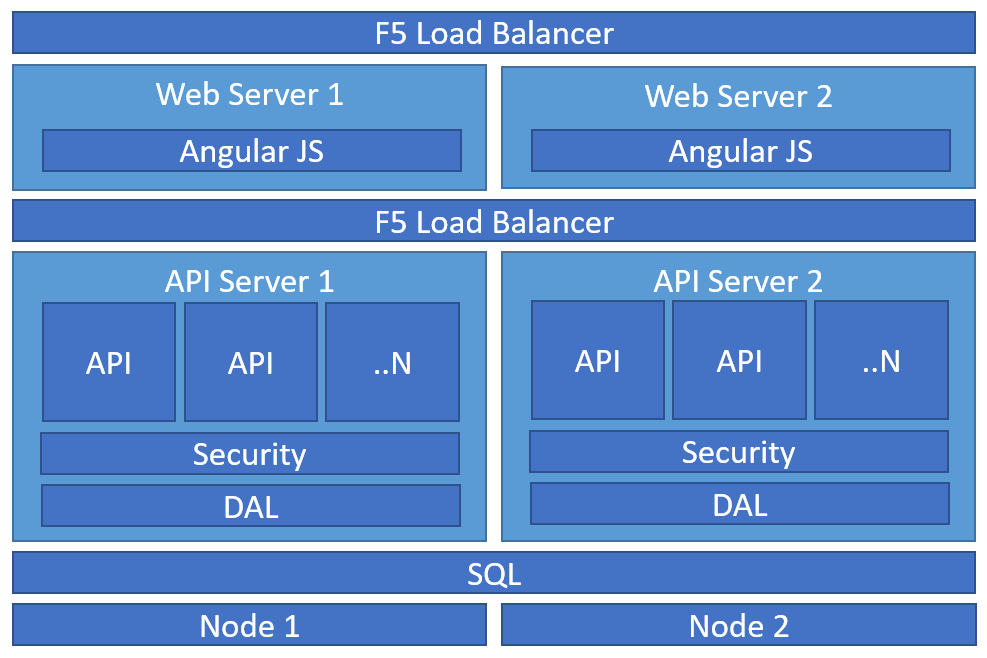
This section provides details of the review and evaluation and process. It describes the objectives, requirements, including all the options considered, and the reasoning behind each decision.

This includes recommendations for the application environment and hosting, business logic components, user interface including graphing and printing components, data Integration and workflow.

## System Architecture



## High Level Application Architecture



**Description:**

The diagram above represents the proposed physical architecture of the servers for Health stats app.

It is recommended to have a couple of front end servers hosting the portal as well as the APIS to make the application highly available and also provides good performance, failover, scalability.

In order for the SQL server data platform to comply with high availability mechanism, clustered should be set to “Always On”.

## Cloud TransfoRmation

MOQ’s recommendation is to host the solution in cloud however due to compliance requirements within MOH it is necessary to keep it on premise, however, the proposed architecture and underlying technology could easily be lifted (as is) into Azure App Services with the Data Tier moving to Azure SQL.

It is important to note that azure is not the only cloud platform the solution can move, amazon and another platform would be considered too.

## 

## Components Review

### Business Logic and Server Processing

This section highlights the proposed technologies/frameworks matching the requirements and the reasons behind choosing one of them for the project.

| Component | What is it? | Reasoning | Considered Options |
| --- | --- | --- | --- |
| APIs -Application Logic:  .Net Core | The web framework used to build core components of portals apart from user interface. | **Key Requirement: Portability**  Although Node JS is increasingly popular, given the **current technical landscape within MOH, the existing skills and the suitability of the platform**, a .Net Core platform was selected. | * Node.JS * .Net Core |
| Runtime  IIS + .Net Core | The web-runtime environment where code is hosted, deployed, executed. | **Constraint: Must run on-premise.**  **Requirement: Must Provide HA.**  .Net Core is supported in IIS, Containers and Service Fabric. Given the technical architecture is not traditional Microservices approach, Service Fabric primarily would provide container orchestration. In addition, it adds additional complexity to the build, with a general skills shortage in the market.  As the actual need for Scale (very low) and Failover is simple, the requirement does not demand the additional technical overhead of Service Fabric or Container technology. Thus, a traditional Web Server (IIS) load balanced topology is suitable.  The technology stack can be ported to a Container environment or directly into cloud services such as Azure App Services if this becomes a future direction. | * Service Fabric * Node.JS * IIS * Containers |

### User Experience

#### User Interface

User interface contains the visual elements and client-side processing required for the application to work property.

MOQ’s recommendation is to use **Angular Framework** as it is a responsive modern web based open source interface with simple packaging and extensive library components, not to mention the strong support from google and community,

Also, de facto use of Type Script providing a high performant easier migration path for traditional non-JavaScript developers.

The following frameworks were considered during selection/evaluation process:

* Net Core ASP MVC / RAZOR (product of Microsoft)
* Angular (backed by Google)
* REACT (backed by Facebook)

#### User Interface Accessibility Compliance:

Link to document HSNSW 2.0 – Level AA standard to be added here

#### Graphing Visualisation:

The main requirement for this item is provide power users/administrators the option to build complex charts that would support updating filtered data in the user interface without a full page post back.

MOQ’s recommendation to leverage the “**Plot.LY**” java script tool that allows charts and dashboards selection on the front end.

**RISKS**:

There are many libraries based on the native **D3.JS platform** and others that use alternative technologies. This space is generally considered fragmented. This means that developer experience is limited.

Selecting a front-end charting library means that traditional BI developers would not be in a position to create new charts as they are coded directly within the web tier. Products like Tableau and **PowerBI** provide for embedded options that can render high fidelity charts natively within html / js driven pages. These tools are typically used by BI visualisation experts. However, these tools do not support the entire solution scope and attract significant additional licencing.

Thus selecting Plot.LY as which is a fit for the charting needs is suitable, with the noted risk that general developer ramp uptime will be required.

The following frameworks were considered during selection/evaluation process:

* Native D3.JS
* Various Chart Libraries
* Plot.ly
* POWERBI

#### Printing Output:

The main requirements are to allow end users to print charts related to specific indicators along with meta data and specific text if needed to.

Printing in a **paginated form** in web based solutions is not a trivial task. Added to this the need for embedding **high-fidelity charts.**

Plot.LY supports the export of chart data to images. **A document generation library** will be used to create standard form (PDF/Word\_ outputs and insert the Images (from the charts) and the text content from the article/indicators. This will allow for proper pagination.

**A separate API** will be created and used to generate the output. The output reports will not look like the rendered Angular pages, due to the nature of formatted printed content.

### Data Management & Processing

#### Data Storage Platform:

The data storage engine/platform to store all information that would be used later the portal.

Requirements mentioned the need for a high performant relational data storage platform that Should not introduce an additional unsupported platform into MOH environment.

MOQ’s recommendation is to use Microsoft SQL Server for the following reasons:

* A relational data storage platform available both on premise and supported in the cloud (future use cases) that works well with technology selections.
* Existing capability and skill within MOH
* Currently a supported platform.

#### Data Transformation

Data transformation is the process of data extraction/transformation before pushing it into the target system database. (Where it will be approved through the workflow before committed to).

The requirement is extract data from CSV files exported from SAS and SAPHaRI, apply required transformations and push the data into system database.

MOQ’s recommendation is to use SSIS (SQL Sever Integration Services) as it has been used widely within the current environment existing skills can be leveraged and potentially some of the existing artefacts can be modified.

Like Microsoft SQL the SSIS component is widely used, and skills are available in the market.

The following frameworks were considered during selection/evaluation process:

* SSIS
* Python
* Azure Data Factory.

#### Versioning

Versioning is storing/auditing multiple versions of data over time.

The requirements are to allow admin/back end users to review indicators results in different times and **compare the new ready to publish results against currently published results** during approval.

The way to achieve this feature is by **adding tracking information including version number as well as date time during the data transformation** stage each time new data is imported from SAPHaRI along with making the data as unapproved until a user approves it through the admin back end.

### Content Management

Content Management system allows admin users to create and manage digital content.

The requirement is to allow admin/back end users to **manage digital content such text, landing pages, images and navigation** in a reliable way.

The recommendation is to build a **custom module to achieve** the desired functionality.

The reason being is that building the Health Stats solution on an industry standard CMS such “Kentico“, “Site Core“ or “Magento” will significantly add complexity to the system where most of the features in the CMS would not be utilised. Ultimately a highly customised CMS would remain reducing the number of the developers available in the market who support it. In addition, the currently envisaged usage processes would need to be adapted to work within the framework provided by the CMS.

For example: a feature like bulk update for metadata such as data source would be hard to natively implement in a traditional CMS, resulting in a custom module.

### Workflow Management

Workflow is one of the core components in Health stats as it manages approvals, reviews and the associated publishing of articles, indicators and topics.

Although the workflow is considered by the business to be complex at a technical level it is not. **Essentially all the workflow approvals are based on a state machine** that can easily be represented in a standard custom state machine. Adding a 3rd Party BPM tool or a Workflow Engine would introduce undue complexity.

The following frameworks were considered during selection/evaluation process:

* .Net Core state machine
* K2 .net
* Windows Workflow Foundation
* Workflow Core

### Logging

#### Application Logging

Logging is used to tracking events happening inside the application, usually errors to help with troubleshooting

The requirement to provide a reliable mechanism for admin to troubleshoot the application.

Based on previous implementations, we recommend using Log4Net, which can push the logs to the database to have one central place holding logs from all servers.

Note: Within the cloud, application insights, provided monitoring and logging function natively as an Azure PaaS service. This includes performance metrics, application health analysis, hardware consumption and reliable dashboards.

The following frameworks were considered during selection/evaluation process:

* Log4Net
* Elmah
* NewRelic
* Application Insights

#### 3rd Party Website Analytics

To further enhance the ability to monitor and measure user and actions, Website Analytics may be incorporated into the website. This 3rd party product must be able to work seamlessly with SPA websites.

#### 

## Environment & CI/CD

Environments are the combination of essential software and hardware to run the application.

Additional environments have been identified discussed to **improve risk and software release control**. The following have been identified as potential candidates to support the suggested software development lifecycle:

**Development:** The environment where developers code from, and a central server.

**Test:** The environment where developers code is ‘tested’ with other dependent systems.

**Staging:** The environment where developers code can be tested by end-users (acceptance criteria), and possibly used for training & demonstration use cases.

**Production:** The production environments require HA capability and need to scale to serve the use cases.

### DevOps Process

Azure DevOps will be holding a CI/CD pipeline to pick-up any code merges as a result of Pull Request review. The system will perform the build, run Unit / Integration and Acceptance tests (with external dependencies been mocked through a mocking framework) then automatically deployed to Development environment. These acceptance test will be built with business-friendly Gherkin language with Specflow tool. This tool integrates seamlessly with the developmental tools.

Once deployed in Development environment, an approval process to deploy to the Test environment takes place. Once Approved then deployment to Test environment can be triggered.

The preparation process for Production release will also be automated and will only to package artefacts for release (pre-prod/prod). It will control release tagging along with environment configurations for Web/API/Database endpoints.

## sUMMARY

### APIS

| Original Architecture | Recommended Architecture | Reasons |
| --- | --- | --- |
| Microservices | **Managed APIs** | * The microservices pattern is not the best fit for this project as it requires the data to be loosely coupled which is not case here. * Requires additional logic to be implemented in order to keep the data consistent * The ‘Services’ noted within the original architecture document have ‘wide’ domains, and thus do not espouse the core principle of single responsibility with microservices. These domains align better to business APIs. |
| Service Fabric | **.Net Web API** | * SF is designed specifically to serve micro services * SF adds extra configure effort and complexity, sometime as much as development * SF will limit the number of developers available to support the application |
| Containers | **No Containers** | * Containers are useful when uptime and scale requirements indicate the need * Containers are useful when supporting microservices architecture * We do not believe the additional complexity of introducing container orchestration software to the environment will evoke the benefits |

### Security

The following summarizes MOQ’s recommendation regarding security

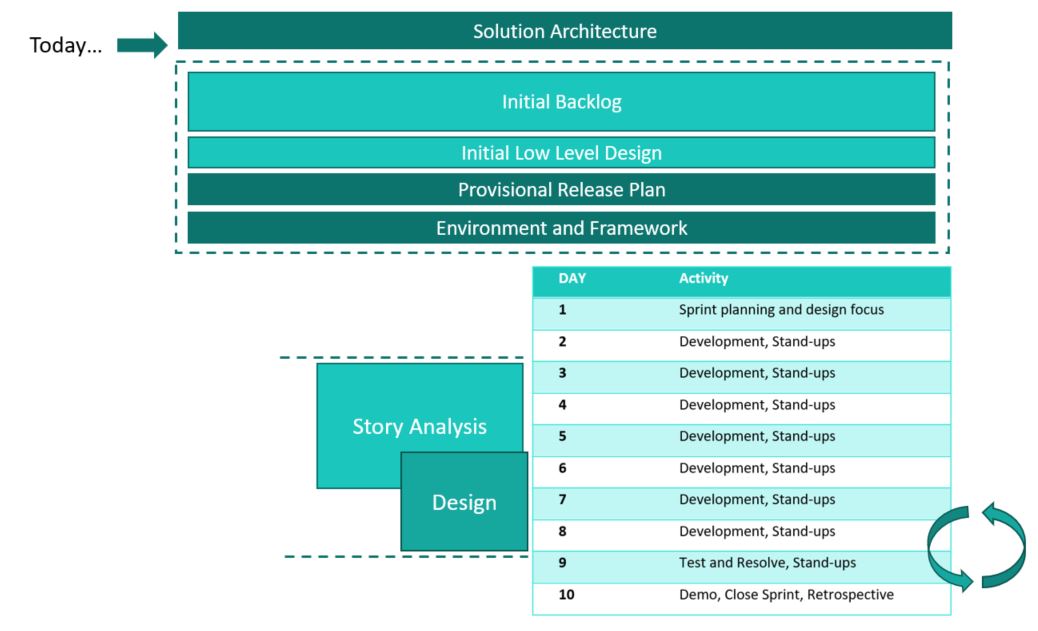
* Application should support mixed mode Authentication Windows & JWT.
* Multiple roles and polices should be available according to requirements.
* Admin user shall have screens allowing him to reset password, assign different roles to different users.

# Project Execution Plan

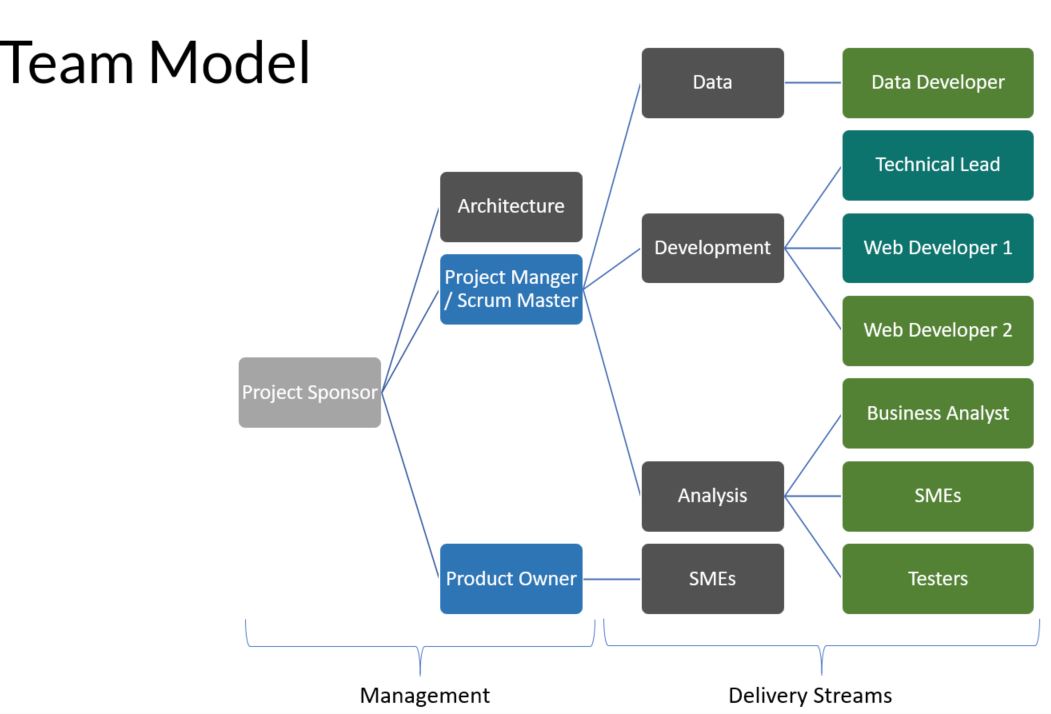
This section details the proposed approach and team composition in order to achieve the target outcome.

## Agile / Recommended Approach

This



## Team Model



# Development Tools and processes

## system development

As ASP.net provides the foundational technology the clear preference for development platform is Visual Studio. Visual Studio Team Services (VSTS) provides a very convenient mechanism for integrating Visual Studio developers into a team and allowing them to integrate their code into a single project.

VSTS goes well beyond integrating developers into a single team. It also provides capabilities for managing an Agile project including specification of requirements via User Stories and work backlogs via Kanban. It also provides project build, test management, defect tracking, deployment, and release management capabilities

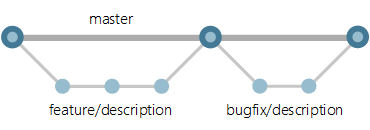
## source control

Any Git-based repository is preferable option for its lightweight and performance. Visual Studio Team Services can provide a Git-based repository together with backlog tasks and Kanban board out-of-the-box for Agile development.

Git stores history as a graph of snapshots (commits) of the entire repository. Each commit also contains a pointer to one or more previous commits. Branches in Git are simply pointers to specific commits, which makes branching process lightweight and easy. Branches defined by names and the mainline branch in Git is named *master*.

It is recommended for developers to create a new branch for every feature to be developed. It's common to have multiple active branches where someone is working on different features in parallel. Therefore, the development process for a feature ticket #123 contains the next steps:

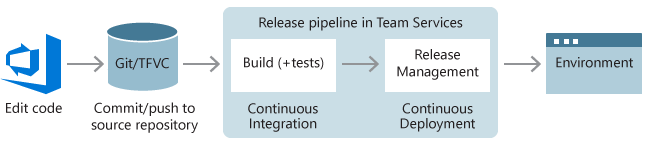
* Sync parent branch to ensure you have the latest copy of the repository. Master is usually a default parent branch, but team may create an isolated branch for a user story and create a multiple child feature branches from it.
* Create a new branch from parent branch, providing the name; it is common to include feature name in the branch name, for example cool-new-feature (kebab style is used to make it URL friendly). Branch folders can also be used to organize branches. It is usually useless to include the author’s name into the branch name, as many developers can contribute into the same branch, but using user names as branch folders is a common practice.



* Develop the code, making commits as often as you need. Every commit will be associated with the related work item (VSTS feature ticket #123 in this case). It is a common practice to keep commits relatively small to be able to undo certain commit and keep the others, if something has been done wrong.
* Once code change is ready to be reviewed, it is useful to switch (checkout) to the parent branch again and perform a sync. There should be not conflicts at this point, as the developer did not make any changes in parent branch directly.
* Switch back to the feature branch again, and perform a branch rebase to parent action. Rebase will re-attach the first feature commit to the latest commit in the parent branch, aligning all the history.
* Resolve all conflicts, if any, and commit them.
* Create a new pull request for this branch, asking for a code review from the teammates.
* Fix all code issues raised and commit the change, until code change passes code review.
* When changes are checked-in, the CI/CD will automatically trigger a build and all automated test will be ran.
* Once build and all automated tests pass, the approved code change will be squashed into a one single commit and committed into the parent branch.

## continuous Integration / continuous delivery

VSTS CI/CD pipeline is recommended for automatic build, test and release management.



It is recommended to configure the build process to be built automatically as soon, as feature branch get merged into master branch.

triggers to different deployment targets should be configured separately:

|  |  |  |
| --- | --- | --- |
| # | Environment | Trigger |
| 1 | Development | Automatically, as soon as a new release has been built. |
| 2 | Test | Code changes have undergone Peer Review and Developer Testing. Automatically, as soon as a new release has been built and |
| 3 | Staging | Automatically, after release manager has approved a staging deployment. |
| 4 | Production | Automatically[[1]](#footnote-1), after business[[2]](#footnote-2) has approved a production deployment. |

## Environments

One environment is updated after each code committed into the master branch. This is how developers can be sure, that their feature works not only on their machine.

Another environment is required for QA staff, where all changes will be deployed together, so the team can be sure changes are not conflicting with each other. This is the QA/Testing environment.

A UAT environment is required in case if we can have an ability to deploy to Production without overwriting the current version. That new version will work as UAT, and once tested, we will redirect all traffic there.

Staging environment would also be helpful to test the backup/restore process for all stateful services. Same process should be used to clone the environments and/or data migration.

## deployment

# disaster recovery

## backup process

* Daily Database Backup will be performed.
* SQL Data will be kept for 30 days.
* SQL Log will be kept for 90 days.

## recovery process

* The initial process is to determine which components are affected and requires recovery.
* If the Application component restoration process will be performed by eHealth systems administrator. Artefacts that were used in the most recent deployments will be used to restore the site. Note that Production deployment does not make use of DevOps CI/CD infrastructure.
* If the Database server is affected and requires restoration of database, then restore database using the data and log backups.

# Server Specifications

## DEV

The Development environment will require the following minimum hardware specifications.

### Web/API Environment

* Load Balancer, Layer 4, SSL Passthrough, Highly Available
* 2x Web Servers (Virtual or Physical)
* 32GB RAM
* 4 CPU's (8 threads)
* Storage:
  + DISK 1 120g SSD: OS (Primary)
  + DISK 2 300g SSD: DATA (Secondary)
  + Storage Parity (Highly Available)
* Network:
  + Throughput 10g
  + Highly Available
* OS: Windows Server 2016 Standard (or above)

### SQL Environment

* HA SQL Environment, Suggested Spec
* SQL Availability Group (2 Nodes)
* OS: Windows Server 2016 Standard (or above)
* APP: SQL 2016 Standard (or above)
* Storage:
  + Disk 1 250g SSD: OS and SQL (binaries)
  + Disk 2 XXXG SSD: Data (size to be determined)
  + Disk 3 XXXG SSD: Logs (size to be determined)
* Network:
  + Throughput 10g
  + Highly Available

## Staging/Production

Staging and Production server specifications are identical.

### Web/API Environment

* Load Balancer, Layer 4, SSL Passthrough, Highly Available
* 2x Web Servers (Virtual or Physical)
  + 32GB RAM
  + 4 CPU's (8 threads)
* Storage:
  + DISK 1 120g SSD: OS (Primary)
  + DISK 2 300g SSD: DATA (Secondary)
  + Storage Parity (Highly Available)
* Network:
  + Throughput 10g
  + Highly Available
* OS: Windows Server 2016 Standard (or above)

### SQL Environment

* HA SQL Environment, Suggested Spec
* SQL Availability Group (2 Nodes)
* OS: Windows Server 2016 Standard (or above)
* APP: SQL 2016 Standard (or above)
* Storage:
  + Disk 1 250g SSD: OS and SQL (binaries)
  + Disk 2 XXXG SSD: Data (size to be determined)
  + Disk 3 XXXG SSD: Logs (size to be determined)
* Network:
  + Throughput 10g
  + Highly Available

Proposed solutions evaluation table.. here

1. This will require port 19000 to be opened, so VSTS CI/CD process can upload new packages into the Service Fabric otherwise, half-manual process will be required, when someone from IT staff will execute a PowerShell script with proper parameters to download and upload new application packages. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)