|  |  |
| --- | --- |
| **Date** |  |
| **Version Number** |  |
| **Owner** |  |
| **Document Id** |  |
| **Primary Audience** |  |
| **Document Location** |  |

*[Project]*

High-Level Solution Design Architecture

|  |  |  |
| --- | --- | --- |
| Review & Approval | | |
| **Reviewed by** | **Job Title/Role** | **Date** |
|  | Data Architect |  |
|  | Quality Assurance |  |
|  | Software Development Manager |  |
| **Approved by** | **Job Title/Role** | **Approval Date** |
|  | System Design Authority |  |
|  | Project Manager |  |
|  | Software Development Manager |  |
|  | Enterprise Architect |  |

|  |  |  |  |
| --- | --- | --- | --- |
| History | | | |
| **Version #** | **Author** | **Comments** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| Support & Reference Materials | | |
| **Ref** | **Title** | **Document Id** |
| 1 | Information Security Policy |  |
| 2 | Data Sensitivity Classification Policy |  |
| 3 | Architecture Policy |  |
| 4 | Architecture Glossary |  |
| 5 | Business Requirements – [Project] |  |
| 6 | Non-Functional Requirements – [Project] |  |
| 7 | Data Architecture – [Project] |  |

This Word document template is based on a hypothetical system upgrade example, which means there is an AS-IS and a TO-BE aspect so that the impact and business and technology can clearly be understood once this design is implemented. If this were a new system, then there wouldn’t need to be an AS-IS section. The hypothetical design outlined in this template consists of a client-server system comprising database servers, internet servers and development-support servers, with three environments: DEV, UAT and PROD, all on virtual networks. The example servers are modelled as on-premises VMs but could just as easily be applied to a cloud-based solution. Remove all comment sections like this here in italics before the final presentation. The Version number and the Document Identity fields should be set up in Word to automatically update when this document is checked back into SharePoint. Remove all comment sections in italics and example code before the final presentation.

Instead of using Visio, this template uses Mermaid markdown code in Visual Studio (mark-down plugins required) or mermaidchart.com for illustrations. The code for the illustrations is included in the guidance text and needs to be adapted to suit the solution.

# Contents

[1. High Level Solution Design Architecture Scope 4](#_Toc190678147)

[2. High-Level Architecture Design Decisions 7](#_Toc190678148)

[3. AS-IS Baseline Architecture 10](#_Toc190678149)

[4. AS-IS Network Architecture 22](#_Toc190678150)

[5. AS-IS Error Architecture 28](#_Toc190678151)

[6. AS-IS Security Architecture 28](#_Toc190678152)

[7. AS-IS Post-Go-Live Architecture 28](#_Toc190678153)

[8. TO-BE Contextual Architecture 29](#_Toc190678154)

[9. TO-BE Conceptual Architecture 29](#_Toc190678155)

[10. TO-BE Logical Architecture 29](#_Toc190678156)

[11. TO-BE Component Definitions 30](#_Toc190678157)

[12. TO-BE Technical Environments 31](#_Toc190678158)

[13. TO-BE Error Architecture 36](#_Toc190678159)

[14. TO-BE Network Architecture 37](#_Toc190678160)

[15. TO-BE Network Services 40](#_Toc190678161)

[16. TO-BE Security Architecture 43](#_Toc190678162)

[17. Backup and Restore 46](#_Toc190678163)

[18. System recovery 47](#_Toc190678164)

[19. Support Model 48](#_Toc190678165)

[20. TO-BE Solution Deployment 49](#_Toc190678166)

[21. Solution Dependencies 54](#_Toc190678167)

[22. Solution Review and Assessment 55](#_Toc190678168)

[23. TO-BE Batch Processes 59](#_Toc190678169)

[Appendix A: Interfaces 61](#_Toc190678170)

[Appendix B: Access Control Choices 71](#_Toc190678171)

[Appendix C: Non-Distributed System Patterns 78](#_Toc190678172)

[Appendix D: Distributed System Patterns 83](#_Toc190678173)

[Appendix E: Scalability Considerations 92](#_Toc190678174)

[Appendix F: API Gateway Configuration 99](#_Toc190678175)

# High Level Solution Design Architecture Scope

## Identification

This [project] is a [functional description].

## The Function of Solution Architecture

The Solution Architecture of a project translates the business requirements into a design concept, which is then translated into IT operations. During this journey, the best-fit solution is defined to solve the set of problems that the business requirements allude to, within the prevailing constraints of governing policies, architectural principles, IT strategy, project budget, skillsets, available technologies, allotted time and the availability of frameworks within the organisation that can support the initiative from a project and business perspective.

## Architectural Concepts

Figure 1 below shows a layered view of the architectural concepts used in the delivery of a complex system, with matching explanations in Table 1. This document is primarily concerned with the “Architecture” bit.

A screenshot of a computer screen

Description automatically generated

Figure 1 Layered view of architectural concepts

[FYI: This is the code in mermaid.js. Now you know. It beats Visio any time.]

```mermaid

block-beta

columns 1

block

Sy("System:\nAll the related parts and rules")

end

block

Ar("Architecture:\nHow it is put together")

Mo("Model:\nHow it works")

end

block

Fr("Framework:\nHow to build it")

Me("Methodology:\nHow to use the tools to build it")

end

block

Bl("Blueprint:\nThe plan/specification for building the system")

De("Design:\nThe end result")

In("Infrastructure:\nPhysical components and resources")

end

style Sy fill:#cdddcd,stroke:#0

style Ar fill:#ddbdbd,stroke:#0,stroke-width:6px,font-size:14px,color:#000

style Mo fill:#ddbdbd,stroke:#0

style Fr fill:#ababab,stroke:#0

style Me fill:#ababab,stroke:#0

style Bl fill:#9090cc,stroke:#0

style De fill:#9090cc,stroke:#0

style In fill:#9090cc,stroke:#0

```

| **Concept** | **Explanation** |
| --- | --- |
| **Architecture** | A high-level abstraction of the system, that defines structure, function, components and interactions, i.e. how it is put together. |
| **Blueprint** | The plan/specification for building the system, i.e. manufacturing/fabricating instructions, e.g. functional and non-functional specifications based on the requirements |
| **Design** | The process and outcome of having configured a system, |
| **Framework** | A foundational set of tools, libraries and best practices, i.e. how to build it, as can be shown with e.g. a functional decomposition. |
| **Infrastructure** | Physical components and resources that constitute the system |
| **Methodology** | A set of methods and principles for building a system, i.e. how to use the tools to build the system, such as Agile and DevOps |
| **Model** | A particular implementation of an architecture, describes the working parts of the system, i.e. how it works, as can be illustrated with e.g. a data model |
| **Observability** | The ability to measure the current holistic state of a system by examining the data collected across multiple layers of the system's architecture. |
| **Scalability** | The ability of a system to handle an increased workload without losing performance or compromising availability. |
| **System** | All the related parts and rules that form the unified solution or enterprise. |

Table 1 Concept explanations

## Document Purpose

The purpose of this High-Level Solution Design (HLD) is to present the existing AS-IS solution or target estate against the new TO-BE solution, in terms of the application and technology architectures, in order to illustrate how the new solution will fit in the target estate or replace the existing solution and what the impact to peripheral systems already on the estate might be. The information and data architectures are covered in the Data Architecture Document [Ref. 7]. It also demonstrates how the solution complies with [company]’s core architectural principles and policies and notes any exception from these.

Specifically, it provides:

* A bird’s eye view of the system by describing the contextual, conceptual and logical breakdown of the solution.
* Changes or additions of business processes and workflows that are impacted by the introduction of this solution
* A overview on changes to operational practices and identifies new required skills due to the resulting business changes and the introduction of new technologies that need to be supported or that the system’s users need to be become acquainted with.
* A description of the application and software modules that are at the heart of the solution
* Computational hardware, storage and other related system components (crypto-modules, CUDA processing units, AI processing devices, if used)
* Networking and security and their respective technical implementations
* Failover prevention and disaster recovery strategies for this solution

## Document Scope

This HLD presents the application- and technology-related aspects of the [project].. The scope of the solution is shown in Figure 1.

This HLD will:

* Present the technical design in terms of infrastructure and software components
* Describe the Configurable Items (CIs) that need to be implemented for this project
* Highlight attributes of the solution architecture that relate to non-functional requirements specified in the NFR [Ref. 6]. These include the Security, Safety, Performance, Maintainability and Availability.

## Intended Audience

The target audience for this document includes business, technical and non-technical, governance, and project management stakeholders. Specific users of this document include solution architects, data architects, developers and test analysts. This HLD uses technical terms which should be understandable to the indicated audience. A general Architecture Glossary exists to aid understanding [Ref. 4].

# High-Level Architecture Design Decisions

## Overriding Design Decisions

The table below identifies key design decisions that were specifically made for the [project]’s solution architecture. The decisions were made with SMEs to ensure agreement across all components that constitute the [project].

| **ID Number** | **Decision** |
| --- | --- |
| SAD-01 |  |
| SAD-02 |  |

Table 2 Solution Architecture-specific Decisions

The design from the Solution Architecture HLD and the design decisions taken from Table 2 are used to shape the data architecture below. The decision rationale is based on the following:

## Exceptions from [company]'s Technology & Data Architecture principles

[List the exceptions from the architectural guiding principles that had to be made in this design]

| **EXP-01:** | ***[Design aspect]*** |
| --- | --- |
| Principle: | *[Principle that was broken]* |
| Description: | *[How is this principle broken]* |
| Justification: | *[Why is this principle broken]* |
| Recovery Plan: | *[How, if ever, will this be remedied]* |
| **EXP-02:** | ***[Design aspect]*** |
| Principle: | *[Principle that was broken]* |
| Description: | *[How is this principle broken]* |
| Justification: | *[Why is this principle broken]* |
| Recovery Plan: | *[How, if ever, will this be remedied]* |
| **EXP-03:** | ***[Design aspect]*** |
| Principle: | *[Principle that was broken]* |
| Description: | *[How is this principle broken]* |
| Justification: | *[Why is this principle broken]* |
| Recovery Plan: | *[How, if ever, will this be remedied]* |

Table 3 Exceptions from guiding architectural principles

## Business Capability Impacts

Show the latest Capability Map of the business and highlight those capabilities that will be impacted by this solution. Note that a Capability Map is not an organizational map. It focuses on what a business does, not how it does it.

The Business Capability Map in Figure 2 below shows the business activities that will be impacted by the implementation of this solution. The impacts are detailed in Table 4 below.

A blue and orange rectangular box with text

Description automatically generated with medium confidence

Figure 2 Affected areas [company]s Capability Map

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Impacted area** | **Impact Details** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

Table 4 Impacted business areas

## Enterprise System & Technology Impacts

Refer to Figure 3, which shows a high-level view of the overall technical architecture in the enterprise. This solution will impact the existing enterprise technology in the marked-up areas and are detailed in Table 4.

A diagram of a cloud computing system

Description automatically generated

Figure 3 High level of the technical architecture

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Impacted Technology** | **Impact Details** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

Table 5 Impacted Systems & Technologies

## Alternative Solution Approaches Explored

Provide a brief overview of what alternative remedies/approaches/vendors have been explored with reasons. Consider these points:

* Vertical scaling by upgrading memory or processors
* Horizontal scaling using sharding, clustering, replication, multiple-read-only etc. techniques
* Disk I/O too slow, or too much Disk I/O instead of in-memory processing
* Possible database configuration changes
* Database tuning: Table design, indexed, compaction strategies, garbage collections, memory usage, query and stored procedure fine-tuning
* Storage partitioning schemes for physical storage devices
* Network tuning, data pipe finessing, client application tuning, API finessing
* Cache the database front-end
* Experts and vendors consulted.

# AS-IS Baseline Architecture

[There is no need for this AS-IS section if there is no existing solution that will be replaced or enhanced by the new solution]

This chapter describes the AS-IS architecture of the existing solution.

## AS-IS Contextual Architecture

[Simple description of the business context under which the solution currently operates.]

The contextual business model describes the business activities that meet the business requirements [Ref. 1].

[Instead of using Visio, you can use this Mermaid.js code in Visual Studio or mermaidchart.com for this illustration:]

flowchart LR

%% Nodes for the Order Fulfillment Process

A[("fab:fa-cart-plus Order Received")]

B["fa:fa-box Inventory Check"]

C["fa:fa-clipboard-list Order Processing"]

D{"Is Item Available?"}

E["fa:fa-shipping-fast Shipping"]

F["fa:fa-truck Dispatch"]

G[("fa:fa-file-invoice Generate Invoice")]

H["fa:fa-envelope Customer Notification"]

I["fa:fa-box-open Order Packed"]

J[("fa:fa-check-circle Order Fulfilled")]

%% Edge connections between nodes

A --> B --> C --> D

D -- Yes --> E --> I --> F --> J

D -- No --> G -- Notify --> H

%% Styling for specific nodes

style A fill:#dddddd

style G fill:#dddddd

style J fill:#dddddd

flowchart LR

    A(Start) --> B{"Validate Payment Info"}

    B -- Valid --> C[Payment Gateway]

    B -- Invalid --> G[Payment Failure]

    C --> D["Transaction Processing"]

    D -- Approved --> E[Confirm Payment]

    D -- Denied --> G[Payment Failure]

    E --> F(End)

    G --> F

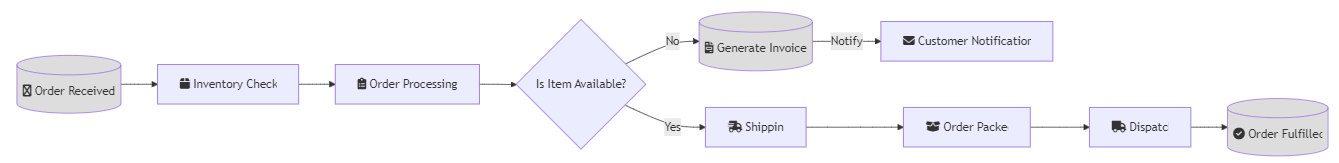


Figure 4 AS-IS Contextual architecture diagram: Customer order fulfilment

A diagram of a computer

Description automatically generated

Figure 5 AS-IS Contextual architecture: Customer order payment

## AS-IS Conceptual Architecture

[Show and explain the current user operation in a workflow diagram. Swim lane diagrams with business area or system in each lane.]

A diagram with text and words

Description automatically generated with medium confidence

Figure 6 Conceptual architecture for Online Payment Process with own Payment Service Provider (PSP)

```mermaid

sequenceDiagram

title Online Payment Process with own Payment Service Provider (PSP)

actor Customer

participant Merchant

participant Payment\nGateway

participant Fraud\nChecker

participant Acquiring\nBank

participant Card\nSchemes

Customer->>Merchant: Place an order

Merchant->>Payment\nGateway: Payment form

Payment\nGateway->>Merchant: "Paying..."

Merchant->>Customer: "Paying..."

Payment\nGateway->>Fraud\nChecker: Fraud check

Fraud\nChecker->>Payment\nGateway: Fraud Fail

Payment\nGateway->>Merchant: Fraud Fail

Payment\nGateway->>Acquiring\nBank: Success

Acquiring\nBank->>Card\nSchemes: Success

Card\nSchemes->>Issuing\nBank: Sufficient funds?

Issuing\nBank->>Card\nSchemes: Pass/Fail

Card\nSchemes->>Acquiring\nBank: Pass/Fail

Acquiring\nBank->>Payment\nGateway: Pass/Fail

Payment\nGateway->>Merchant: Pass/Fail

Merchant->>Customer: Pass/Fail

```

A diagram with many lines

Description automatically generated with medium confidence

Figure 7 Conceptual architecture for Online Payment Process with 3rd-party Payment Service Provider (PSP), e.g. Stripe, PayPal]

```mermaid

sequenceDiagram

title Online Payment Process with 3rd-party Payment Service Provider (PSP), eg. Stripe, PayPal, etc

actor Customer

participant Merchant

participant Payment\nGateway

participant Fraud\nChecker

participant Acquiring\nBank

participant Card\nSchemes

participant PSP

Customer->>Merchant: Place an order

Merchant->>Payment\nService: Payment event

Merchant->>Payment\nService: Payment form\nwith details

Payment\nService->>Merchant: Paying...

Merchant->>Customer: Paying...

Payment\nService->>Wallet\nDatabase: Stores Payment monetary details

Payment\nService->>Ledger\nDatabase: Stores transaction details

Payment\nService->>PSP: Payment monetary details

PSP->>Card\nSchemes: Success

Card\nSchemes->>Issuing\nBank: Sufficient funds?

Issuing\nBank->>Card\nSchemes: Pass/Fail

Card\nSchemes->>PSP: Pass/Fail

PSP->>Payment\nService: Pass/Fail

Payment\nService->>Wallet\nDatabase: Update merchant

Payment\nService->>Ledger\nDatabase: Update merchant

Payment\nService->>Merchant: Pass/Fail

Merchant->>Customer: Pass/Fail

```

## AS-IS Logical Architecture

[Show the application and computational components, network infrastructure, human interface components and storages currently in use in a figure and tables.]

Figure: AS-IS Logical architecture

### AS-IS Component Definitions

[Explanation of the above figure, with brief summary of the technologies involved.]

Definition Vendor Explanation

### AS-IS Client Technology

[Explanation with respect to the above figure, e.g. min browser spec, OS]

### AS-IS Server technology

[Explanation with respect to the above figure]

### AS-IS Support technology

[Describe how the AS-IS is supported – does it use VNC, XWindows, RDP, MS Teams, etc? Who is the current support team? How are software updates performed?]

## AS-IS Application Architecture

[Describe the applications and components in detail and show their interfaces and how the business requirements is achieved.]

## AS-IS Technical Environments

[Describe the technical environments that are used to produce the AS-IS design or are currently supporting the AS-IS design. If they were torn down after the development or deployment, indicate this.]

Example Options:

1. The DEV and the UAT environments will be reused for the Development and Test environments, and the existing PRODUCTION environment will be reused on cut-over.

2. New environments, in which case these need to be specified at the same level of detail as the AS-IS.

| **Environment** | **Function** | **Status** |
| --- | --- | --- |
| DEV | Development | Active |
| TEST | Testing | Mothballed, etc… |
| UAT | User Acceptance Test | Mothballed, etc… |
| QA | Quality Control | Mothballed, etc… |
| PROD | Production | Active |

Table 6 Summary of AS-IS technical environments

### AS-IS DEV (Development) environment

[Describe the development environment and how developers code on it. Are there any particular DevOps processes worth noting?]

### AS-IS TEST/UAT/QA environment

[Describe the TEST and UAT environments in terms of servers and storage and network. Any particular DevOps processes that are worth noting, such as the application’s automatic promotion from DEV to TEST to UAT to QA]

### AS-IS PROD (Production) environment

[Describe the PROD environment in terms of servers and storage and network.]

### AS-IS Client environment

[Describe the technical aspects of the client – usually a device that supports a thin-client interface such as a browser]

|  |  |
| --- | --- |
| **Item** | **Specification** |
| **Device** | Company-Standard laptop |
| **Operating system** | Windows 7 |
| **Inputs** | Mouse, keyboard, touchscreen, headset |
| **Outputs** | Graphical screen, printer |
| **Outputs Programs** | Edge Browser Version 9.10.6 (64-bit) |

Table 7 AS-IS Client specification

## AS-IS Client Environments details

### AS-IS DEV Client Environment

The design of the DEV client environment includes capacity for hosting the development of other development projects. One development workstation is required per developer.

| **Device:** | **DEV-environment client definition** |
| --- | --- |
| Utilisation: | Development Workstation, 1 off per developer |
| Location: | Development Centre |

| **Device Description** | **Specification** | **Notes** |
| --- | --- | --- |
| Replication Required | No |  |
| Operating System | Win11 |  |
| Patch level | Current |  |
| Patch method | PatchMyPC |  |
| CPUs | xx |  |
| Memory | xx GB |  |
| NICs | xx |  |
| VM Machine | No |  |
| VM High Availability | N/A |  |
| VM Anti-Affinity Rule | N/A |  |
| VM Affinity Rule | N/A |  |

| **Applications:** | **License:** |
| --- | --- |
| Visual Studio 20xx | Enterprise License |
| Dot Net xx | Community |
| Tortoise Subversion | Open-source Plug-In to Visual Studio |
| SQL Server | Express |
| IIS x.xx | Community |

Table 8 Physical Architecture: DEV-environment client definition

### AS-IS PROD Client Environment

The design of the production client environment includes capacity for hosting the application and other standard, peripheral support applications for communications and documentation.

| **Device:** | **PROD-environment client definition** |
| --- | --- |
| Utilisation: | Development Workstation, 1 off per developer |
| Location: | Development Centre |

| **Device Description** | **Specification** | **Notes** |
| --- | --- | --- |
| Replication Required | No |  |
| Operating System | Win11 |  |
| Patch level | Current |  |
| Patch method | PatchMyPC |  |
| CPUs | xx |  |
| Memory | xx GB |  |
| NICs | xx |  |
| VM Machine | No |  |
| VM High Availability | N/A |  |
| VM Anti-Affinity Rule | N/A |  |
| VM Affinity Rule | N/A |  |

| **Applications:** | **License:** |
| --- | --- |
| Visual Studio 20xx | Enterprise License |
| Dot Net xx | Community |
| Tortoise Subversion | Open-source Plug-In to Visual Studio |
| SQL Server | Express |
| IIS x.xx | Community |

Table 9 Physical Architecture: PROD-environment client definition

## AS-IS Server Environment details

### AS-IS DEV Server Environment

| **Device:** | **DEV-environment Database & App Server definition** |
| --- | --- |
| Utilisation: | Database & Application Server |
| Location: | DC1 |

| **Device Description** | **Specification** | **Notes** |
| --- | --- | --- |
| Replication Required | No |  |
| Operating System | W2K19SP1 |  |
| Patch level | Current |  |
| Patch method | PatchMyPC |  |
| CPUs | xx |  |
| Memory | xxGB |  |
| NICs | xx |  |
| VM Machine | No |  |
| VM High Availability | N/A |  |
| VM Anti-Affinity Rule | N/A |  |
| VM Affinity Rule | N/A |  |

| **Applications:** | **License:** |
| --- | --- |
| SQL Server 20xx | Enterprise License |

Table 10 Database & App server technical details

| **Device:** | **DEV-environment Web Server Definition** |
| --- | --- |
| Utilisation: | Web Server |
| Location: | DC1 |

| **Device Description** | **Specification** | **Notes** |
| --- | --- | --- |
| Replication Required | No |  |
| Operating System | W2K19SP1 |  |
| Patch level | Current |  |
| Patch method | PatchMyPC |  |
| CPUs | xx |  |
| Memory | xx GB |  |
| NICs | xx |  |
| VM Machine | No |  |
| VM High Availability | N/A |  |
| VM Anti-Affinity Rule | N/A |  |
| VM Affinity Rule | N/A |  |

| **Applications:** | **License:** |
| --- | --- |
| IIS x.xx | Enterprise License |
| DotNet xx | Enterprise License |

Table 11 Web server technical details

### AS-IS UAT/TEST/QA Server Environment

[Follow with the above DEV example]

### AS-IS PROD Server Environment

[Follow with the above DEV example. It is likely that the production environment will have 2 load-balanced web servers]

## AS-IS Storage Architecture

[Provide physical details of the actual storage used to support based on the initial storage requirements and expected annual growth before the storage must be extended, or the anticipated lifetime of this system.]

### AS-IS Storage Overview

This section describes the storage infrastructure in the architecture. Table xx shows a solution overview: Requirement Design Storage Capacity This has not been defined yet and is designed for worst-case - see below. Performance This has not been defined yet and is designed for worst-case: a high-performance tier-1 database storage is provided in case the project requires local search capability. Data Growth This has not been defined yet and is designed for worst-case Archiving There is no archiving in this trial. No requirements defined yet. Data change rate % Anticipated 1% daily change *Table xx Storage Overview*

### AS-IS DEV Environment Storage Architecture

This section describes the storage architecture for the DEV environment.

| **Server Node:** | **PVUKxxxyyyIIS01 (Web server)** |
| --- | --- |

| **Local/SAN** | **OS/Bin/Data/Page** | **FS Type** | **FS Mapping** | **Tier** | **Size (GB)** |
| --- | --- | --- | --- | --- | --- |
| SAN | OS NTFS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Data | NTFS | F-drive | 2 | 32 |

Table 12 DEV Web server storage mounts

| **Server Node:** | **PVUKxxxyyyGIT01 (GIT Version Control server)** |
| --- | --- |

| **Local/SAN** | **OS/Bin/Data/Page** | **FS Type** | **FS Mapping** | **Tier** | **Size (GB)** |
| --- | --- | --- | --- | --- | --- |
| SAN | OS | XFS | / | 2 | 32 |
| SAN | Page | XFS | /swap | 2 | 8 |
| SAN | Binaries | XFS | /opt | 2 | 32 |
| SAN | Data | XFS | /var | 2 | 128 |

Table 13 DEV Version Control server storage mounts

| **Server Node:** | **PVUKxxxyyySQL01 (Database server)** |
| --- | --- |

| **Local/SAN** | **OS/Bin/Data/Page** | **FS Type** | **FS Mapping** | **Tier** | **Size (GB)** |
| --- | --- | --- | --- | --- | --- |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | F-drive | 2 | 128 |
| SAN | DB Log | NTFS | G-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | H-drive | 1 | 32 |
| SAN | DB Backup | NTFS | I-drive | 3 | 128 |
| SAN | Full text Search | NTFS | J-drive | 1 | 32 |

Table 14 DEV Database server storage mounts

### AS-IS UAT Environment Storage Architecture

This section describes the storage architecture for the UAT environment.

| **Server Node:** | **PVUKxxxyyyIIS02 (Web server)** |
| --- | --- |

| **Local/SAN** | **OS/Bin/Data/Page** | **FS Type** | **FS Mapping** | **Tier** | **Size (GB)** |
| --- | --- | --- | --- | --- | --- |
| SAN | OS NTFS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Data | NTFS | F-drive | 2 | 32 |

Table 15 UAT Web server storage mounts

| **Server Node:** | **PVUKxxxyyySQL02 (Database server)** |
| --- | --- |

| **Local/SAN** | **OS/Bin/Data/Page** | **FS Type** | **FS Mapping** | **Tier** | **Size (GB)** |
| --- | --- | --- | --- | --- | --- |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | F-drive | 2 | 128 |
| SAN | DB Log | NTFS | G-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | H-drive | 1 | 32 |
| SAN | DB Backup | NTFS | I-drive | 3 | 128 |
| SAN | Fulltext Search | NTFS | J-drive | 1 | 32 |

Table 16 UAT Database server storage mounts

### AS-IS PROD Environment Storage Architecture

This section describes the storage architecture for the PROD (Production) environment.

| **Server Node:** | **PVUKxxxyyyIIS03 (Web server)** |
| --- | --- |

| **Local/SAN** | **OS/Bin/Data/Page** | **FS Type** | **FS Mapping** | **Tier** | **Size(GB)** |
| --- | --- | --- | --- | --- | --- |
| SAN | OS NTFS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Data | NTFS | F-drive | 2 | 32 |

Table 17 PROD Web server 1 storage mounts

| **Server Node:** | **PVUKxxxyyyIIS04 (Web server)** |
| --- | --- |

| Local/SAN | OS/Bin/Data/Page | FS Type | FS Mapping | Tier | Size(GB) |
| --- | --- | --- | --- | --- | --- |
| SAN | OS NTFS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Data | NTFS | F-drive | 2 | 32 |

Table 18 PROD Web server 2 storage mounts

| **Server Node:** | **PVUKxxxyyySQL03 (Database server)** |
| --- | --- |

| **Local/SAN** | **OS/Bin/Data/Page** | **FS Type** | **FS Mapping** | **Tier** | **Size(GB)** |
| --- | --- | --- | --- | --- | --- |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | F-drive | 2 | 128 |
| SAN | DB Log | NTFS | G-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | H-drive | 1 | 32 |
| SAN | DB Backup | NTFS | I-drive | 3 | 128 |
| SAN | Fulltext Search | NTFS | J-drive | 1 | 32 |

Table 19 PROD Database server storage mounts

## Data Storage Design

This section describes the storage for each of the system environments, which are DEV, QA and PROD:

* Structured data in the relational database:
* Table spaces on the RDBMS, if more than the default
* SAN Data tiering used for each table space on the RDBMS
* Unstructured data (documents, audio, images, video)
* Non-relational databases a.k.a. “no-SQL” databases, e.g. key-value stores, graph databases, document stores, column stores
* File systems for heterogeneous files.
* Amend this table accordingly or cloud storage.

### DEV Environment Database Storage Design

| **Server Node:** | | **PVUKxxxyyySQL01 (DEV Database server)** | | | |
| --- | --- | --- | --- | --- | --- |
| **Local/SAN** | **OS/Bin/Data/Page** | **FS** | **Mapping** | **Tier** | **Size (GB)** |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | G-drive | 2 | 128 |
| SAN | DB Indexes | NTFS | H-drive | 2 | 32 |
| SAN | DB Log | NTFS | I-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | J-drive | 1 | 32 |
| SAN | DB Backup | NTFS | J-drive | 3 | 128 |
| SAN | Full text Search | NTFS | K-drive | 1 | 256 |

Table 20 DEV Database server storage mounts

### QA Environment Database Storage Design

| **Server Node:** | | **PVUKxxxyyySQL02 (QA Database server)** | | | |
| --- | --- | --- | --- | --- | --- |
| **Local/SAN** | **OS/Bin/Data/Page** | **FS** | **Mapping** | **Tier** | **Size (GB)** |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | G-drive | 2 | 128 |
| SAN | DB Indexes | NTFS | H-drive | 2 | 32 |
| SAN | DB Log | NTFS | I-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | J-drive | 1 | 32 |
| SAN | DB Backup | NTFS | J-drive | 3 | 128 |
| SAN | Full text Search | NTFS | K-drive | 1 | 256 |

Table 21 QA Database server storage mounts

### PROD Environment Database Storage Design

| **Server Node:** | | **PVUKxxxyyySQL03 (PROD Database server)** | | | |
| --- | --- | --- | --- | --- | --- |
| **Local/SAN** | **OS/Bin/Data/Page** | **FS** | **Mapping** | **Tier** | **Size (GB)** |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | G-drive | 2 | 128 |
| SAN | DB Indexes | NTFS | H-drive | 2 | 32 |
| SAN | DB Log | NTFS | I-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | J-drive | 1 | 32 |
| SAN | DB Backup | NTFS | J-drive | 3 | 128 |
| SAN | Full text Search | NTFS | K-drive | 1 | 256 |

Table 22 PROD Database server storage mounts

# AS-IS Network Architecture

Show the AS-IS devices, how they are connected to the various networks in data centres, locations and clouds, and the information flow of the most pertinent types of transactions. Indicate what normal network traffic in production use is, and what support traffic is. Also indicate VNets and the network resilience design.

## AS-IS Client Network Details

### DEV Environment

The DEV environment is used for developing and unit-testing of incremental releases.

| **Client Id** | **Use Case** | **IP** |
| --- | --- | --- |
| # | [test case set 1] | [aaa.bbb.ccc.ddd] |
| # | [test case set 2] | [aaa.bbb.ccc.eee] |
| # | [test case set 3] | [aaa.bbb.ccc.fff] |
| # | [test case set 4] | [aaa.bbb.ccc.ggg] |

Table 23 DEV Client IP Configurations, specific details

| **Network Setting** | **Value** |
| --- | --- |
| Gateway | x.x.x.x |
| Mask | x.x.x.x |
| DNS1 | x.x.x.x |
| DNS2 | x.x.x.x |

Table 24 DEVt Client IP Configuration – common details

### UAT Environment

The UAT environment is used for UAT-testing and End-to-End testing of incremental releases.

| **Client Id** | **Use Case** | **IP** |
| --- | --- | --- |
| # | [test case set 1] | [aaa.bbb.ccc.ddd] |
| # | [test case set 2] | [aaa.bbb.ccc.eee] |
| # | [test case set 3] | [aaa.bbb.ccc.fff] |
| # | [test case set 4] | [aaa.bbb.ccc.ggg] |

Table 25 DEV Client IP Configurations, specific details

| **Network Setting** | **Value** |
| --- | --- |
| Gateway | x.x.x.x |
| Mask | x.x.x.x |
| DNS1 | x.x.x.x |
| DNS2 | x.x.x.x |

Table 26 DEVt Client IP Configuration – common details

### PROD Environment

The PROD environment is used by all other business users for running the business.

| **Client Id** | **Use Case** | **IP** |
| --- | --- | --- |
| # | [test case set 1] | [aaa.bbb.ccc.ddd] |
| # | [test case set 2] | [aaa.bbb.ccc.eee] |
| # | [test case set 3] | [aaa.bbb.ccc.fff] |
| # | [test case set 4] | [aaa.bbb.ccc.ggg] |

Table 27 DEV Client IP Configurations, specific details

| **Network Setting** | **Value** |
| --- | --- |
| Gateway | x.x.x.x |
| Mask | x.x.x.x |
| DNS1 | x.x.x.x |
| DNS2 | x.x.x.x |

Table 28 DEVt Client IP Configuration – common details

## Server Network Connection Details

### Network settings common to all environments

The following network settings are shared by all environments:

| Network Device | Description | IP Address |
| --- | --- | --- |
| DNS |  | 17#####03 |
| DNS |  | 17#####03 |
| Citrix Terminal Server |  | 17######7 |
| Gateway Server | See Table xx for details |  |

Table 29 Server IP Configurations - Common

### DEV Environment

| **Server Type** | **DNS Name** | **IP** |
| --- | --- | --- |
| DB | PVUKDEVKSKDB01.[DOMAIN].[Company].ORG | 17####8##1/28 |
| APP | PVUKDEVKSKAPP01.[DOMAIN].[Company].ORG | 17####8##2/28 |
| WEB | PVUKDEVKSKIIS01.[DOMAIN].[Company].ORG | 17####8##3/27 |

Table 30 Server IP Configuration – DEV Server-specific

| **Server Type** | **IP** | **Mask** | **GW** | **VLAN** |
| --- | --- | --- | --- | --- |
| DB | 17####7##80 | 25##5##5##/40 | 17####7##77 | 343 |
| APP | 17####7##64 | 25##5##5##/40 | 17####7##61 | 342 |
| WEB | 17####7##37 | 25##5##5##/24 | 17####7##34 | 341 |

Table 31 DEV Server Gateways and VLAN

### UAT Environment

| **Server Type** | **DNS Name** | **IP** |
| --- | --- | --- |
| DB | PVUKUATKSKDB01.[DOMAIN].[Company].ORG | 17####8##1/28 |
| APP | PVUKUATKSKAPP01.[DOMAIN].[Company].ORG | 17####8##2/28 |
| WEB | PVUKUATKSKIIS01.[DOMAIN].[Company].ORG | 17####8##3/27 |

Table 32 Server IP Configuration – UAT Server-specific

| **Server Type** | **IP** | **Mask** | **GW** | **VLAN** |
| --- | --- | --- | --- | --- |
| DB | 17####7##80 | 25##5##5##/40 | 17####7##77 | 344 |
| APP | 17####7##64 | 25##5##5##/40 | 17####7##61 | 345 |
| WEB | 17####7##37 | 25##5##5##/24 | 17####7##34 | 346 |

Table 33 UAT Server Gateways and VLAN

### PRODUCTION Environment

There are 2 load-balanced webservers. See Table 20 below for details on load balancing.

| **Server Type** | **DNS Name** | **IP** |
| --- | --- | --- |
| DB | PVUKDEVKSKDB01.[DOMAIN].[Company].ORG | 17####8##1/28 |
| APP | PVUKDEVKSKAPP01.[DOMAIN].[Company].ORG | 17####8##2/28 |
| WEB | PVUKDEVKSKIIS01.[DOMAIN].[Company].ORG | 17####8##3/27 |
| WEB | PVUKDEVKSKIIS02.[DOMAIN].[Company].ORG | 17####8##4/27 |

Table 34 Server IP Configuration – PROD Server-specific

| **Server Type** | **IP** | **Mask** | **GW** | **VLAN** |
| --- | --- | --- | --- | --- |
| DB | 17####7##80 | 25##5##5##/40 | 17####7##77 | 347 |
| APP | 17####7##64 | 25##5##5##/40 | 17####7##61 | 348 |
| WEB | 17####7##37 | 25##5##5##/24 | 17####7##34 | 349 |

Table 35 PROD Server Gateways and VLAN

## Client Network Connection Details

### Configuration of Clients on the Network

| **Client location** | **Name** | **Client Type** | **IP Address** | **Format** | **Patch Info** |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table 36 Client IP Configuration - Specific settings

| **Network Setting** | **Value** |
| --- | --- |
| Gateway | x.x.x.x |
| Mask | x.x.x.x |
| DNS1 | x.x.x.x |
| DNS2 | x.x.x.x |
| VLAN | 410 |
| Proxy | sip-sysapp.ukdmz |
| Proxy Port | 8080 |
| Proxy user | [PROJECT]-allocated ID |
| Proxy password | [PROJECT]-generated |

Table 37 Client IP Configuration – Common settings

| **Network Setting** | **Value** |
| --- | --- |
| DNS1 Helper IP | 17#####0 |
| DNS2 Helper IP | 17#####1 |
| VLAN | 410 |

Table 38 Remote Site Router Configuration

## AS-IS Hardware Load Balancing

### AS-IS DEV Environment Hardware load balancing

Example: There is no hardware load balancing in this environment.

### AS-IS UAT Environment Hardware load balancing

Example: There is no hardware load balancing in this environment.

### AS-IS PROD Environment Hardware load balancing

Example: There is hardware load balancing between the two web servers in this environment. This is achieved on with an F5 network load balancer – see details below.

| **Node** | **HLB Node** | **Port** | **Monitor Method** | **Server down landing page** |
| --- | --- | --- | --- | --- |
| PVUKxxxyyyIIS05 | PVUKxxxyyyIIS03 | 80 | Ping & HTTP | Get every 5 seconds, timeout 16 seconds |
|  | PVUKxxxyyyIIS04 | 80 | Ping & HTTP Get every 5 seconds, timeout 16 seconds | TBA |

Table 39 PROD Hardware Load Balancing

### AS-IS Network Traffic – Client

Figure x: Client network traffic steps between network components

| **Step** | **Description** | **Type** | **Impact** |
| --- | --- | --- | --- |
| 1 | Cloud applications | HTTPS | Minimal |
| 2 | Database | TCP/IP | Minimal |

Table 40 Client network traffic Steps

### AS-IS Network Traffic – Operational and Support

Figure x: Operational and Support network traffic Steps

| Step | Description | Type | Impact |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |

Table 41 Operational and Support network traffic Steps

### AS-IS Network Resilience

There is no network resilience.

# AS-IS Error Architecture

[Describe error handling, logging, notifications, error severities, integration with the estate’s operational monitoring system]

# AS-IS Security Architecture

Describe pertinent security features, such as encryption on the move and at rest, authentication mechanisms, access-controlled roles and allowed operations.

# AS-IS Post-Go-Live Architecture

Describe the support system in place for this solution, and contact details to first-line support of the respective vendors involved

# TO-BE Contextual Architecture

[Simple description of the business context under which the solution currently operates. If there is a difference between the AS-IS and the TO-BE contextual views, this must be clearly highlighted here.]

The business context of this solution has not changed. However, it has been redesigned to improve efficiency and scalable performance.

To recap, the contextual business model is shown below and describes the business activities that meet the business requirements [Ref. 1].

[Instead of using Visio, you can use this Mermaid.js code in Visual Studio or mermaidchart.com for this illustration:]

flowchart LR

%% Nodes for the Order Fulfillment Process

A[("fab:fa-cart-plus Order Received")]

B["fa:fa-box Inventory Check"]

C["fa:fa-clipboard-list Order Processing"]

D{"Is Item Available?"}

E["fa:fa-shipping-fast Shipping"]

F["fa:fa-truck Dispatch"]

G[("fa:fa-file-invoice Generate Invoice")]

H["fa:fa-envelope Customer Notification"]

I["fa:fa-box-open Order Packed"]

J[("fa:fa-check-circle Order Fulfilled")]

%% Edge connections between nodes

A --> B --> C --> D

D -- Yes --> E --> I --> F --> J

D -- No --> G -- Notify --> H

%% Styling for specific nodes

style A fill:#dddddd

style G fill:#dddddd

style J fill:#dddddd

flowchart LR

    A(Start) --> B{"Validate Payment Info"}

    B -- Valid --> C[Payment Gateway]

    B -- Invalid --> G[Payment Failure]

    C --> D["Transaction Processing"]

    D -- Approved --> E[Confirm Payment]

    D -- Denied --> G[Payment Failure]

    E --> F(End)

    G --> F

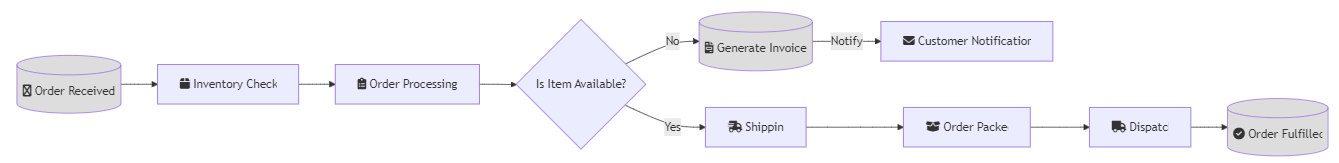


Figure 8 AS-IS Contextual architecture diagram: Customer order fulfilment

A diagram of a computer

Description automatically generated

Figure 9 AS-IS Contextual architecture: Customer order payment

# TO-BE Conceptual Architecture

Show and explain the current user operation in a workflow diagram. Swim lane diagrams with business area or system in each lane. Indicate the changes if any between the AS-IS and the TO-BE views.

Figure: TO-BE Conceptual use case for ...

# TO-BE Logical Architecture

*[Show the application and computational components, human interface components and storages currently in use. Show in a figure. Indicate the changes if any between the AS-IS and the TO-BE versions]*

Figure: Logical architecture of the client solution

## System Design Pattern

The following general reusable solutions approaches were used

*[Explain the use of the (new|inherited) distributed system design pattern, if any, and how the chosen patterns measures up in terms of software engineering effort, computer capabilities, system availability and business risk amelioration.]*

* Classic Patterns
* Layered: Presentation, Application, Business and Data. Good for simple implementations
* Pipe-filter:
* Client-Server:
* Model/View/Controller: Data, presentation and user interaction
* Event bus: Publisher / Subscriber approach
* Microserices: API interfaces to services
* Broker: Coordinates communication between components
* Blackboard:
* Master/Slave: Work distribution to slaves, master collects the result. Only for certain problems that can be suitable decomposed, e.g. mathematics or database replications.
* Distributed Systems
* Peer to peer: Peer is both client and server. No guarantee of performance, poor security, but highly robust.
* Distributed Systems
* Ambassador
* Circuit Breaker
* CQRS - Command Query Responsibility Segregation
* Event Sourcing
* Leader Election
* Publisher / Subscriber
* Sharding
* Strangler Fig Tree - gradually replacing components

# TO-BE Component Definitions

[Explanation of the above figure, with brief summary of the technologies involved. Indicate the new or changed components.]

Definition Vendor Explanation

*Table xx Component definitions and explanations*

## Client technology

[Explanation with respect to the above figure, e.g. min browser spec, OS, if different to the AS-IS ]

## Server technology

[Explanation with respect to the above figure, if different to the AS-IS]

## Support technology

[Describe how the TO-BE is supported if different to the AS-IS]

# TO-BE Technical Environments

[Describe the technical environments that will be deployed, if they are different to the AS-IS. If they will be torn down after the development or deployment, indicate this. Example: 3 environments. Normally QA, TEST and UAT are separate environments]

[Describe the technical environments that are used to produce the AS-IS design, or are currently supporting the AS-IS design. If they were torn down after the development or deployment, indicate this. Example Options:

1. The DEV, UAT environments will be reused for the development and test, The existing PRODUCTION environment will be reused on cut-over.
2. New environments, in which case these need to be specified at the same level of details as the AS-IS.]

Environment Function Status DEV Development Active UAT/TEST/QA Test,QA,User Acceptance Active PROD Production Active *Table xx Summary of TO-BE technical environments*

## TO-BE DEV Environment

## TO-BE UAT Environment

## TO-BE PROD Environment

## TO-BE Client environment

[Describe the technical aspects of the client – usually a device that support a thin-client interface such as a browser]

Item Specification Device [Company]-Standard laptop Operating system Windows 7 Inputs Mouse, keyboard, touchscreen, headset Outputs Graphical screen, printer Programs Edge Browser Version 9###10##6 (Official build) (64-bit) *Table xx AS-IS Client specification*

## TO-BE Client Environments details

## TO-BE DEV Client Environment

The design of the DEV client environment includes capacity for hosting the development of other development projects. One development workstation is required per developer.

Utilisation: Development Workstation, 1 off per developer Location: Development Centre Device Description Specification Notes Replication Required No Operating System Win11 Patch level Current Patch method MS SCCM 20xx CPUs xx Memory xxGB NICs xx VM Machine No VM High Availability N/A VM Anti-Affinity Rule N/A VM Affinity Rule N/A Applications: Visual Studio 20xx Enterprise License DotNet xx GIT client Open-source Plug-In to Visual Studio SQL Server Express IIS x.xx *Table xx Physical Architecture: DEV-environment client definition*

## TO-BE UAT Client Environment

The design of the UAT client environment includes capacity for hosting the application and other standard, peripheral support applications for communications and documentation. UAT testing will be conducted in a secure test environment by testers. Testing will commence following user training and completion of the relevant test plans.

Utilisation: Standard issue laptop, 1 off per user Location: Secure Testing Centre Device Description Specification Notes Replication Required No Operating System Win11 Patch level Current Patch method MS SCCM 20xx CPUs xx Memory xxGB NICs xx VM Machine No VM High Availability N/A VM Anti-Affinity Rule N/A VM Affinity Rule N/A Applications: Standard apps Enterprise License GIT client Open-source Plug-In to Visual Studio Test Console Test management application Test system System under test with apps and libs *Table xx Physical Architecture: UAT-environment client definition*

## TO-BE PROD Client Environment

The design of the production client environment includes capacity for hosting the application and other standard, peripheral support applications for communications and documentation.

Utilisation: Standard issue laptop, 1 off per user Location: Secure Operations Centre Device Description Specification Notes Replication Required No Operating System Win11 Patch level Current Patch method MS SCCM 20xx CPUs xx Memory xxGB NICs xx VM Machine No VM High Availability N/A VM Anti-Affinity Rule N/A VM Affinity Rule N/A Applications: Standard apps Enterprise License Test system System under test with apps and libs *Table xx Physical Architecture: PROD-environment client definition*

## TO-BE Server Environments details

## TO-BE DEV Server Environment

This section lists the servers in the DEV environment. The GIT code repository service is a shared service between other projects.

| **Server Criteria** | **Value** |
| --- | --- |
| Utilisation: | Database Server |
| Location: | Development Centre |
| Device Description | Specification Notes |
| Replication Required | No |
| Operating System | W2K19SP1 |
| Patch level | Current |
| Patch method | MS SCCM 20xx |
| CPUs | xx |
| Memory | xxGB |
| NICs | xx |
| VM Machine | Yes |
| VM High Availability | No |
| VM Anti-Affinity Rule | No |
| VM Affinity Rule | No |
| Applications: | SQL Server 20xx Enterprise License |

Table 42 Database server technical details

| **Server Criteria** | **Value** |
| --- | --- |
| Utilisation: | Web server |
| Location: | Development Centre |
| Device Description Specification | Notes |
| Replication Required | No |
| Operating System | W2K19SP1 |
| Patch level | Current |
| Patch method | MS SCCM 20xx |
| CPUs | xx |
| Memory | xxGB |
| NICs | xx |
| VM Machine | Yes |
| VM High Availability | No |
| VM Anti-Affinity Rule | No |
| VM Affinity Rule | No |
| Applications: | IIS x.xx Enterprise License, DotNet xx |

Table 43 Web server technical details

| **Server Criteria** | **Value** |
| --- | --- |
| Utilisation: | Version Control Server |
| Location: | Development Centre |
| Device Description Specification | Notes |
| Replication Required | No |
| Operating System | RedHat 9.x |
| Patch level | Current |
| Patch method | Red Hat YUM |
| CPUs | xx |
| Memory | xxGB |
| NICs | xx |
| VM Machine | No |
| VM High Availability | N/A |
| VM Anti-Affinity Rule | N/A |
| VM Affinity Rule | N/A |
| Applications: | GIT Open-source License, Apache Server |

Table 44 GIT Version Control server technical details

## TO-BE UAT/TEST/QA Server Environment

*[Follow the above DEV example]*

## TO-BE PROD Server Environment

*[Follow the above DEV example. It is likely that the production environment will have 2 load-balanced web servers]*

## TO-BE Storage Architecture

## TO-BE Storage Overview

This section describes the storage infrastructure in the architecture. Table xx shows a solution overview: Requirement Design Storage Capacity This has not been defined yet and is designed for worst-case - see below. Performance This has not been defined yet and is designed for worst-case: a high-performance tier-1 database storage is provided in case the project requires local search capability. Data Growth This has not been defined yet and is designed for worst-case Archiving There is no archiving in this trial. No requirements defined yet. Data change rate % Anticipated 1% daily change *Table xx Storage Overview*

* **TO-BE DEV Environment Storage Architecture**

This section describes the storage architecture for the DEV environment. Server Node: PVUKxxxyyyIIS01 (Web server) Local/SAN OS/Bin/Data/Page FS Mapping Tier Size(GB) SAN OS NTFS C-drive 2 32 SAN Page NTFS D-drive 2 8 SAN Binaries NTFS E-drive 2 32 SAN Data NTFS F-drive 2 32 *Table xx DEV Web server storage mounts*

Server Node: PVUKxxxyyyGIT01 (GIT Version Control server) Local/SAN OS/Bin/Data/Page FS Mapping Tier Size(GB) SAN OS XFS / 2 32 SAN Page XFS /swap 2 8 SAN Binaries XFS /opt 2 32 SAN Data XFS /var 2 128 *Table xx DEV Version Control server storage mounts*

Server Node: PVUKxxxyyySQL01 (Database server) Local/SAN OS/Bin/Data/Page FS Mapping Tier Size(GB) SAN OS NTFS C-drive 2 32 SAN Page NTFS D-drive 2 8 SAN Binaries NTFS E-drive 2 32 SAN Application Data NTFS F-drive 2 32 SAN DB Data NTFS F-drive 2 128 SAN DB Log NTFS G-drive 1 32 SAN DB TempDB NTFS H-drive 1 32 SAN DB Backup NTFS I-drive 3 128 SAN Fulltext Search NTFS J-drive 1 32 *Table xx DEV Database server storage mounts*

* **TO-BE UAT Environment Storage Architecture**

This section describes the storage architecture for the UAT environment. Server Node: PVUKxxxyyyIIS02 (Web server) Local/SAN OS/Bin/Data/Page FS Mapping Tier Size(GB) SAN OS NTFS C-drive 2 32 SAN Page NTFS D-drive 2 8 SAN Binaries NTFS E-drive 2 32 SAN Data NTFS F-drive 2 32 *Table xx UAT Web server storage mounts*

Server Node: PVUKxxxyyySQL02 (Database server) Local/SAN OS/Bin/Data/Page FS Mapping Tier Size(GB) SAN OS NTFS C-drive 2 32 SAN Page NTFS D-drive 2 8 SAN Binaries NTFS E-drive 2 32 SAN Application Data NTFS F-drive 2 32 SAN DB Data NTFS F-drive 2 128 SAN DB Log NTFS G-drive 1 32 SAN DB TempDB NTFS H-drive 1 32 SAN DB Backup NTFS I-drive 3 128 SAN Fulltext Search NTFS J-drive 1 32 *Table xx UAT Database server storage mounts*

## TO-BE PROD Environment Storage Architecture

This section describes the storage architecture for the PROD (Production) environment.

Server Node: PVUKxxxyyyIIS03 (Web server) Local/SAN OS/Bin/Data/Page FS Mapping Tier Size(GB) SAN OS NTFS C-drive 2 32 SAN Page NTFS D-drive 2 8 SAN Binaries NTFS E-drive 2 32 SAN Data NTFS F-drive 2 32 *Table xx PROD Web server 1 storage mounts*

Server Node: PVUKxxxyyyIIS04 (Web server) Local/SAN OS/Bin/Data/Page FS Mapping Tier Size(GB) SAN OS NTFS C-drive 2 32 SAN Page NTFS D-drive 2 8 SAN Binaries NTFS E-drive 2 32 SAN Data NTFS F-drive 2 32 *Table xx PROD Web server 2 storage mounts*

Server Node: PVUKxxxyyySQL03 (Database server) Local/SAN OS/Bin/Data/Page FS Mapping Tier Size(GB) SAN OS NTFS C-drive 2 32 SAN Page NTFS D-drive 2 8 SAN Binaries NTFS E-drive 2 32 SAN Application Data NTFS F-drive 2 32 SAN DB Data NTFS F-drive 2 128 SAN DB Log NTFS G-drive 1 32 SAN DB TempDB NTFS H-drive 1 32 SAN DB Backup NTFS I-drive 3 128 SAN Fulltext Search NTFS J-drive 1 32 *Table xx PROD Database server storage mounts*

## TO-BE Application Architecture

[Describe the applications that makeup the solution and their respective components in detail, and show their interfaces and how the end result of the business requirements are achieved.]

### [AppName1] Overview

Figure 2 [Example application and its components]

[Example: Figure 2 is a Vendor-supplied representation of the logical architecture of the client-side application and applies to the product. The data flow on the client application is detailed in Figure xx - also a Vendor-supplied representation.] Component Description Presentation layer Business layer Integration layer System support layer Device Executive Supervisor Monitor Downloader Event Logger X

*Table xx Application Components for client*

### [AppName2] Overview

[Rinse and repeat for every application in the solution]

# TO-BE Error Architecture

[Describe error handling, logging, notifications, error severities, integration with the estate’s operational monitoring system]

# TO-BE Network Architecture

[Show the TO-BE devices, how they are connected to the various networks in data centres, locations and clouds, and the information flow of the most pertinent types of transactions. Indicate what is normal network traffic in production use, and what is support traffic. Also indicate VNets and the network resilience design.]

## Network Architecture Overview

Network Design Required Reason Load Balancing Yes Required for webservers, although not currently used. SSL Offloading No No encrypted traffic Network Security Yes New client devices served by data-centre servers Network Performance No Not anticipated Network Resilience Yes Datacentre internal network resilience. Part of [Company] System standard pattern. Non group-compliant IP Addressing No Use group-compliant IP addressing scheme for client devices and data centre devices *Table xx Network Architecture Overview*

* **Network Traffic**

## Network Traffic – Batch data update

[Example if batches are run:] Figure xx shows the network traffic for daily data updates from the [Company] System data back-end to the client solution. This batch is scheduled to run daily at 06:00. Data take-on to all the client devices is completed by 07:00. The total data load per client device is in the region of 200K. This data is eventually distributed to each client.

Figure: Network flows for batch jobs

* **Flow Route Purpose Type Frequency Volume Data centre AbInitio server Initiates data extraction process Daily 5MB Data centre Reference data update ETL process Daily 5MB *Table xx Network traffic flows - Batch data update* No potential network bottlenecks are anticipated. Risk Network Route Impact**

*Table xx Potential bottlenecks - Batch data update*

## Network Traffic – Operational and Support

Operational and support traffic is anticipated to exist during standard operating hours in the standard location

*Figure xx shows the network traffic.*

*Figure: Network flows for batch jobs*

| **Flow** | **Route** | **Purpose** | **Type** | **Network** | **Frequency** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Table 45 Network traffic flows - Batch data update

No potential network bottlenecks are anticipated.

| **Risk** | **Network** | **Route** | **Impact** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Table 46 Potential bottlenecks - Operational support

## Network Traffic – Part Search and Cross reference

*Figure xx shows the network traffic.*

*Figure: Network flows for batch jobs*

| **Flow** | **Route** | **Purpose** | **Type** | **Network** | **Frequency** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Table 47 Network traffic flows – Part Search and cross reference

* **No potential network bottlenecks are anticipated. Risk Network Route Impact**

*Table xx Potential bottlenecks - Part Search and cross reference*

**Network Traffic – Direct Client**

Figure xx shows the client network traffic.

Figure: Network flows for client network traffic.

| **Flow** | **Route** | **Purpose** | **Type** | **Network** | **Frequency** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Table 48 Network traffic flows – client network traffic.

* **No potential network bottlenecks are anticipated. Risk Network Route Impact**

*Table xx Potential bottlenecks - Client network traffic.*

## Network Resilience

[Example if the solution has network resilience] The servers on all the environments are fully resilient. This is conceptually explained below in Figure xx.

Figure: Conceptual network resilience by interconnecting switches, firewalls and load balancers

## Server Network Connection Details

**Network settings common to all environments**

| **Network Device** | **Description** | **IP Address** |
| --- | --- | --- |
| DNS |  | 17#####03 |
| DNS |  | 17#####03 |
| Citrix Terminal Server |  | 17######7 |
| Gateway Server | See Table xx for details |  |

Table 49 Server IP Configurations - Common

**DEV Environment**

| **Server Type** | **DNS Name** | **IP** |
| --- | --- | --- |
| DB | PVUKDEVKSKDB01.[DOMAIN].[Company].ORG | New |
| APP | PVUKDEVKSKAPP01.[DOMAIN].[Company].ORG | New |
| WEB | PVUKDEVKSKIIS01.[DOMAIN].[Company].ORG | New |

Table 50 Server IP Configuration – DEV Server-specific

| **Server Type** | **IP** | **Mask** | **GW** | **VLAN** |
| --- | --- | --- | --- | --- |
| DB | 17####7##80 | 25##5##5##/40 | 17####7##77 | 343 |
| APP | 17####7##64 | 25##5##5##/40 | 17####7##61 | 342 |
| WEB | 17####7##37 | 25##5##5##/24 | 17####7##34 | 341 |

Table 51 DEV Server Gateways and VLAN

## UAT Environment

| **Server Type** | **DNS Name** | **IP** |
| --- | --- | --- |
| DB | PVUKUATKSKDB01.[DOMAIN].[Company].ORG | New |
| APP | PVUKUATKSKAPP01.[DOMAIN].[Company].ORG | New |
| WEB | PVUKUATKSKIIS01.[DOMAIN].[Company].ORG | New |

Table 52 Server IP Configuration – UAT Server-specific

| **Server Type** | **IP** | **Mask** | **GW** | **VLAN** |
| --- | --- | --- | --- | --- |
| DB | 17####7##80 | 25##5##5##/40 | 17####7##77 | 343 |
| APP | 17####7##64 | 25##5##5##/40 | 17####7##61 | 342 |
| WEB | 17####7##37 | 25##5##5##/24 | 17####7##34 | 341 |

Table 53 UAT Server Gateways and VLAN

**PRODUCTION Environment**

| **Server Type** | **DNS Name** | **IP** |
| --- | --- | --- |
| DB | PVUKPRDKSKDB01.[DOMAIN].[Company].ORG | New |
| APP | PVUKPRDKSKAPP01.[DOMAIN].[Company].ORG | New |
| WEB | PVUKPRDKSKIIS01.[DOMAIN].[Company].ORG | New |

Table 54 Server IP Configuration – PROD Server-specific

| **Server Type** | **IP** | **Mask** | **GW** | **VLAN** |
| --- | --- | --- | --- | --- |
| DB | 17####7##80 | 25##5##5##/40 | 17####7##77 | 343 |
| APP | 17####7##64 | 25##5##5##/40 | 17####7##61 | 342 |
| WEB | 17####7##37 | 25##5##5##/24 | 17####7##34 | 341 |

Table 55 PROD Server Gateways and VLAN

## Client Network Connection Details

**Configuration of Clients on the Network**

| Client location | Name | Client Type | IP Address | Format | Patch Info |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table 56 Client IP Configuration - Specific settings

| **Network Setting** | **Value** |
| --- | --- |
| Gateway | x.x.x.x |
| Mask | x.x.x.x |
| DNS1 | x.x.x.x |
| DNS2 | x.x.x.x |
| VLAN | 410 |
| Proxy | sip-sysapp.ukdmz |
| Proxy Port | 8080 |
| Proxy user | [PROJECT]-allocated ID |
| Proxy password | [PROJECT]-generated |

Table 57 Client IP Configuration – Common settings

| **Network Setting** | **Value** |
| --- | --- |
| DNS1 Helper IP | 17#####0 |
| DNS2 Helper IP | 17#####1 |
| VLAN | 410 |

Table 58 Remote Site Router Configuration

# TO-BE Network Services

## Network File Shares

[Example given below:]

### DEV Environment

| **Share** | **Shared from** | **Access** |
| --- | --- | --- |
| \PVUKDEVSKAPP01.[DOMAIN].[Company].ORG[project]\in | F:\Data[project]\in | Everyone |
| \PVUKDEVSKAPP01.[DOMAIN].[Company].ORG[project]\out | F:\Data[project]\out | Everyone |

Table 59 Network File shares on DEV environment

### UAT Environment

| **Share** | **Shared from** | **Access** |
| --- | --- | --- |
| \PVUKTSTKSKAPP01.[DOMAIN].[Company].ORG[project]\in | F:\Data[project]\in | Everyone |
| \PVUKTSTKSKAPP01.[DOMAIN].[Company].ORG[project]\out | F:\Data[project]\out | Everyone |

Table 60 Network File shares on UAT environment

### PRODUCTION Environment

| **Share** | **Shared from** | **Access** |
| --- | --- | --- |
| \PVUKPRDKSKAPP01.[DOMAIN].[Company].ORG[project]\in | F:\Data[project]\in | Everyone |
| \PVUKPRDKSKAPP01.[DOMAIN].[Company].ORG[project]\out | F:\Data[project]\out | Everyone |

Table 61 Network File shares on PROD environment

Network services that are consumed by the servers and the client devices, as detailed below:

| **Service** | **Reason** | **Details** |
| --- | --- | --- |
| DNS | Resilience and flexibility and future DR | [TBA], [TBA] |
| DHCP |  |  |
| WINS |  |  |
| NTP | Clock sync. | TBA |
| AD | IAM | Domain: XXXX |
| SMTP | SSRS report notification | TBA. Source email: noreply@[Company] System.com |

Table 62 Server-side Network Services

| **Service** | **Reason** | **Details** |
| --- | --- | --- |
| DNS | Resilience and flexibility and future DR | [TBA], [TBA] |
| DHCP |  |  |
| WINS |  |  |
| NTP | Clock sync. | TBA |
| AD | IAM | Domain: XXXX |
| SMTP |  |  |

Table 63 Client Network Services

## Network Infrastructure

This section details the resulting network infrastructure

## Conceptual network infrastructure

[Show where the solution is located on the various VLANs and VNETs]

## Logical network view

[Heavy-duty network implementations views if the project requires this.] ##0 Physical network view [Heavy-duty network implementations views if the project requires this.]

## Switch-configuration context

[Heavy-duty network implementations views if the project requires this.]

# TO-BE Security Architecture

[Describe pertinent security features, such as encryption on the move and at rest, authentication mechanisms, access-controlled roles and allowed operations, requirements for firewall, security zones, and interfaces across these zones.]

## Security Overview

[This section explains how security and compliance is implemented for the solution.]

| **Requirement** | **Implementation** |
| --- | --- |
| Data Sensitivity Classification | Anonymous and not commercially sensitive |
| Business impact if information is disclosed | Negligible |
| Business impact if data is changed by adversary? | Significant |
| Implementation Approach? | Third-party application |
| Security operations | [Brand-X] AV on each client and server |
| Logging of security events & data | Windows Event Logger, stored indeterminately |

Table 64 Data Compliance

| **Requirement** | **Implementation** |
| --- | --- |
| Electrical compliance | Yes |
| Desktop Support Compliance | Yes |
| Waste compliance | No |
| Data Protection Act | No |
| PCI DSS | No |
| Distance Selling Act | No |
| Disability Discrimination | No |

Table 65 Legal Compliance

| **Requirement** | **Implementation** |
| --- | --- |
| How will users authenticate to the system? | There is no user authentication on client devices. All users are treated as anonymous. |
| Is Active Directory to be used for authentication | No, as there is no user authentication. |
| How will users gain authorisation to the system and data? | Users are not required to view system data. |
| Are passwords transferred over the network in clear text? | No, as there is no user authentication. |
| Highlight any part of the system not based on the least privilege model? | None. |

Table 66 Authentication, Authorisation and Access Control

## Data Security

[Describe how the following aspects were incorporated into the design:]

* Data is encrypted at rest
* Data is encrypted in transit
* Access control to data
* Data held to ransom mitigation
* Password implementation
* Data integrity monitoring

## System Security

\*[Describe how the following aspects were incorporated into the design:]

* How software vulnerabilities are mitigated
* Operating system integrity monitoring

## Security Approaches

[Describe how the following aspects were incorporated into the design:]

**Perimeter Security Approach**

**Traditional zone layers**

**Zero-Trust Approach**

Treat every component, service and user as continuosly-exposed to hostile security threats.

* Never trust, always verify
* Implement the least privilege
* "Assume breach". How is the system design to aid quick recovery from break, and to minimize the attack surface?

## Overall System Security Operations

*[How are the following operations ]*

Protect: Careful public information, social media, server error messages, disable unused ports & services, use honeypot decoys, use firewalls, user awareness, DKIM & SPF email validation, DNS filtering, Web filtering, network segmentation Detect: Rootkit checker, config checker, SSL deep packet inspection Respond: Sandbox Recover: SOP for such an event. Restore or Reimage.

## Security Zones

Figure xx shows the solution’s security zones and trust boundaries that are implemented by means of firewalls and switches and corresponding network policies. No confidential data is transmitted between the security zones. The zones are: Id Security Zone Description A Un-trusted External Public network B Un-trusted Internal DMZ Hosts the access gateway with strong policies C Trusted Internal DMZ Citrix server farm for access servers in the Trusted Internal zones D Semi-trusted Internal: Location network User network with restricted policies E Trusted internal: User-facing Web server F Trusted internal: Application-facing Application and database server G Trusted internal: Enterprise Services Product search (and other future services) *Table xx Security Zones*

Figure 2 shows security zones and how they are bound to each other:

## Client Device Security

[Example, if the solution entails a dedicated client:]

### Secure Client Enclosure

The client consists of a barebones-PC and touch screen enclosed in a steel cabinet. When closed, the steel cabinet prevents access to the PC. The enclosure is secured with a barrel lock, of which there are only two keys. Both barrel keys are in the care of the deployment-location manager. The client device does not have a mouse or a keyboard. It is also not possible to plug a mouse or a keyboard into the PC unless access is gained into the enclosure.

### OS build on client devices

*[Example for security of a simple embedded client device:]*

The base OS build for the client devices is one that has been accredited by [Company] System Desktop Support Services. All client devices are based on this build. The client devices are not registered on Active Directory. The build features the following:

• The build is based on the XXXX Operating System. • The client boots up into a default local user account without password authentication. The user account itself is, however, strongly password-protected. This account can only execute client-related processes. • The client has a local administrator account which is password-protected with a strong password. • The client boots up in a local user account and the [PROJECT] Application Launch service is started. This is responsible for launching client applications based on configuration settings. • There is no desktop screen or start bar on the screen. With no client application running, the screen displays a neutral grey. • The only way to invoke any response from the device when no interactive application runs, is through the use of a plugged-in keyboard with the Ctrl-Alt-Del key combination. • No network port restrictions are in place. • The client devices are not part of the XXXX Active Directory (AD) domain. •

## Servers on the network

*[Example:]*

The servers are built according to the standard [Company] System build, which mandates certain levels of security. They are virtual servers hosted on blade server housed in the Data Centre in a cage. Physical access to the data centre and the cage is controlled by security personnel. All servers in all 3 environments are on the XXXXX Active Directory (AD) domain. Remote Desktop Access is enabled and users that belong to the Remote Desktop Group can access the server desktop. All services run as system service accounts.

## Network Security

[Example:] Firewall policies ensure that: • The UAT web server is the only server that can connect to the test client devices. • Test Client devices can only access servers in the UAT environment.

#TO-BE Post-Production Architecture [Describe the support system in place for this solution, contact details to first line support of the respective vendors involved, how to maintain the system, steps to take for self-recovery]

# TO-BE Operational Framework

## Backup and Restore

There is no backup and recovery process in place for the DEV environment. There is limited back-up and recovery process in place for the UAT environment. The PRODUCTION environment has full backup capabilities.

## System recovery

There is no system recovery. This is not a business-critical system.

## Support Model

Support for [component x, component y] is entirely managed by the vendor of these components. The remainder of this solution is supported by [Company] System's support team.

## Application Codebase

The vendor manages the code base for the duration of this trial.

## Backup and Recovery

In lieu of the fact this project is a trial, no backup has been implemented on any of the environments. Requirement Implementation Backup Details No local backups, no off-server backups, no off-site backups Backup Schedule None Backup Durations N/A Backup Retention None Additional backup services None Recovery Services None

*Table xx Backup and recovery design*

## Disaster Recovery

There is no DR implemented for this solution.

## Development and Test Environments

Figure: UAT Testing Environment

# TO-BE Solution Deployment

*[Choose one or more of these possible deployment approaches for deploying the solution into the final Production environment and embellish accordingly. Also describe the technical operation of the deployment to the target devices – will it be Docker, file-copied, packaged in an Installation Package (e.g. an msi file), etc...?]*

## Big-Bang Deployment

A diagram of a software application

Description automatically generated

Figure 9 Big-Bang Deployment

```mermaid

---

title: Big-Bang Deployment

---

graph LR

CA("fa:fa-users<br>Users"):::light

subgraph IN[Server-side]

LB("fa:fa-share-nodes<br>Load<br>Balancer"):::db

S1("fa:fa-gears<br>Application<br>Version 1.0"):::light

S2("fa:fa-gears<br>Application<br>Version 2.0"):::gray

end

CA --> LB

LB --> S1 & S2

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

### Application

* This is often the only choice when an intricate database upgrade is involved

### Characteristics

* Requires downtime
* Downtime needs to be short
* Needs much preparation and testing
* Need to have a robust back-out plan
* Consider data implications in roll-back plan

## Rolling Deployment

A diagram of a computer application

Description automatically generated

Figure 10 Rolling Deployment

```mermaid

---

title: Rolling Deployment

---

graph LR

CA("fa:fa-users<br>Users"):::light

subgraph IN[Server-side]

LB("fa:fa-share-nodes<br>Load<br>Balancer"):::db

subgraph RS[TO-BE Application]

S2("fa:fa-gears<br>Application<br>Version 2.0"):::gray

S3("fa:fa-gears<br>Application<br>Version 1.0"):::light

S4("fa:fa-gears<br>Application<br>Version 1.0"):::light

end

subgraph LS[AS-IS Application]

S1("fa:fa-gears<br>Application<br>Version 1.0"):::light

end

end

CA --> LB

LB --"Before deployment"--> LS

LB --"Phased deployment"--> RS

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

### Application

* When no downtime can be afforded
* Only expose a small part of the new system at any time
* Balances Risk and User impact in a controlled way

### Characteristics

* Allows incremental updates of parts of the system
* Deploy to selected servers, user groups, geographical areas
* Repeat until entire system is updated
* Detect anomolies early on and mitigate with roll-back on affected geographies
* Reduces risk of system-wide issues
* Slower deployment process
* Prevents downtime

## Blue-Green Deployment

A diagram of a software application

Description automatically generated

Figure 11 Blue-Green Deployment

```mermaid

---

title: Blue-Green Deployment

---

graph LR

CA("fa:fa-users<br>Users"):::light

QA("fa:fa-users<br>QA Testers"):::light

subgraph IN[Server-side]

LB("fa:fa-share-nodes<br>Load<br>Balancer"):::db

subgraph BLUE[Active environment]

S3("fa:fa-gears<br>Application<br>Version 1.0"):::gray

end

subgraph GREEN[Idle environment]

S1("fa:fa-gears<br>Application<br>Version 1.1"):::gray

end

end

GREEN:::green

BLUE:::blue

QA --> LB

CA --> LB

LB --"Users"--> BLUE

LB --"QA Testers"--> GREEN

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef blue fill:#99f,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef green fill:#9f9,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

### Application

* When incremental roll-out is required
* Roll-out to all user communities / business departments needs to be tested

### Characteristics

* Blue (active) and Green (testing) environment are of similar infrastructure
* Incrementally deploy to idle green environment while blue is still in use
* Allows unfettered testing of complete green environment, catch & correct issues
* Use load balancer to switch servers from Blue to Green on test acceptance
* Roll-back by switching from Green back to Blue

*[Explain how you will ensure that rollback by switching back to blue will be without problems.]*

## Canary Deployment

A screenshot of a computer

Description automatically generated

Figure 12 Canary Deployment

```mermaid

---

title: Canary Deployment

---

graph LR

CA("fa:fa-users<br>Users"):::light

subgraph IN[Server-side]

LB("fa:fa-share-nodes<br>Load<br>Balancer"):::db

subgraph RS[TO-BE Application]

subgraph SU[Some users]

S3("fa:fa-gears<br>Application<br>Version 1.1"):::gray

end

subgraph MU[Most users]

S4("fa:fa-gears<br>Application<br>Version 1.0"):::gray

end

end

subgraph LS[AS-IS Application]

S1("fa:fa-gears<br>Application<br>Version 1.0"):::gray

end

end

CA --> LB

LB --"Before deployment"--> LS

LB --"Some users"--> SU

LB --"Most users"--> MU

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

### Application

* When incremental roll-out is required
* When targeted roll-out to user communities / business departments is required
* Usually used in conjunction with Blue-Green deployment

### Characteristics

* Incremental roll-out based on subset of users / servers / geographical area is selected as test subjects
* Back-out has minimal impact
* Combine with Rolling Deployment for best results
* Not so easy when rolling database schema changes

*[Explain how the test subjects will be monitored, and how unimpeded rollbacks would be implemented. ]*

## Feature Toggle

A diagram of a software application

Description automatically generated

Figure 13 Feature Toggle

```mermaid

---

title: Feature Toggle Deployment

---

graph LR

CA("fa:fa-users<br>Users"):::light

subgraph IN[Server-side]

LB("fa:fa-share-nodes<br>Load<br>Balancer"):::db

subgraph TB[TO-BE Application]

S1("fa:fa-gears<br>Application<br>Version 1.0<br>+ new feature"):::gray

S2("fa:fa-gears<br>Application<br>Version 1.0<br>+ new feature"):::gray

end

end

CA --> LB

LB --"A: Feature-enabled users"--> TB

LB --"B: Legacy users"--> TB

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

### Application

* Managing new features in an application
* Need to do A/B Testing of a feature
* Used in conjunction with Canary Rollout

### Characteristics

* Offers control for new features
* Adding toggles to code can add complexity to the code-base, a.k.a. toggle-debt
* Old replaced features accumulate and add to redundant code base

# Solution Dependencies

The following list is a non-exhaustive list of applications and in-flight projects within [company]’s technical estate, which this project is dependent on.

| **System/Project** | **Content** | **Interface** | **Comments** | **I/O** |
| --- | --- | --- | --- | --- |
| CSD | Product Search | Web service | [comment] |  |
| CSD | Image Retrieval | Web service | [comment] |  |

Table 67 Projects that this project is dependent on

There are no applications or in-flight projects that are dependent on this project:

| **System/Project** | **Content** | **Interface** | **Comments** | **I/O** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

Table 68 Projects that are dependent on this project

* **Operational Requirements**

This section provides an overview of the business processes affected by this solution in order to provide an understanding of the operational requirements and how the solution interacts with [Company] System's IT Operations.

| **Requirement** | **Implementation** |
| --- | --- |

**Operational Requirements and Implementations:**

| **Requirement** | **Implementation** |
| --- | --- |
| Monitor and alert service view | None |
| Partner Services | None. |
| Systems Management | Windows-based solution |
| Incident Management | L2 support provided by vendor |
| Change and Release Management | None. Vendor's responsibility |
| Capacity Management | None |
| Service Continuity | None |

**Solution Front-end:**

| **Requirement** | **Implementation** |
| --- | --- |
| Application Software Delivery | Ad-hoc, delivered by vendor |
| Client Asset Management | None |
| Security Auditing | None |
| Remote Control | MS Terminal Services and VNC |

**Solution Back-end:**

| **Requirement** | **Implementation** |
| --- | --- |
| Server Software Delivery | MS SCCM 20XX |
|  | Red Hat YUM |
| Patch management | MS SCCM 20XX |
|  | Red Hat YUM |
| Security Auditing | None |
| Remote Control | MS Terminal Services |
|  | Linux VNC |
|  | SSH Terminal |
| Anti-Malware | [Brand-X] AV |
|  | Linux Malware Detect |

Table 69 Operational Requirements and Implementations

# Solution Review and Assessment

## Technology Stack Overview

Table 70 below lists all the technologies used in this solution.

| **Technology** | **Description** |
| --- | --- |
| SQL Server RDBMS |  |
| Version | XXXX SPxx |
| Decision Criteria | RDBMS storage required 98% availability |
| Alternatives Considered | None. Specified by client |
| Impact of Technology Choice | None anticipated – the existing solution is SQL Server-based. |
| Microsoft .NET Framework |  |
| Version | X.X |
| Decision Criteria | None. Specified by client |
| Alternatives Considered | None |
| Impact of Technology Choice | None anticipated – .NET is downward-compatible. |
| Windows Server O/S |  |
| Version | XXXX R2 |
| Decision Criteria | None. Specified by client |
| Alternatives Considered | None. |
| Impact of Technology Choice | None anticipated |
| Red Hat Server O/S |  |
| Version | Enterprise Edition ## |
| Decision Criteria | None. Specified by client |
| Alternatives Considered | None. |
| Impact of Technology Choice | None anticipated |

Table 70 Summary of all technologies used by the solution

## Business Requirements Document Mapping

The business requirements are listed in [Ref. 6]. Table 71 lists the chief requirements that this solution satisfies.

| **Requirement Ref.** | **Solution implementation** |
| --- | --- |
| BRD-1 | *[Paragraph cross-references]* |
| BRD-2 | *[Paragraph cross-references]* |
| etc.. | *[Paragraph cross-references]* |

Table 71 BRD Mapping

## Functional Requirements Mapping

| **Requirement Ref.** | **Solution implementation** |
| --- | --- |
| FRS-1 | *[Paragraph cross-references]* |
| FRS-2 | *[Paragraph cross-references]* |
| etc.. | *[Paragraph cross-references]* |

Table 72 FRS Mapping

## Non-Functional Requirements Mapping

| **Requirement Ref.** | **Solution implementation** |
| --- | --- |
| NFR-1 | *[Paragraph cross-references]* |
| NFR-2 | *[Paragraph cross-references]* |
| etc.. | *[Paragraph cross-references]* |

Table 73 NFR Mapping

## Assessment against Technology Compliance

| **Criteria** | **RAG** | **Rationale** |
| --- | --- | --- |
| Servers and Devices |  |  |
| Complexity | Green | This is a repeatable solution being deployed |
| Technology | Green | Standard [Company] System application and on the roadmap |
| Volumetric | Green | An increase can be comfortably accommodated |
| Storage |  |  |
| Complexity | Green | This is a repeatable solution being deployed |
| Technology | Green | Standard [Company] System application and on the roadmap |
| Volumetric | Green | An increase can be comfortably accommodated |
| Network |  |  |
| Complexity | Green | This is a repeatable solution being deployed |
| Technology | Green | Standard [Company] System application and on the roadmap |
| Volumetric | Green | An increase can be comfortably accommodated |
| Infrastructure services |  |  |
| Complexity | Green | This is a repeatable solution being deployed |
| Technology | Green | Standard [Company] System application and on the roadmap |
| Volumetric | Green | An increase can be comfortably accommodated |
| Application services |  |  |
| Complexity | Green | This is a repeatable solution being deployed |
| Technology | Red | A non-standard [Company] System application and not on the roadmap |
| Volumetric | Green | An increase can be comfortably accommodated |
| Security |  |  |
| Complexity | Green | This is a repeatable solution being deployed |
| Technology | Green | Standard [Company] System application and on the roadmap |
| Volumetric | Green | An increase can be comfortably accommodated |
| Systems Management |  |  |
| Complexity | Green | This is a repeatable solution being deployed |
| Technology | Green | Standard [Company] System application and on the roadmap |
| Volumetric | Green | An increase can be comfortably accommodated |

Table 74 Technology RAG status

## Community Assessment

| **Question** | **Assessment** | **Comment** |
| --- | --- | --- |
| How will the project impact the energy usage of IT? | Increase | [Example: New virtual servers] |
| Will the project outcome impact the wider energy or fuel costs of [Company] System? | Increase | [Example: Business efficiencies will reduce overall energy costs] |
| If dedicated hardware, are there any energy management systems included in the product to ensure power savings? | Yes | [Example: Power Management is enabled that scales CPU clock cycles.] |
| Was energy usage a significant factor in the choice of the solution architecture? | No | [Example: Not specified by vendor] |
| Other environmental impacts: | Yes | [Example: Printing] |

Table 75 Assessment against Community Concerns

## Assessment against IT Principles

| **IT Principle** | **RAG** | **Comment** |
| --- | --- | --- |
| Consistent Delivery |  |  |
| Establish common processes and systems worldwide |  |  |
| Change the business process before changing the package |  |  |
| Design for group repeatability – build once use many |  |  |
| Centralise development and support |  |  |
| Get it right first time |  |  |
| Create an agile environment able to respond to business and systems change |  |  |
| One version of the truth – there will be one source for key business data |  |  |
| Automate complex jobs |  |  |
| Flexible & open inter-application data exchange to reduce complexity |  |  |
| Where practical data should flow through systems, not be batched |  |  |
| Create solutions that actively identify problems and fix them |  |  |
| Understand the marketplace |  |  |
| Buy before build except where there is a competitive advantage |  |  |
| Our systems are secure and legal |  |  |
| Cost-effective solutions |  |  |
| We will use IT to reduce overall operating costs of the business |  |  |

Table 76 Assessment against IT principles

## Assessment against Information principles

**Integration Principles**

| **Id** | **Principle to be met** | **RAG** | **Comment** |
| --- | --- | --- | --- |
| 1 | Design for group repeatability |  |  |
| 2 | Create an agile environment able to respond to business and systems change |  |  |
| 3 | The Integration Layer is for integration |  |  |
| 4 | Flexible and open exchange of data between applications to minimise application spaghetti |  |  |
| 5 | Where practical data should flow through the systems, not be daily batches |  |  |
| 6 | Supportability is vital |  |  |

Table 77 Assessment against Integration Principles

... and more

**Data Principles**

| **Id** | **Principle to be met** | **RAG** | **Comment** |
| --- | --- | --- | --- |
| 1 | Corporate data is described in a [Company] System way. |  |  |
| 2 | Design for group repeatability |  |  |
| 10 | One version of the truth |  |  |
| 11 | Data is made readily available |  |  |
| 12 | Data must be owned and managed throughout its life |  |  |
| 13 | Data is clean and accurate |  |  |
| 14 | Data is Secure |  |  |

Table 78 Assessment against Data Principles

... and more

## Exceptions from the Technology & Data Architecture principles

[List the exceptions that had to be made in this design:]

| **Exception:** | **EXP-1: [Design aspect]** |
| --- | --- |
| Principle: | [Principle that was broken] |
| Description: | [How is this principle broken] |
| Justification: | [Why is this principle broken] |
| Recovery Plan: | [How, if ever, will this be remedied] |
| **Exception:** | **EXP-2: [Design aspect]** |
| Principle: | [Principle that was broken] |
| Description: | [How is this principle broken] |
| Justification: | [Why is this principle broken] |
| Recovery Plan: | [How, if ever, will this be remedied] |

Table 79 Solution Exceptions

# TO-BE Batch Processes

## Introduction

This chapter lists the batch processes that are run on the database, which perform the following types for functions:

Function 1

Function 2

Etc…

The batches are controlled by the [SQL Server AT Server, CRON job, Windows AT job, Oracle DBMS\_JOB, Airflow, etc…] scheduler.

## Batch Processes

[State purpose and schedule of each batch job. Use the UNIX CRON specification to indicate the batch schedule ]

|  |  |  |  |
| --- | --- | --- | --- |
| **ProcessId** | **Name** | **Description** | **CRON Schedule** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |

Table 80 Batch Processes that support the operation of this solution

1. Interfaces

The interfaces serve as a bridge between components to exchange data and events.

1. Choice of API Architectural Styles

This section explains the types of interfaces used in this solution and the reason for their choice. Select the API implementation technique used in the design and explain why they were chosen. These are the most popular API styles. Define the API authentication methods used, the upgrade paths of the APIs and the tooling used to maintain the API.

SOAP (Simple Object Access Protocol) API style

**Characteristics:**

* XML-based for enterprise applications for exchanging structured information
* Mature, but complex and verbose
* Used in Financial Services, payment gateways, and other enterprise services
* Complex and verbose - overkill for lightweight and mobile applications
* Strict standards, used in more regulated environments
* Stateful

A diagram of a computer code

Description automatically generated with medium confidence

Figure 14 Conceptual SOAP interface

```mermaid

graph LR

Server1["fa:fa-server<br>Server 1"]:::gray

Server2["fa:fa-server<br>Server 2"]:::gray

Server1 --XML--> Server2

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

ReST (Representational State Transfer) API style

**Characteristics:**

* Resource-based for web servers
* Lightweight
* Very prevalent
* Simple
* Caches HTTP GET-requests very well
* Not optimal for real-time data
* Not optimal for highly-connected data model - multiple calls are required to get at specific data
* APIs are versioned for backwards compatibility
* Stateless
* Easy to scale
* Allows for the independent natures of clients and servers
* No built-in security
* Difficult to design a consistent URI path scheme in a complex project

*A diagram of a software error

Description automatically generated*

Figure 15 Conceptual ReST interface

```mermaid

graph LR

Client[fa:fa-user<brClient]:::gray

Server[fa:fa-server<brServer]:::gray

Client --"POST /products<br>+ JSON Resource"--> Server

Server -."200 OK".-> Client

Server -."400 Client error".-> Client

Server -."500 Server error".-> Client

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

**ReST API Design Sanity checks:**

* URIs are in lower case
* URI words are hyphen-separated
* URI words are not file extensions
* Branches are plural nouns
* Slashes indicate hierarchy
* URI nouns and path schemes are consistently applied across project
* Resources are named after nouns ("products"), not verbs ("getProducts")
* Pagination is used to deal with large datasets: /products?limit=16&offset=50
* Versioning is used, e.g.: api/v3/products

**ReST CRUD Operations and Responses**

| **HTTP Verb** | **CRUD Operation** | **Collection (e.g. /customers)** | **Item (e.g. /customers/{id})** |
| --- | --- | --- | --- |
| **POST** | Create | 201 (Created), 'Location' header with link to /customers/{id} containing new ID. | 404 (Not Found), 409 (Conflict). |
| **GET** | Read | 200 (OK), list of customers. | 200 (OK), single customer. 404 (Not Found) |
| **PUT** | Update/Replace | 405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection. | 200 (OK) or 204 (No Content). 404 (Not Found) |
| **PATCH** | Update/Modify | 405 (Method Not Allowed), unless you want to modify the collection itself. | 200 (OK) or 204 (No Content). 404 (Not Found) |
| **DELETE** | Delete | 405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable. | 200 (OK). 404 (Not Found) |

**HTTP Status Codes**

| **Code** | **1xx Informational code** |
| --- | --- |
| 100 | Continue |
| 101 | Switching Protocols |
| 102 | Processing (WebDAV) |

| **Code** | **2xx Client-side Success code** |
| --- | --- |
| 200 | OK |
| 201 | Created |
| 202 | Accepted |
| 203 | Non-Authoritative Information |
| 204 | No Content |
| 205 | Reset Content |
| 206 | Partial Content |
| 207 | Multi-Status (WebDAV) |
| 208 | Already Reported (WebDAV) |
| 226 | IM Used |

| **Code** | **3xx Redirection code** |
| --- | --- |
| 300 | Multiple Choices |
| 301 | Moved Permanently |
| 302 | Found |
| 303 | See Other |
| 304 | Not Modified |
| 305 | Use Proxy |
| 306 | (Unused) |
| 307 | Temporary Redirect |
| 308 | Permanent Redirect (experimental) |

| **Code** | **4xx Client-side Error code** |
| --- | --- |
| 400 | Bad Request |
| 401 | Unauthorized |
| 402 | Payment Required |
| 403 | Forbidden |
| 404 | Not Found |
| 405 | Method Not Allowed |
| 406 | Not Acceptable |
| 407 | Proxy Authentication Required |
| 408 | Request Timeout |
| 409 | Conflict |
| 410 | Gone |
| 411 | Length Required |
| 412 | Precondition Failed |
| 413 | Request Entity Too Large |
| 414 | Request-URI Too Long |
| 415 | Unsupported Media Type |
| 416 | Requested Range Not Satisfiable |
| 417 | Expectation Failed |
| 418 | I'm a teapot (RFC 2324) |
| 420 | Enhance Your Calm (Twitter) |
| 422 | Un-processable Entity (WebDAV) |
| 423 | Locked (WebDAV) |
| 424 | Failed Dependency (WebDAV) |
| 425 | Reserved for WebDAV |
| 426 | Upgrade Required |
| 428 | Precondition Required |
| 429 | Too Many Requests |
| 431 | Request Header Fields Too Large |
| 444 | No Response (Nginx) |
| 449 | Retry With (Microsoft) |
| 450 | Blocked by Windows Parental Controls (Microsoft) |
| 451 | Unavailable For Legal Reasons |
| 499 | Client Closed Request (Nginx) |

| **Code** | **5xx Server-side Error code** |
| --- | --- |
| 500 | Internal Server Error |
| 501 | Not Implemented |
| 502 | Bad Gateway |
| 503 | Service Unavailable |
| 504 | Gateway Timeout |
| 505 | HTTP Version Not Supported |
| 506 | Variant Also Negotiates (Experimental) |
| 507 | Insufficient Storage (WebDAV) |
| 508 | Loop Detected (WebDAV) |
| 509 | Bandwidth Limit Exceeded (Apache) |
| 510 | Not Extended |
| 511 | Network Authentication Required |
| 598 | Network Read timeout error |
| 599 | Network Connect timeout error |

GraphQL API style

**Characteristics:**

* Supports "mutations" (i.e. insert/update/delete of data)
* Supports "subscriptions" (i.e. client notification of a server-side data modification)
* Presents a uniform client interface end-point to heterogeneous backend databases as viewed through a single schema
* It is also a Query Language
* Flexible query capabilities
* Able to get at a very specific piece of data in one operation, where ReST required a series of iterative calls
* One can specify resources and the required data fields in the query
* There is no risk of “over-fetching” or “under-fetching” of data, as in ReST
* GraphQL has a reduced network load
* Faster responses
* Steep learning curve
* More server-side processing
* More difficult to cache, does not use GET which is easy to cache, only POST
* More complex schema-based requests, e.g.

GET /graphql?query={book(id:"123"){title,authors{name}}

vs

GET /books/123

* Requires client-side and server-side tooling (e.g. Apollo, Postman) and libraries (e.g. schema.graphql, codegen.yml, operation.graphql)
* Can overload server with client-initiated full-table scans

A diagram of a server

Description automatically generated

Figure 16 Conceptual GraphQL interface

```mermaid

graph LR

Client[fa:fa-user<brClient]:::light

Server[fa:fa-server<br>Server]:::gray

Data1[(Database<br>1)]:::db

Data2[(Database<br>2)]:::db

Data3[(Database<br>3)]:::db

Client --"HTTP POST JSON Query"--> Server

Server --"SQL sub-query"--> Data1

Server --"SQL sub-query"--> Data2

Server --"SQL sub-query"--> Data3

Server -."JSON response".-> Client

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

gRPC API style

gRPC is roughly 7 times faster than ReST when receiving data and 10 times faster when sending data. This is mainly due to the tight packing of the Protocol Buffers and the use of HTTP/2 streaming. Top companies are switching to gRPC.

Application:

* Remote Procedure Call for inter-service calls
* Strongly typed API schema definition in a .proto file, which auto-generates client and server code
* Most modern style, preferred to standard RPC

Use cases:

* High-performance solutions
* Mobile applications
* Used for micro-services architectures

Characteristics:

* High performance for network services
* Uses protocol buffers by default to encode structured data
* Protocol buffer encoding is highly efficient (5x faster than JSON)
* Uses HTTP/2 Streams to reduce latency
* Handles multiple requests simultaneously to increase throughout
* Very limited browser support
* Toolset has wide language support

A close-up of a sign

Description automatically generated

Figure 17 Conceptual gRPC interface

```mermaid

graph LR

Server1["fa:fa-server<brServer 1"]:::gray

Server2["fa:fa-server<brServer 2"]:::gray

Server1 --"Binary request"--> Server2

Server2 -."Binary response".-> Server1

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

**gRPC-web**

WebgRPC is a JavaScript implementation of gRPC for Web clients and sends gRPC requests directly to the server. gRPC-web uses the Envoy proxy to forward gRPC requests from the client to the server.

WebSocket API style

Application:

* Synchronous communications
* Persistent connections
* Live-chat apps and real-time gaming

Characteristics:

* Bi-directional
* Real-time
* Persistent connection
* Full-duplex TCP connection
* Low-latency data exchange

```mermaid

graph TD

Server[fa:fa-server<br>Websocket<br>Server]:::gray

ClientApp1["fa:fa-user<br>Messenger 1"]:::light

ClientApp2["fa:fa-user<br>Messenger 2"]:::light

Server --"Push message:<br>How are you?"--> ClientApp1

ClientApp2 --"How are you?"--> Server

Server --"Push message:<br>Hello!"--> ClientApp2

ClientApp1 --"Hello!"--> Server

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#bbb,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

A diagram of a chat bot

Description automatically generated*```*

Figure 18 Conceptual WebSocket interface

Webhooks API style

**Application:**

* Asynchronous for real-time event-driven applications
* Used to integrate to external services
* When complete decoupling and resilience is required

**Characteristics:**

* Uses HTTP
* One of few ways web apps can inter-communicate
* Signals an event
* Does not request data
* Returned data (if any) is asynchronously returned
* Decouples asynchronously to external systems using events
* Retries and failures can elegantly deal with isolation
* Often uses a message broker for resilience and failure management

A diagram of a computer

Description automatically generated

```mermaid

graph LR

AppServer["fa:fa-gear<br>Event generator"]

Broker["fa:fa-gears<br>Event Broker"]

WebhookService["fa:fa-server<br>Webhook service"]

Server1["fa:fa-server<br>HTTP API 1"]

Server2["fa:fa-server<br>HTTP API 2"]

Server3["fa:fa-server<br>HTTP API 3"]

AppServer --"JSON event data"--> Broker

Broker --"JSON event data"--> WebhookService

WebhookService --> Server1

WebhookService --> Server2

WebhookService --> Server3

WebhookService -."async".-> AppServer

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

Figure 19 Conceptual Webhooks interface

AJAX (Asynchronous JavaScript and XML)

* Collection on Web technologies
* Asynchronous requests
* Used build-in XML request object

Database Replication Interfaces

A close-up of a white background

Description automatically generated

Figure 20 Conceptual Database Replication interface

```mermaid

graph LR

DB1[("Database<br>1")]:::db

DB2[("Database<br>>2")]:::db

DB1 -.Replication.-> DB2

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

1. API Performance Optimization

*Indicate how, why and which, if any, of the following design approaches have been used to optimize API performance.*

* **Results Caching**

Cache results of expensive computations. Use Redis or MemCached where key-value pairs are used.

* **Database connection pooling**

Maintain a pool of open database connections instead of opening and negotiating a new DB connection for every API call. For "serverless" database solutions that can’t use connection pooling, use an Amazon RDS (“Relational Database Services”) proxy.

* **Joined queries in API**

Fetch a single joined query instead of N+1 separate queries to reduce round-trips to the database and API calls. GraphQL intrinsically solves this.

* **Pagination**

Chunk large data up using a pagination scheme, e.g. specify which page, page size, and number of pages in the API’s query operation

* **JSON Serialize/de-serialize**

Use a fast, lightweight serializer to create JSON payloads and unpack JSON responses

* **Payload Compression**

Compress large API data payloads using Brotli compression, or use the intrinsic compression offered by CloudFlare, or use gRPC

* **Asynchronous Logging**

Log all events asynchronously so that the application does not need to wait for the log write to be committed before it can continue.

1. Access Control Choices

Explain the authentication schemes that are used for access control between components and users. Here we list the most popular forms of authentication mechanisms. *The authentication and identity verification scheme should strike a balance between the aspects below. For example, consider the different user experiences of initial sign-up and regular logging on.*

A diagram of a process

Description automatically generated

Figure 21 Balance between user experience and other aspects

```mermaid

graph LR

A["fa:fa-industry<br>Operational<br>Efficiency"]:::gray

B["fa:fa-handcuffs<br>Fraud Loss<br>Prevention"]:::gray

C["fa:fa-user<br>Optimal User<br>Experience"]:::gray

D["fa:fa-clock<br>Regular<br>Compliance"]:::gray

A --> B

B --> C

C --> D

D --> A

classDef gray fill:#999,stroke:#fff,stroke-width:2px,font-size:14px, color:#fff;

```

1. OAUTH2 Access control between [APP] and [3rdPartyApp] for [User]

Consider this example scenario: [User] is already authenticated on [App] and gives [3rdPartyApp] permission to access App's resources without exposing [User]'s authentication credentials to [3rdPartyApp].

Embellish this diagram in the Solution Security chapter with relevant details.

A screenshot of a computer

Description automatically generated

Figure 22 OAuth2 Conceptual example

```mermaid

sequenceDiagram

actor User

participant 3rdPartyApp

participant AppAuth

participant AppResource

autonumber

User->>3rdPartyApp: Print an Image for me!

3rdPartyApp->>AppAuth: Authorise Service request(userid,scope)

AppAuth->>User: RequestPermission Dialog

User->AppAuth: Approve request

AppAuth->>3rdPartyApp: PermissionGranted(authcode)

Note over 3rdPartyApp,AppAuth: Generate a shared secret

3rdPartyApp->>AppAuth: GetAccessToken(authcode,clientid,secret)

Note over AppAuth: Verify(authcode,clientid,secret)

AppAuth->>3rdPartyApp: IssueAccessToken(accesstoken)

3rdPartyApp->>AppResource: RequestResource(image)

AppResource->>3rdPartyApp: GetResource(image)

Note over 3rdPartyApp: ProcessResource(image)

Note over AppAuth: On expiry of token, <br/>issue a new tokenn

3rdPartyApp->>AppAuth: RefreshToken

AppAuth->>3rdPartyApp: IssueAccessToken

```

1. SSH Keys

Cryptographic keys are used to securely access remote systems and devices.

A screenshot of a diagram

Description automatically generated

Figure 23 Client/Server authentication over SSH

```mermaid

sequenceDiagram

participant Client

participant Server

Note over Client, Server: Initial set-up

Client->>Client: Generate PUBLIC/PRIVATE keypair

Client->>Server: Copy Client's PUBLIC key to<br/>Server's authorized\_keys file

Note over Client,Server: Start a new Session

Client->>Server: Send Client's PUBLIC key

Server->>Server: Is it in the<br/>authorized\_keys file?

Server->>Client: Send NONCE encrypted with<br/> Client's PUBLIC key

Client->>Client: Uses PRIVATE key to<br/>decrypt Nonce and<br/>computes HASH for NONCE

Client->>Server: Send NONCE's HASH

Server->>Server: Confirms HASH,<br/>Establish session

activate Server

Client<<->>Server: Communicate both ways<br/>with same PUBLIC/PRIVATE keypair:<br/>Commands, file transfers

Client->>Server: End session

deactivate Server

```

1. SSL Certificates

**Summary:** Digital certificates ensure secure and encrypted commnication between clients and servers

```mermaid

sequenceDiagram

participant Client

participant Server

participant CA as Certificate<br/>Authority

Note over Client,Server: Start a new Session

Client->>Server: Client's issues secure session request

Server->>Client: Sends X.509 Certificate <br/>with Server's public key

Server->>Client: Send Msg encrypted with<br/> Client's PUBLIC key

Client->>CA: Authenticates Certificate

Client->>Client: Generate symmetric key &<br/>encrypts it with Server's public key

Client->>Server: Send key-bundle

Server->>Server: Decrypt to get symmetric key

activate Server

Client<<->>Server: Communicate both ways<br/>using symmetric key-encrypted<br/> data

Client->>Server: End session

deactivate Server

A screenshot of a computer

Description automatically generated

Figure 24 Conceptual example of SSL authentication

1. User Credentials

A user’s authentication information is used to verify and grant access to systems, devices and services. As a rule, we always communicate over an encrypted SSL channel for public network traffic.

```mermaid

sequenceDiagram

participant Client

participant Server

participant UserDatabase

Note over Client,Server: Already established SSL connection...

Client->>Server: UserId and Password<br/>over encrypted HTTPS<br/>Connection

Server->>Server: Decrypts with SSL Certificate<br/>

Server->>UserDatabase: Username lookup

Server->>UserDatabase: Hashed Password verify

Server->>Client: Notify "Access granted"

activate Server

Client<<->>Server: Communicate both ways<br/>using symmetric key-encrypted<br/> data

Client->>Server: End session

deactivate Server

```

A screenshot of a computer

Description automatically generated

Figure 25 Conceptual Client Server Authentication Scheme with user credentials

1. Non-Distributed System Patterns

*[Indicate which one of these system design patterns was used in this solution, or add your own new pattern.]*

1. Layered Architecture

Provide abstraction and encapsulation with each layer having a distinct responsibility. Typically, the layers are presentations, business logic, data access and persisted data This allows a designer to make changes in one layer that will not adversely affect other layers. A specialisation of this layered architecture pattern is the Model-View-Controller.

A diagram of a business process

Description automatically generated

Figure 26 Example of how to layer architecture in abstract layers

```mermaid

graph TD

A1[fa:fa-user-tie<br>Presentation Layer]:::gray

A2[fa:fa-building<br>Business Layer]:::gray

A3[fa:fa-door-open<br>Data Access Layer]:::gray

A4[fa:fa-database<br>Data Storage Layer]:::gray

A1 --> A2

A2 --> A3

A3 --> A4

classDef gray fill:#999,stroke:#fff,stroke-width:2px,font-size:14px, color:#fff;

```

1. Event-driven architecture

Promotes the production and consumption of events between loosely coupled components. Components do not communicate directly with the application but react to published events.

A diagram of a message broker

Description automatically generated

```mermaid

flowchart LR

A1[fa:fa-envelope <br>Event<br>Producer 1]:::gray

A2[fa:fa-envelope <br>Event<br>Producer 2]:::gray

A3[fa:fa-envelope <br>Event<br>Producer 3]:::gray

A4[fa:fa-envelope-open <br>Event-in<br>Consumer 4]:::gray

A5[fa:fa-envelope-open <br>Event-in<br>Consumer 5]:::gray

A6[fa:fa-envelope-open <br>Event-in<br>Consumer 6]:::gray

C1[fa:fa-server <br>Message<br>Broker]

A1 --> C1

A2 --Produce events--> C1

A3 --> C1

C1 --> A4

C1 --Process events--> A5

C1 --> A6

classDef gray fill:#999,stroke:#fff,stroke-width:2px,font-size:14px, color:#fff;

```

1. Microkernel architecture

A small, core application with the ability to extend its functionality with plug-in components. Coupling between the plug-in components should be kept to a minimum, if at all.

A diagram of a care system

Description automatically generated

```mermaid

flowchart TD

A1[fa:fa-plug <br>Plug-in<br>Component 1]:::gray

A2[fa:fa-plug <br>Plug-in<br>Component 2]:::gray

A3[fa:fa-plug <br>Plug-in<br>Component 3]:::gray

A4[fa:fa-plug <br>Plug-in<br>Component 4]:::gray

A5[fa:fa-plug <br>Plug-in<br>Component 5]:::gray

A6[fa:fa-plug <br>Plug-in<br>Component 6]:::gray

C1[fa:fa-server <br>Core <br>System]

C1 <--> A1 & A2 & A3 & A4 & A5 & A6

classDef gray fill:#999,stroke:#fff,stroke-width:2px,font-size:14px, color:#fff;

```

1. Microservices architecture

A collection of small, loosely coupled services. Each service represents its business functionality and can contain its data model and database. Communication to the services is typically done via APIs and it promotes design modularization and agility. The trade-off is in the management of inter-service communication and maintaining data consistency between the data models.

A diagram of a application

Description automatically generated

Figure 27 Conceptual Microservices architecture

```mermaid

graph TD

Application:::gray

APIGateway:::light

ServiceA["fa:fa-gear<brService A"]:::gray

ServiceB["fa:fa-gear<brService B"]:::gray

ServiceC["fa:fa-gear<brService C"]:::gray

DBA[("DB A")]:::db

DBB[("DB B")]:::db

DBC[("DB C")]:::db

Application --> APIGateway

APIGateway --> ServiceA

APIGateway-->ServiceB

APIGateway-->ServiceC

ServiceA-->DBA

ServiceB-->DBB

ServiceC-->DBC

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#ddd,stroke:#000,stroke-width:2px,font-size:16px,color:#000

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

1. CQRS Command Query Responsibility Segregation

Separating "read" and "write" operations to storage, by hosting separate storages, each best-configured for their respective reading or writing operations. An individual service keeps the two storages consistent, with an accepted latency. This is often a message queue with a publisher/subscriber implementation.

A diagram of a blue cylinder

Description automatically generated

Figure 28 Conceptual CQRS example for a shopping application

```mermaid

flowchart LR

A(fa:fa-user<br>Client):::gray

B(fa:fa-exclamation<br>Command):::gray

C(fa:fa-question<br>Query):::gray

D[(fa:fa-pen<br>Write<br>Database)]:::db

E[(fa:fa-book<br>Read<br>Database)]:::db

F(fa:fa-desktop<br>User<br>Interface):::gray

G(fa:fa-rotate<br>Eventual<br>Data consistency<br/>service):::light

%% Connections

A ==Browse and<br>order shopping==> F

F ==Write<br>Order==> B

F <==Read<br>Products==> C

B ==> D

E ==> C

D -.-> G

G -.-> E

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff,stroke-dasharray: 5 5

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

1. Monolithic architecure

A single deployable unit/executable forms the basis for first-iteration solutions and smaller solutions. It is a middle-ground that allows refactoring the design into a microservices, microkernel or event-driven architecture. This exemplifies how simplicity can trump complexity.

Used by:

Stack Overflow

A specialization of this pattern is the modular monolith, where the boundaries between functions in the codebase are more clearly demarcated. This allows for easier maintenance and scalability.

A diagram of a diagram

Description automatically generated

Figure 29 Monolithic architecture, like we did in the 90's

```mermaid

flowchart TD

A[fa:fa-user Client]:::gray

subgraph Monolith [Single Monolithic Instance]

Lib1[fa:fa-gears<br>Procurement<br>management]:::gray

Lib2[fa:fa-gears<br>Inventory<br>management]:::gray

Lib3[fa:fa-gears<br>User<br>management]:::gray

Lib4[fa:fa-gears<br>Payment<br>management]:::gray

Lib5[fa:fa-gears<br>Order<br>management]:::gray

Lib6[fa:fa-gears<br>Logistics<br>management]:::gray

end

Monolith:::light

DB[(fa:fa-industry<br>Database)]:::db

A ==> Lib1 & Lib2 & Lib3 & Lib4 & Lib5 & Lib6

Lib1 & Lib2 & Lib3 & Lib4 & Lib5 & Lib6 ==> DB

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff,stroke-dasharray: 5 5

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

1. Distributed System Patterns

*[Indicate which one of these system design patterns was used in this solution or add your new pattern.]*

1. Ambassador Pattern

Offload functions to an intermediate proxy service to run e.g. logging, performance monitoring, retries, to reduce latency, enhance security and improve overall architecture (for extendibility). Example: "Envoy" proxy.

A diagram of a company

Description automatically generated

Figure 30 Ambassador distributed pattern

```mermaid

---

title: Ambassador Pattern

---

graph LR

CA("fa:fa-user-gear<br>Client<br>Application"):::gray

AS("fa:fa-server<br>Ambassador<br>Service"):::gray

LS("fa:fa-tree<br>Logging<br>Service"):::gray

RS("fa:fa-gear<br>Retry<br>Service"):::gray

MS("fa:fa-eye<br>Monitoring<br>Service"):::gray

CA --> AS

AS --> LS & RS & MS

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

1. Circuit-breaker Pattern

Prevents the cascading of failures when one service in a line of dependent services fails. Used with micro-services and cloud-based applications, where failures are more likely to occur and allows for a graceful degradation and for the service to recover while the service/thread is isolated with a semaphore. See Netflix/Hystrix Library.

A diagram of a problem

Description automatically generated

Figure 31 Circuit Problem Pattern

flowchart LR

SA(fa:fa-server<brService A)

SB(fa:fa-server<brService B<br>Failed):::gray

SC(fa:fa-server<brService C<br>Cascaded Fail):::light

SD(fa:fa-server<brService D<br>Cascaded Fail):::light

SA --> SB

SB --> SC

SC --> SD

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

A diagram of a circuit solution

Description automatically generated

Figure 32 Circuit Solution Pattern

```mermaid

---

title: Circuit Solution Pattern

---

flowchart LR

SA(fa:fa-server<brClosed):::gray

SB(fa:fa-server<brOpen):::gray

SC(fa:fa-server<brHalf-Open):::gray

SF((fa:fa-server<brStill<br>failing)):::dotted

F((fa:fa-server<brFailure)):::dotted

FTR((Failure<br>threshold<br>reached)):::dotted

S1((Success)):::dotted

S2((Success)):::dotted

OOT((Open on<br>timeout<br>Attempt reset)):::dotted

SC -.-> F

F -.-> SB

SA -.-> FTR

FTR -.-> SB

SB -.-> SF

SF -.-> SB

SB -.-> OOT

OOT -.-> SC

SA -.-> S1

S1 -.-> SA

SC -.-> S2

S2 -.-> SA

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef dotted fill:#fff,stroke:#000,stroke-width:1px,font-size:10px,color:#000,stroke-dasharray: 5 5

```

1. CQRS Command Query Responsibility Segregation

Separating "read" and "write" operations to storage, by hosting separate storages, each best-configured for their respective reading or writing operations. A separate service keeps the two storages consistent, with an accepted latency. This is often a message queue with a publisher/subscriber implementation.

A diagram of a computer network

Description automatically generated

Figure 33 CQRS Pattern

1. Event Sourcing

New state-representing event record for every new state change, instead of updating the same record. Useful for retrospective debugging and “time-travel” system analysis.

1. Leader Election

Provide a failover of the one and only one active leader service.

The “decision maker” in a cluster of peers is elected by a leader election service, such as Apache Zookeeper, so that only one is ever a leader. This avoids conflicts in decision-making and ensures consistent decision-making.

A screenshot of a computer

Description automatically generated

Figure 34 Concept of Leader Election

1. Publisher / Subscriber

Implemented using a message queue component.

A diagram of a group

Description automatically generated

Figure 35 Publisher/Subscriber concept

```mermaid

---

title: Message Publisher/Subscriber Concept

---

graph LR

Producer("fa:fa-envelope<br>Message<br>Producer"):::light

Broker("fa:fa-handshake<br>Broker"):::light

subgraph IN[Message Processing]

CG1("fa:fa-server<br>Consumer<br>Group 1"):::gray

CG2("fa:fa-server<br>Consumer<br>Group 2"):::gray

CG3("fa:fa-server<br>Consumer<br>Group 3"):::gray

end

Producer --"fa:fa-envelope"--> Broker

Broker --"fa:fa-envelope"--> CG1 & CG2 & CG3

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

**Characteristics:**

* Services emit events and listeners listen for events that they are interested in.
* Useful for propagating changes to multiple components.
* Better independent component scalability by decoupling event senders and receivers
* Loosely coupling between components, resulting in better system modularity
* Fault tolerant systems

**Message Queue implementation choices:**

**IBM Message Queue**

* Basic
* Still used in legacy systems
* Persistent and non-persistent queues
* No message acknowledgement
* No streaming

**RabbitMQ**

* Flexible & dynamic model
* Multiple messaging protocols AMQP, MQTT, STOMP
* Routing, queueing, publisher-subscriber features
* Allows extending functionality
* Clustering, load distribution, high availability
* Content-based routing
* Message acknowledgement
* No streaming

**Google Pub/Sub**

**Kafka**

* High throughput, real-time data streaming, high volume
* Handles multiple message producers simultaneously by partitioning on message keys
* Handles multiple consumers simultaneously by partitioning on message topics
* Comprehensively scalable platform
* Fault tolerant, message consumer failover, rebalances remaining consumers
* Brokers have a partitioned architecture, distributed commit log
* Broker failover is managed by KRaft (previously Zookeeper)
* Horizontal scaling
* High availability
* High data durability
* Replication
* Message replay a.k.a. "Time travel"
* Used for log aggregation on large estates, change data capture, real-time event streaming, performance metrics capturing, IoT
* Used by Netflix, X, LinkedIn, Airbnb & Uber

**Pulsar**

More performant than Kafka

* Cloud-native
* Multi-tenancy support
* Tiered storage - allows cheap access to historical data
* Geo-replication - good for DR and Data Locality
* Stream processing
* Many IO connectors for interfacing
* Message replay a.k.a. "Time travel"
* No Enterprise commercial backing yet

1. Sharding

Split a storage service up into separate services. This feature is built into MongoDB, Cassandra DB and others. Reduces network latency, speeds up query execution, better data locality,

A diagram of a database

Description automatically generated

Figure 36 Database Sharding concept, e.g. along lexicographic boundaries

```mermaid

---

title: Database Sharding Concept

---

graph TD

subgraph S0[Single Server]

OriginalDB[("Original<br>Database")]:::db

end

subgraph S1[Server 1]

Shard1[("Data<br>Shard<br>A-F")]:::db

end

subgraph S2[Server 2]

Shard2[("Data<br>Shard<br>G-M")]:::db

end

subgraph S3[Server 3]

Shard3[("Data<br>Shard<br>N-Q")]:::db

end

subgraph S4[Server 4]

Shard4[("Data<br>Shard<br>R-Z")]:::db

end

OriginalDB-->Shard1

OriginalDB-->Shard2

OriginalDB-->Shard3

OriginalDB-->Shard4

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff,stroke-dasharray: 5 5

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

1. API-Interfaced Cloud-based solution

A diagram of a cloud based cluster

Description automatically generated

Figure 37 Cloud-based distributed API Architecture example

```mermaid

---

title: Cloud-based distributed API Architecture example

---

graph LR

APIGateway["API<br>Gateway<br>Implementation"]

ClientApp["Client App"]

ClientCache[("Client-side<br>cache")]:::db

LB["Load<br>Balancer"]

ServiceA["Service A"];

ServiceB["Service B"];

LBCache[("Cache")]:::db

Redis["Redis"];

Elastic["elastic"];

Kafka["kafka"];

InMemCache[("In Memory<br>Cache")]:::db

CDN["Content<br>Delivery<br>Network"]

CDNStaticData[("Static Data")]:::db

DB1[("DB1")]:::db

DB2[("DB2")]:::db

IndexedData[("Indexed<br>Data")]:::db

MessageCache[("Message<br>Cache")]:::db

subgraph cluster\_CDN[CDN]

CDN

CDNStaticData

end

subgraph cluster\_Client[Client]

ClientApp

ClientCache

end

subgraph cluster\_LOAD["Load Balancer"]

LB

LBCache

end

subgraph cluster\_API["API Gateway"]

APIGateway

end

subgraph cluster\_DCache["Distributed Cache"]

Redis

InMemCache

end

subgraph cluster\_FullText["Search Service"]

Elastic

IndexedData

end

subgraph cluster\_MessageBroker["Message Broker"]

Kafka

MessageCache

end

subgraph S1[API Service A]

ServiceA

DB1

end

subgraph S2[API Service B]

ServiceB

DB2

end

ClientApp-->CDN

ClientApp-->LB

LB-->APIGateway

APIGateway-->ServiceA

APIGateway-->ServiceB

APIGateway-->Redis

APIGateway-->Elastic

APIGateway-->Kafka

ServiceA-->DB1

ServiceB-->DB2

Kafka<-->MessageCache

Elastic-->IndexedData

Redis<-->InMemCache

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff,stroke-dasharray: 5 5

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

1. Scalability Considerations

This appendix shows the workings and considerations in the design to ensure future scalability, such that the system is able to handle the increased workload without losing performance.

1. Anticipated growth

*Show how the computational and storage requirements will increase as the input load grows, based on the anticipated complexity estimate curve. Bear in mind that there will always be a limit to the ability to scale a system. If that limit is reached well within the solution’s expected lifespan, then the system should be redesigned. If relevant, also show how the system can dynamically be scaled when met by unexpected user demand, such as what a box office service might encounter when Taylor Swift tickets go on sale.*

1. Complexity estimates with Big O notation

Decide how the growth of business entities (events, users, documents, assets etc...) will affect the solution's computational load.

* **O(1) - Constant time notation**,

The runtime remains steady regardless of input size.

*Example:* Access an element in an array by index and inserting/deleting an element in a hash table.

* **O(n) - Linear time notation**

The runtime grows in direct proportion to the input size.

*Example:* Find the max or min element in an unsorted array.

* **O(log n) - Logarithmic time notation**

The runtime increases slowly as the input grows.

*Example:* A binary search on a sorted array and operations on balanced binary search trees.

* **O(n^2) - Quadratic time notation**

The runtime grows exponentially with input size.

*Example:* A simple sorting algorithms like bubble sort, insertion sort, and selection sort.

* **O(n^3) - Cubic time notation**

The runtime escalates rapidly as the input size increases.

*Example:* Multiply two dense matrices using the naive algorithm.

* **O(n logn) - Linearithmic time notation**.

This is a blend of linear and logarithmic growth.

*Example:* Efficient sorting algorithms like merge sort, quick sort, and heap sort

* **O(2^n) - Exponential time notation**

The runtime doubles with each new input element.

*Example:* Recursive algorithms solve problems by dividing them into multiple subproblems.

* **O(n!) - Factorial time notation**, W

The runtime "skyrockets" with input size.

*Example:* Permutation-generation problems, social media re-sharing.

* **O(sqrt(n)) - Square root** **time notation**

The runtime increases relative to the input's square root.

*Example:* Search within a range such as the Sieve of Eratosthenes for finding all primes up to n.

1. Design strategies

* **Avoid tight coupling between components**

List exceptions where there is tight coupling, what, if any, workarounds were applied in the design, and how that may or may not affect future scalability

* **Avoid the use of centralized components**

List exceptions where centralized components were used, what, if any, workarounds were applied in the design, and how that may or may not affect future scalability.

* **Avoid the use of high-latency components**

List exceptions where high latency components were used, what, if any, workarounds were applied in the design, and how that may or may not affect future scalability

1. Compute scaling Techniques

Load balancing

This is either a hardware, software or cloud-based service that distributes the computational load to scale users, improve performance, prevent server bottle-necks and provide robust availability, by distributing traffic across multiple servers. The solution can be scaled by add adding further servers.

A diagram of a server

Description automatically generated

Figure 38 Load Balancing concept

```mermaid

---

title: Load Balancing

---

graph LR

CA("fa:fa-display<br>Client<br>Application"):::light

subgraph IN[Internal Network]

LB("fa:fa-share-nodes<br>Load<br>Balancer"):::db

S1("fa:fa-server<br>Server<br>1"):::gray

S2("fa:fa-server<br>Server<br>2"):::gray

S3("fa:fa-server<br>Server<br>3"):::gray

end

CA --> LB

LB --> S1 & S2 & S3

classDef gray fill:#999,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef light fill:#aaa,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

classDef db fill:#336,stroke:#000,stroke-width:2px,font-size:16px,color:#fff

```

**Hardware options:**

Robust and stable appliances. Used in data centres and high-demand systems

* CISCO
* F5
* CitrixADC
* Juniper AE

**Software options:**

Cost effective, flexible, more suitable for a wider range of applications

* HAProxy
* NGiNX
* Apache JK

**Cloud options:**

Management burden is shifted to the cloud provider

* Amazon ELB
* Azure LB

These can all be categorized into Operational classes:

**Network Layer 4 Load Balancing (transport layer)**

**Features:**

* Route traffic based on TCP addresses and ports.
* Faster and more efficient
* Do not inspect network traffic content

**Use case:**

* Generally used for basic load balancing tasks

**Examples:**

* Amazon ELB
* F5
* HA Proxy
* Apace JK
* Azure LB

**Layer 7 Load Balancing (application layer)**

**Features:**

* Works with HTTP & HTTPS
* Route traffic based on HTTP headers, cookies, URLs,
* Ideal for complex applications that require complex routing
* Can be an SSL terminator and perform the encryption and decryption so that servers don’t need to do this
* Centralises certificate management and security policy
* Requires more processing

**Use case:**

* Content-based routing, e.g. scaling API
* Better routing control
* Simplify security management

**Examples:**

* Azure Application Gateway
* NGiNX
* Juniper AE

**Global Server Load Balancing (GSLB)**

**Features:**

* Load-balances global traffic
* Uses DNS routing to direct users to nearest data centre
* Provide failover across regions and high availability

**Use cases:**

* Serves global users

**Examples:**

* A10
* Radware
* CitrixADC
* CloudFlare

**Distribution algorithms:**

* Round Robin – simplest, no client persistence
* Sticky Round Robin – persists client session on server using a cookie
* Weighted Round Robin – favours the more capable servers
* IP/URL hashing – no client persistence, good for serving static content
* Least Connections – favours the server with the lowest count of active sesisons
* Least Time – favours the server that has shown the best response times

**Other benefits of Load Balancers:**

* Traffic metrics of – connection count, request rates
* Health metrics of server health and network health
* Performance metrics of latency and throughput
* Error metrics of lost connections and HTTP errors
* **Caching**
* **Event-drive architecture**
* **Massively Parallel architecture**
* **Map-Reduce**

1. Network scaling strategies
2. Storage scaling strategies
3. Database scaling strategies

*[Which of the following strategies were used in the design to scale this?]*

Indexing

Indexes are mostly fine-tuned in the course of running the solution.

* Avoid full table scan by picking the best columns to index on
* Small tables can sometimes be faster if they are not indexed
* Indexing can slow down write operations - find the right balance

Denormalisation

This approach hugely improves the user experience on large systems. In particular, this is a very useful strategy for social media platforms where the time delay between writes and subsequent reads is not mission-critical.

* Reduce complex joins between tables by duplicating data at write-time in tables that would otherwise be joined in a fully normalized schema
* Updates must be carefully managed to maintain consistency of data across the database
* Accept the cost of the time delay incurred for data to only eventually become consistent across the database.

Caching

* Store more frequently accessed data on faster storage
* Execute more frequently run queries and its data on faster memory-only databases
* Can be implemented on an application level or the middleware level.
* Cache needs to remain up to date with the most recent data - explain how the cache invalidation works:
  + Cache is refreshed on a time basis
  + Cache is refreshed on an event

Replication

An added benefit of this approach is that it: enhances fault tolerance

* Master database content is replicated to slave databases
* Synchronous replication: Immediate consistency but introduces latency
* Asynchronous replication: Zero-cost, but introduces temporary data inconsistency between master and slave server

Sharding

* Each shard is a database that contains a separate part of the data
* Reduces workload on each individual server
* Effective horizontal scaling approach
* Need to choose sharding key carefully
* Introduces more complexity
* May require changes to application CRUD queries
* Adding shards can be challenging

Vertical Scaling

This is often the first step in scaling a database since it is easy to implement and does not affect the solution architecture, although it has limitations.

* Add more resources to the database server by adding more CPUs, RAM or storage
* Balance vertical scaling with getting an additional load-balanced database server for the added benefit of getting failover redundancy in the design.

Materialized Views

This approach is useful in BI.

**Benefits:**

* Avoid the recalculation of values and aggregates by storing the computed values and aggregates in the database and reducing the computational load on the database
* Need to be periodically refreshed to remain up-to-date, which can be resource-intensive and time-consuming.
* Balance the cost of refreshing materialized views with the required refresh frequency

1. API Gateway Configuration

*Include this Appendix if the solution uses an API Gateway.]*

1. Conceptual architecture of an API Gateway

**A diagram of a application

Description automatically generated**

Figure 39 Conceptual API Gateway

```mermaid

graph LR

C1["Client App<br>A"];

C2["Client App<br>B"];

C3["Client App<br>C"];

API["API<br>Gateway"];

S1["API Service<br>A"];

S2["API Service<br>B"];

S3["API Service<br>C"];

C1 -->API

C2 -->API

C3 -->API

API --> S1

API --> S2

API --> S3

```

1. Reasons to have an API Gateway

* Isolate yourself when consuming public API services
* Provide an authentication layer to hosted APIs, which is especially for APIs consumed by the public
* Provide consistent API frontend to a group of heterogeneous API services on the back end
* Central location to secure, throttle, route and monitor all API network traffic
* Central location to monitor and report on API usage
* Acts as a balancer of API services for scaling and performance improvement, as opposed to a Load Balancer that balances traffic and actual server load

Also:

* Feature payment barriers with membership control and consumption measurement for publicly traded API services

1. Differences between API Gateways and Load Balancers

API Gateways manage API calls, while Load Balancers manage network traffic:

* API Gateways operate on the network Layer 7 only whereas Load Balancers can operate on Layer 4 and at in some cases Layer 7 too.
* An API Gateway is ideal for a microservices architecture that needs centralized API request management, whereas a Load Balancer ensures server availability and distributing network traffic
* The API Gateway is a single-entry point for clients and handles API composition, request routing and protocol translations
* An API Gateway offers rate limiting of API calls on a per-user basis, authentication, and even monetizing of API calls, whereas a Load Balancer only aims to optimize infrastructure resource usage for best performance by managing network traffic and server utilization.

1. API Gateway Options

Here are a few commercial offerings:

* Amazon API Gateway
* Google APIGEE API Gateway
* Azure API Gateway
* Kong API Gateway