System Architecture Document

Project Name: <TBD>

for <Customer Name> ● July 19, 2022



Naico Information Technology Services(P) Ltd.  
Office No. 0405B, 4th Floor, SCK 01,  
Smartcity (Kochi) SEZ, Kakkanad, Kochi, Kerala, Pin: 682042

Copyright © 2022 NaicoITS All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from NaicoITS.

All copyright, confidential information, patents, design rights and all other intellectual property rights of whatsoever nature contained herein are and shall remain the sole and exclusive property of NaicoITS. The information furnished herein is believed to be accurate and reliable.

However, no responsibility is assumed by NaicoITS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

The [Company] name and [Company] logo are trademarks or registered trademarks of NaicoITS.

All other trademarks are the property of their respective owners

System Architecture Document

[Customer Company name]

[Project / Product Name]

Version: V1A Revision Date: July 19,2022

|  |  |  |
| --- | --- | --- |
| **Executive Sponsor** | | |
| [Name] | [Email] | |
| Signature | | Date |

|  |  |  |
| --- | --- | --- |
| **Software Architect** | | |
| [Name] | [Email] | |
| Signature | | Date |

|  |  |  |
| --- | --- | --- |
| **Engineering Manager** | | |
| [Name] | [Email] | |
| Signature | | Date |

|  |  |  |
| --- | --- | --- |
| **Project Manager** | | |
| [Name] | [Email] | |
| Signature | | Date |

|  |  |  |
| --- | --- | --- |
| **Quality Engineering Lead** | | |
| [Name] | [Email] | |
| Signature | | Date |

Table of Contents

1. [1 Executive Summary 9](#_Toc109287610)

[1.1 Purpose of this document 9](#_Toc109287611)

[1.2 Identification 9](#_Toc109287612)

[1.3 Scope 9](#_Toc109287613)

[1.4 Relationship to Other Plans 9](#_Toc109287614)

[1.5 Methodology, Tools, and Techniques 10](#_Toc109287615)

[1.6 Policies, Directives and Procedures 10](#_Toc109287616)

[1.7 Definitions, Acronyms, and Abbreviations 10](#_Toc109287617)

1. [2 High level Overview 11](#_Toc109287618)

[2.1 Background Information 11](#_Toc109287619)

[2.2 System Evolution Description 11](#_Toc109287620)

[2.3 Current Process 11](#_Toc109287621)

[2.4 Proposed Process 11](#_Toc109287622)

[2.5 Technology Forecast 11](#_Toc109287623)

[2.6 User Characteristics 11](#_Toc109287624)

[2.7 User Problem Statement 11](#_Toc109287625)

[2.8 User Objectives 11](#_Toc109287626)

1. [3 Architecture Goals and Objectives 12](#_Toc109287627)

[3.1 Goals 12](#_Toc109287628)

[3.2 Objectives 13](#_Toc109287629)

1. [4 Architecture Constraints 15](#_Toc109287630)

[4.1 Constraints 15](#_Toc109287631)

1. [5 System Architecture Representation 16](#_Toc109287632)

[5.1 Hardware Architecture – High Level 16](#_Toc109287633)

[5.2 Software Architecture – High Level 16](#_Toc109287634)

[5.2.1 Scenarios 17](#_Toc109287635)

[5.2.2 Logical View 18](#_Toc109287636)

[5.2.3 Process View 18](#_Toc109287637)

[5.2.4 Implementation View 19](#_Toc109287638)

[5.2.5 Deployment View 19](#_Toc109287639)

[5.3 Communications Architecture – High Level 20](#_Toc109287640)

1. [6 System Architectural Characteristics 21](#_Toc109287641)

[6.1 Hardware Architecture Characteristics 21](#_Toc109287642)

[6.1.1 Operational Architecture Characteristics 21](#_Toc109287643)

[6.1.2 Structural Architecture Characteristics 22](#_Toc109287644)

[6.1.3 Cross-Cutting Architecture Characteristics 23](#_Toc109287645)

[6.2 Software Architecture Characteristics 24](#_Toc109287646)

[6.2.1 Operational Architecture Characteristics 24](#_Toc109287647)

[6.2.2 Structural Architecture Characteristics 25](#_Toc109287648)

[6.2.3 Cross-Cutting Architecture Characteristics 26](#_Toc109287649)

[6.3 Communication Architecture Characteristics 27](#_Toc109287650)

[6.3.1 Operational Architecture Characteristics 27](#_Toc109287651)

[6.3.2 Structural Architecture Characteristics 28](#_Toc109287652)

[6.3.3 Cross-Cutting Architecture Characteristics 28](#_Toc109287653)

1. [7 System Domain Concerns 30](#_Toc109287654)

[7.1 Hardware Architecture Decisions 30](#_Toc109287655)

[7.2 Software Architecture Decisions 30](#_Toc109287656)

[7.3 Communication Architecture Decisions 31](#_Toc109287657)

1. [8 System Architectural Decisions 32](#_Toc109287658)

[8.1 Hardware Architecture Decisions 32](#_Toc109287659)

[8.2 Software Architecture Decisions 32](#_Toc109287660)

[8.3 Communication Architecture Decisions 32](#_Toc109287661)

1. [9 System Design Principles 33](#_Toc109287662)

[9.1 Hardware Design Principles 33](#_Toc109287663)

[9.2 Software Design Principles 33](#_Toc109287664)

[9.3 Communication Design Principles 33](#_Toc109287665)

1. [10 System Design Trade-offs 34](#_Toc109287666)

[10.1 Hardware Design Trade-offs 34](#_Toc109287667)

[10.2 Software Design Trade-offs 34](#_Toc109287668)

[10.3 Communication Design Trade-offs 34](#_Toc109287669)

1. [11 System Architecture - Detailed View 35](#_Toc109287670)

[11.1 Hardware Detailed View 35](#_Toc109287671)

[11.2 Software Detailed View 36](#_Toc109287672)

[11.2.1 Use-Case View 36](#_Toc109287673)

[11.2.2 Logical View 40](#_Toc109287674)

[11.2.3 Process View 43](#_Toc109287675)

[11.2.4 Implementation View 47](#_Toc109287676)

[11.2.5 Deployment View 49](#_Toc109287677)

[11.3 Communications Detailed View 50](#_Toc109287678)

1. [12 External Interface View 51](#_Toc109287679)

[12.1 Interface Architecture 51](#_Toc109287680)

[12.2 Interface Detailed View 51](#_Toc109287681)

1. [13 Human-Machine Interface View 54](#_Toc109287682)

[13.1 Interface Design Rules 54](#_Toc109287683)

[13.2 Inputs 54](#_Toc109287684)

[13.3 Outputs 54](#_Toc109287685)

[13.4 Navigation Hierarchy 54](#_Toc109287686)

[13.4.1 Screen [x.1] 54](#_Toc109287687)

[13.4.2 Screen [x.2] 55](#_Toc109287688)

[13.4.3 Screen [x.3] 55](#_Toc109287689)

1. [14 Data View 56](#_Toc109287690)

[14.1 Database Management System 56](#_Toc109287691)

[14.1.1 Entity Relationship diagram - High level 56](#_Toc109287692)

[14.1.2 Implementation - High level 57](#_Toc109287693)

[14.2 Non-Database Management System Files 58](#_Toc109287694)

1. [15 Size and Performance 59](#_Toc109287695)
2. [16 Quality 60](#_Toc109287696)

[16.1 Applicability to Common Services Architecture 60](#_Toc109287697)

1. [17 Architectural Mechanism 61](#_Toc109287698)

[17.1 Analysis Mechanisms 61](#_Toc109287699)

[17.2 Analysis-to-Design-to-Implementation Mechanisms Map 61](#_Toc109287700)

[17.3 Implementation Mechanisms 62](#_Toc109287701)

[17.3.1 <Mechanism subsection> 62](#_Toc109287702)

1. [18 System Integrity Controls 63](#_Toc109287703)
2. [19 Conclusion 64](#_Toc109287704)
3. [20 Appendix A 65](#_Toc109287705)

[20.1 Architecture Decision Records 65](#_Toc109287706)

1. [21 Appendix B 66](#_Toc109287707)

[21.1 Requirements Traceability Matrix 66](#_Toc109287708)

[21.2 Packaging and Installation 66](#_Toc109287709)

[21.3 Design Metrics 66](#_Toc109287710)

[21.4 Glossary of Terms 66](#_Toc109287711)

Document History

Paper copies are valid only on the day they are printed. Contact the author if you are in any doubt about the accuracy of this document.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Summary of Changes | Author |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Reference Documents

Please see the following documents for more information:

|  |  |  |
| --- | --- | --- |
| Document Name | Version | Author |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Distribution List

This document has been distributed to:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | | | | Position | Company | Action | | |
|  | | | |  |  |  | | |
|  | | | |  |  |  | | |
|  | | | |  |  |  | | |
|  | | | |  |  |  | | |
|  | | | |  |  |  | | |
|  |  |  | | | |  |

# Executive Summary

Provide a brief introduction to the system for which this design is being undertaken.

## Purpose of this document

Describe the purpose of the document and its intended audience, for example:

The purpose of this document is to describe in sufficient detail how the proposed system is to be constructed. The System Design Document translates the Requirement Specifications into a document from which the developers can create the actual system. It identifies the top-level system architecture, and identifies hardware, software, communication, and interface components.

## Identification

Identify the system and software to which this document applies, including, identification number(s), title(s), abbreviation(s), version number(s), and release number(s). Identify all standards (ANSI, ISO, IEEE, etc) that apply to the design document.

## Scope

The scope establishes the boundaries of the design document and should describe features outside of the scope, for example, if certain requirements were not included in the design due to budgetary or time constraints.

|  |
| --- |
| Includes |
|  |
|  |
|  |

|  |
| --- |
| Excludes |
|  |
|  |
|  |

## Relationship to Other Plans

Describe this document’s relation to other plans, such as:

* Functional Requirements [version]
* Configuration Management Plan [version]
* Software Quality Assurance Plan [version]

## Methodology, Tools, and Techniques

Describe the software tools (or techniques) required for performing design documents tasks, e.g. software for managing changes requests that may be made during the development phase.

## Policies, Directives and Procedures

Identify any constraints or requirements placed on this document by policies, directives, or procedures.

## Definitions, Acronyms, and Abbreviations

Provide definitions for all terms, acronyms, and abbreviations required to properly interpret how they are used in this document. Provide a reference to the project’s glossary document if necessary, in the appendix section.

# High level Overview

Provide a brief introduction to the proposed system. Outline how the system will fit into the company’s business and technology environments and discuss any strategic issues if appropriate.

## Background Information

Outline any background information that is relevant to the propose design, for example, business drivers, such as the need for the company to offer customer’s new services or compliance issues, such as security controls that must be incorporated into the system design.

## System Evolution Description

**[Optional]** Describe how to migrate the existing system(s) to a more efficient system, or alternately moving an existing system to a future implementation.

## Current Process

**[Optional]** Describe the current processes that are in place (if applicable). This may help place the overall design in context.

## Proposed Process

**[Optional]** Describe the proposed process. Reference any supporting documents, if relevant.

## Technology Forecast

**[Optional]** Outline the emerging technologies that are expected to be available in a given timeframe(s), and how they may impact the future development of system the architecture.

## User Characteristics

Describe the features of the user community, and their proficiency with software systems etc.

## User Problem Statement

Describe the major problem(s) experienced by the user community.

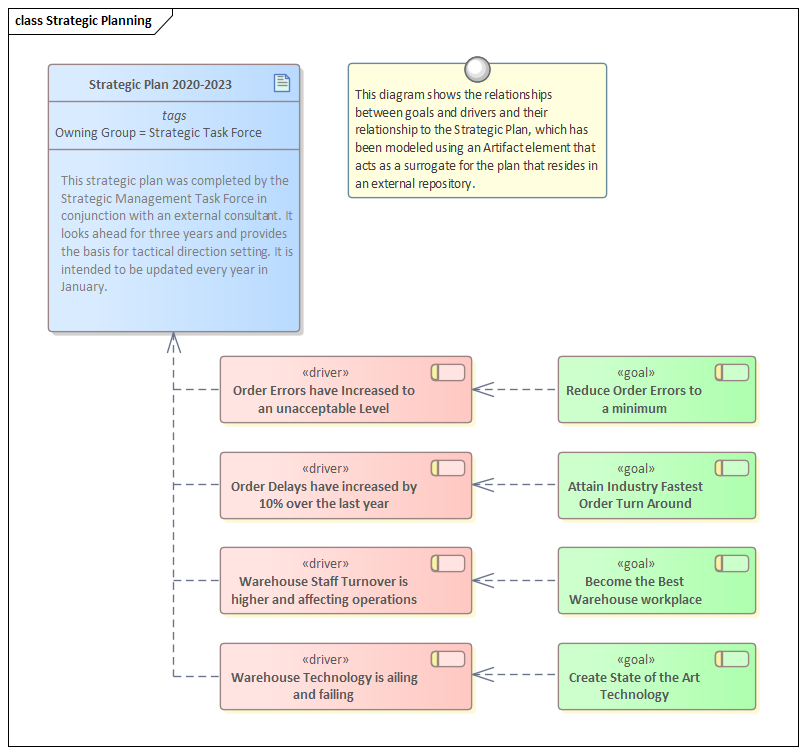
## User Objectives

Outline the users’ objectives and requirements for the new system. Where appropriate, include a "wish list" of desirable features.

# Architecture Goals and Objectives

Goals represent the decomposition of the strategy and are aspirations that the enterprise intends to achieve. They are important because they are statements that can be used as anchor points for the architecture.

For example:



## Goals

Describe the key Goals from the Architecture Design perspective to meet the business Drivers, Mission, Vision and Strategy plan.

Diagram

Description automatically generated

## Objectives

Describe the key Objectives in orders meet the from the Architecture Goals.

Diagram

Description automatically generated

# Architecture Constraints

## Constraints

Detail any constraints that are placed upon the system design, such as schedules, costs, or technical constrains, such as the company’s commitment to a specific development platform or programming language.

# System Architecture Representation

For instance, the Architecture description can be looked at from the below perspective as well.

ISO/IEC/IEEE 42010, Systems, and software engineering — Architecture description, defines the contents of an architecture description

Diagram

Description automatically generated

## Hardware Architecture – High Level

**[Optional – As per the applicability of the System under design]** Describe the system hardware architecture and indicate whether the processing system is distributed or centralized. List and describe the hardware modules with and diagrams showing the connectivity between the modules. If possible, identify the type/number/location of servers, workstations, processors, backup systems, output devices, etc.

## Software Architecture – High Level

Describe the overall system software and organization. List and describe the software modules (i.e. including functions, subroutines, or classes), programming languages, and development tools.

Describe all software required to support the system, and specify the physical location of all software systems. Identify database platforms, compilers, utilities, operating systems, communications software, etc.

Diagrams should map to the Functional Requirements Document’s data flow diagrams.

Provide diagrams that illustrate the segmentation levels down to the lowest level. Include names and reference numbers for all features on the diagrams. Include a narrative that expands on and enhances the understanding of the functional breakdown.

Describe what software architecture the system implements. To illustrate the architecture, include the commonly used set of views known as the “4 + 1 architecture view model”.

The “4+1 architecture view model” illustrates five uniquely viewed perspectives into the design of the architecture: Use-Case, Logical, Process, Deployment, and Implementation. Each viewpoint describes:

* One or more system model(s) and view(s) of those models;
* Stakeholders interested in the view(s); and
* Stakeholder concerns in the view(s).

Diagram

Description automatically generated

### Scenarios

The Use-Case View to describe functional and non-functional significant architectural requirements (SARs). Use cases constitute the glue that unifies all the other views. Model elements in this view include:

* Actors
* Use Cases
* Classes
* Collaborations
* Audience:

All the stakeholders of the system, including the end-users.

* Area:

Describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system. Other than the basic work flow the documents addresses the exception cases, exception outputs, and other related use cases.

### Logical View

The Logical View to describe key design mechanisms, architecturally important design elements, their interdependencies and the organization of these elements into subsystems and layers. Model elements in this view include:

* Objects
* Classes
* Collaborations
* Interactions
* Packages
* Subsystems
* Audience:

Designers, Programmers, Testing staff

* Area:

Functional requirements, object hierarchy, system layers

Describes the design of object model. Also describes the subsystems of the system and their relationships.

### Process View

The Process View to describe processes and threads, and the allocation of the logical and/or the implementation elements to these processes and threads. Model elements in this view include:

* Tasks, threads, processes
* Interactions
* Audience:

Integrators, Programmers

* Area:

Non-functional requirements, describes the design's concurrency and synchronization aspects.

Elaborates the run time behavior of the system

### Implementation View

The **Implementation View / Development view** to describe key implementation elements such as code artifacts, executables, and modules. Model elements in this view include:

* Modules
* Subsystems
* Interfaces
* Audience:

Programmers, Code testers

* Area:

Software components: describes the modules and subsystem divisions of the system.

### Deployment View

The **Deployment View / Physical View** to describe system nodes such as computers or routers, and the allocation of the logical, implementation, or process elements to those nodes. Model elements in this view include:

* Nodes
* Modules

In addition to the above views, a "Data View" should be included in the Architectural Representation whenever persistent data objects are included in the system (they usually are in most software systems).

* Audience:

Database administrators, System engineers, Deployment managers

* Area:

Persistence: describes the architecturally significant persistent elements in the data model. Describes the mapping of the software onto the hardware and shows the system's distributed aspects

## Communications Architecture – High Level

**[Optional – As per the applicability of the System under design]** Describe communications within the system, such as Local Area Networks (LANs), buses, etc. Include the communications architecture(s) being implemented, such as X.25, Token Ring, etc.

Provide a diagram depicting the communications path(s) between the system and subsystem modules.

# System Architectural Characteristics

One of the key aspects is defining, Discovering, and otherwise analyzing all the things the system must to do that isn’t directly related to the domain functionality: Architectural Characteristics (it is sometimes referred as quality attributes also).

An architecture characteristic meets three criteria:

* Specifies a non-domain Design consideration.
* For instance: Performance of the intended system, Prevent Technical debt.
* Influences some structural aspects of the design
* For instance: Security aspects In-application payment processing or third-party payment processor
* Is Critical or important to application success
* Consideration should be choosing the fewest architecture characteristics which not leading complexity to the design.

## Hardware Architecture Characteristics

**[Optional – As per the applicability of the System under design]** Describe the system hardware architecture Characteristics based on the applicability based on the following sub sections.

### Operational Architecture Characteristics

Based on the applicability describes capabilities focusing on the aspects such as performance, scalability, elasticity, availability, and reliability. Here are some of the key aspects to be evaluated upon.

Operational architecture Characteristics heavily overlap with operations and DevSecOps concerns, forming the intersection of those concerns in many software systems.

#### Availability

Describes how long the system will need to be available (if 24/7,steps need to be in place to allow the system to be up and running quickly in case of failure.

#### Continuity

Describes the Disaster recovery capability.

#### Performance

Describes or outlines or includes stress testing, peak analysis, analysis of the frequency of functions used, capacity required and response times. Performance acceptance sometimes requires an exercise of its own, taking months to complete.

#### Recoverability

Describes the Business continuity requirements (for e.g. In case of disaster, How quickly is the system required to be online again ?) This will affect the backup strategy and requirements for duplicate hardware.

#### Reliability

Assess if the system needs to be fail-safe. If it fails, will it cost the company large sums of money?

#### Safety

Assess if it is mission critical in a way that affects lives. If it fails, will it cost the company large sums of money?

#### Robustness

Ability to handle error and boundary conditions while running if the internet connection goes down or if there’s a power outrage or hardware failure.

#### Scalability

Ability for the system to perform and operate as the number of users or requests increases.

### Structural Architecture Characteristics

Based on the applicability describes Code quality concerns, such as good modularity, controlled coupling between components, readable code, and host of other internal Quality assessments.

#### Configurability

Describes the Ability for the end users to easily change aspects of the software’s configuration (through usable interfaces)

#### Extensibility

Describes the approach or outlines how important it is to plug new pieces of functionality in.

#### Installability

Describes or outlines ease of system installation on all necessary platforms.

#### Leverageability/Reuse

Describes the Ability to leverage common components across multiple products.

#### Maintainability

Describes the approach or outlines thought process in terms how easy it is to apply changes and enhance the system.

#### Portability

Describes the thought process in case the system need to run on more than one platform?

#### Upgradeability

Describes the Ability to easily /quickly upgrade from a previous version of this application /solution to newer version on servers and clients.

### Cross-Cutting Architecture Characteristics

Based on the applicability describes cross cutting architectural characteristics, which are form important design consideration and leverage across the system. The key ones are considered below.

#### Accessibility

Describes the thought process in terms of enabling access to all your users, including those with disabilities like colorblindness or hearing loss.

#### Archivability

Describes the thought process in terms answering the question such as - Will the data need to be archived or deleted after a period of time ? (for e.g. customer accounts are to be deleted after three months or marked as obsolete and archived to a secondary database for future access)

#### Authentication

Describes security considerations to ensure users are who they say they are.

#### Authorization

Describes security considerations to ensure users can access only certain functions within the application (by use case, subsystem, webpage, business rule , field level etc.)

#### Legal

Describes what legislative constraints is the system operating in (data protection, Sarbanes Oxley, GDPR etc.)? What reservation rights does the company require? Any regulations regarding the way the application is to be built or deployed?

#### Privacy

Describes ability to hide transactions from internal company employees (encrypted transactions so even DBAs and network architects cannot see them).

#### Security

Describes Does the data need to be encrypted in the database? Encrypted for network communication between internal systems? What type of authentication needs to be in place for remote user access?

#### Supportability

Describes the thought process in terms of what level of technical support is needed by the application? What level of logging and other facilities are required to debug errors in the system.

#### Usability/achievability

Describes the level of training required for users to achieve their goals with the application/solution. Usability requirements need to be treated as seriously as any other architectural issue.

## Software Architecture Characteristics

Describe the system Software architecture Characteristics based on the applicability based on the following sub sections.

### Operational Architecture Characteristics

Based on the applicability describes capabilities focusing on the aspects such as performance, scalability, elasticity, availability, and reliability. Here are some of the key aspects to be evaluated upon.

Operational architecture Characteristics heavily overlap with operations and DevSecOps concerns, forming the intersection of those concerns in many software systems.

#### Availability

Describes how long the system will need to be available (if 24/7,steps need to be in place to allow the system to be up and running quickly in case of failure.

#### Continuity

Describes the Disaster recovery capability.

#### Performance

Describes or outlines or includes stress testing, peak analysis, analysis of the frequency of functions used, capacity required and response times. Performance acceptance sometimes requires an exercise of its own, taking months to complete.

#### Recoverability

Describes the Business continuity requirements (for e.g. In case of disaster, How quickly is the system required to be online again ?) This will affect the backup strategy and requirements for duplicate hardware.

#### Reliability

Assess if the system needs to be fail-safe. If it fails, will it cost the company large sums of money?

#### Safety

Assess if it is mission critical in a way that affects lives. If it fails, will it cost the company large sums of money?

#### Robustness

Ability to handle error and boundary conditions while running if the internet connection goes down or if there’s a power outrage or hardware failure.

#### Scalability

Ability for the system to perform and operate as the number of users or requests increases.

### Structural Architecture Characteristics

Based on the applicability describes Code quality concerns, such as good modularity, controlled coupling between components, readable code, and host of other internal Quality assessments.

#### Configurability

Describes the Ability for the end users to easily change aspects of the software’s configuration (through usable interfaces)

#### Extensibility

Describes the approach or outlines how important it is to plug new pieces of functionality in.

#### Installability

Describes or outlines ease of system installation on all necessary platforms.

#### Leverageability/Reuse

Describes the Ability to leverage common components across multiple products.

#### Localization

Describes approach or outlines the support for multiple languages on entry/query screens in data fields; on reports, multibyte character requirements and units of measure or currencies.

#### Maintainability

Describes the approach or outlines thought process in terms how easy it is to apply changes and enhance the system.

#### Portability

Describes the thought process in case the system need to run on more than one platform? (for e.g. does the front end need to run against Oracle as well as SAP DB).

#### Upgradeability

Describes the Ability to easily /quickly upgrade from a previous version of this application /solution to newer version on servers and clients.

### Cross-Cutting Architecture Characteristics

Based on the applicability describes cross cutting architectural characteristics, which are form important design consideration and leverage across the system. The key ones are considered below.

#### Accessibility

Describes the thought process in terms of enabling access to all your users, including those with disabilities like colorblindness or hearing loss.

#### Archivability

Describes the thought process in terms answering the question such as - Will the data need to be archived or deleted after a period of time ? (for e.g. customer accounts are to be deleted after three months or marked as obsolete and archived to a secondary database for future access)

#### Authentication

Describes security considerations to ensure users are who they say they are.

#### Authorization

Describes security considerations to ensure users can access only certain functions within the application (by use case, subsystem, webpage, business rule , field level etc.)

#### Legal

Describes what legislative constraints is the system operating in (data protection, Sarbanes Oxley, GDPR etc.)? What reservation rights does the company require? Any regulations regarding the way the application is to be built or deployed?

#### Privacy

Describes ability to hide transactions from internal company employees (encrypted transactions so even DBAs and network architects cannot see them).

#### Security

Describes Does the data need to be encrypted in the database? Encrypted for network communication between internal systems? What type of authentication needs to be in place for remote user access?

#### Supportability

Describes the thought process in terms of what level of technical support is needed by the application? What level of logging and other facilities are required to debug errors in the system.

#### Usability/achievability

Describes the level of training required for users to achieve their goals with the application/solution. Usability requirements need to be treated as seriously as any other architectural issue.

## Communication Architecture Characteristics

**[Optional – As per the applicability of the System under design]** Describe the Communication architecture Characteristics based on the applicability.

### Operational Architecture Characteristics

Based on the applicability describes capabilities focusing on the aspects such as performance, scalability, elasticity, availability, and reliability. Here are some of the key aspects to be evaluated upon.

Operational architecture Characteristics heavily overlap with operations and DevSecOps concerns, forming the intersection of those concerns in many software systems.

#### Availability

Describes how long the system will need to be available (if 24/7,steps need to be in place to allow the system to be up and running quickly in case of failure.

#### Performance

Describes or outlines or includes stress testing, peak analysis, analysis of the frequency of functions used, capacity required and response times. Performance acceptance sometimes requires an exercise of its own, taking months to complete.

#### Reliability

Assess if the system needs to be fail-safe. If it fails, will it cost the company large sums of money?

#### Robustness

Ability to handle error and boundary conditions while running if the internet connection goes down or if there’s a power outrage or hardware failure.

### Structural Architecture Characteristics

Based on the applicability describes Code quality concerns, such as good modularity, controlled coupling between components, readable code, and host of other internal Quality assessments.

#### Configurability

Describes the Ability for the end users to easily change aspects of the software’s configuration (through usable interfaces)

#### Extensibility

Describes the approach or outlines how important it is to plug new pieces of functionality in.

#### Leverageability/Reuse

Describes the Ability to leverage common components across multiple products.

#### Maintainability

Describes the approach or outlines thought process in terms how easy it is to apply changes and enhance the system.

### Cross-Cutting Architecture Characteristics

Based on the applicability describes cross cutting architectural characteristics, which are form important design consideration and leverage across the system. The key ones are considered below.

#### Authentication

Describes security considerations to ensure users are who they say they are.

#### Authorization

Describes security considerations to ensure users can access only certain functions within the application (by use case, subsystem, webpage, business rule , field level etc.)

#### Legal

Describes what legislative constraints is the system operating in (data protection, Sarbanes Oxley, GDPR etc.)? What reservation rights does the company require? Any regulations regarding the way the application is to be built or deployed?

#### Privacy

Describes ability to hide transactions from internal company employees (encrypted transactions so even DBAs and network architects cannot see them).

#### Security

Describes Does the data need to be encrypted in the database? Encrypted for network communication between internal systems? What type of authentication needs to be in place for remote user access?

#### Supportability

Describes the thought process in terms of what level of technical support is needed by the application? What level of logging and other facilities are required to debug errors in the system.

# System Domain Concerns

Many Architects and domain stakeholders want to prioritize the final list of architecture characteristics that the application or system must support.

Rarely will all stakeholders agree on the priority of each and every characteristic. A better approach is to have the domain stakeholders select the top three most important characteristics from the final list ( in any order). Not only is this much easier to gain consensus on, but it also fosters discussions about what is important and helps the architect analyze trade-offs when making vital architecture decisions.

Fortunately, there is usually a translation from domain concerns to architecture characteristics.

For example: The below table shows some of the more common concerns and the corresponding “-ilities” that support them.

|  |  |
| --- | --- |
| Domain concern | Architecture Characteristics |
| Mergers and acquisitions | Interoperability, scalability, adaptability, extensibility |
| Time to market | Agility, testability, deployability |
| User satisfaction | Performance, availability, fault tolerance, testability, deployability, agility, security |
| Competitive advantage | Agility, testability, deployability, scalability, availability, fault tolerance. |
| Time and budget | Simplicity, feasibility. |

**Note** : Please note the above the concerns and key Characteristics would differ in priority or different also based on the domain area.

## Hardware Architecture Decisions

**[Optional – As per the applicability of the System under design]** Describe the system hardware Key Domain concerns based on the applicability.

## Software Architecture Decisions

**[Optional – As per the applicability of the System under design]** Describe the system software Key Domain concerns based on the applicability.

## Communication Architecture Decisions

**[Optional – As per the applicability of the System under design]** Describe the system communication Key Domain concerns based on the applicability.

# System Architectural Decisions

Architecture decisions define the rules for how a system should be constructed.

For example, an architecture decision could be that only the business and services layers within a layered architecture can access the database , restricting the presentation layer from making direct database calls. Architecture decisions form the constraints of the system and direct the development teams on what is and what isn’t allowed.

If a particular architecture decision cannot be implemented in one part of the system due to some conditions or other constraints , that decision (or rule) can be broken through something called a variance.

Most of the organization have variance models that are used by architecture review board (ARB). An exception to particular architecture decision is analyzed by the ARB and is either approved or denied based on justification.

## Hardware Architecture Decisions

**[Optional – As per the applicability of the System under design]** Describe the system hardware architecture based on the applicability.

## Software Architecture Decisions

**[Optional – As per the applicability of the System under design]** Describe the system software architecture decisions based on the applicability.

## Communication Architecture Decisions

**[Optional – As per the applicability of the System under design]** Describe the system software architecture decisions based on the applicability.

# System Design Principles

A design principle differs from an architecture decision in that design principle is a guideline rather than a hard -and-fast rule.

For example, A design principle states that the development teams should leverage asynchronous messaging between services within a microservices architecture to increase performance.

As architecture decision (rule) could never cover every condition and option for communication between services , so design principle can be used to provide guidance for the preferred method (in this case asynchronous messaging ) to allow the developer to choose a more appropriate communication protocol (such as REST or gRPC) given specific circumstance.

## Hardware Design Principles

**[Optional – As per the applicability of the System under design]** Describe the system hardware Design principles based on the applicability.

## Software Design Principles

**[Optional – As per the applicability of the System under design]** Describe the system software Design principles based on the applicability.

## Communication Design Principles

**[Optional – As per the applicability of the System under design]** Describe the system Communication design principles based on the applicability.

# System Design Trade-offs

Discuss the tradeoffs involved with the design chosen and the reasons for your choices. For example, an increase in security controls will likely entail a decrease in ease-of-use; an increase in the flexibility of a system typically entails a decrease in the simplicity of that system. For this reason, the designer must decide to put a higher value on some attributes over others.

For example:

Some areas to consider include:

* Flexibility
* Interoperability
* Performance
* Reliability and robustness
* Usability

## Hardware Design Trade-offs

**[Optional – As per the applicability of the System under design]** Describe the system hardware Design Trade-offs based on the applicability.

## Software Design Trade-offs

**[Optional – As per the applicability of the System under design]** Describe the system software Design Trade-offs based on the applicability.

## Communication Design Trade-offs

**[Optional – As per the applicability of the System under design]** Describe the system Communication Design Trade-offs based on the applicability.

# System Architecture - Detailed View

This chapter describes the proposed Architecture Design in detail. Provide the necessary information for the development team to Further drill down, design and development of the hardware components, system software components and Application software components leading to a functional product.

Every Component should map back to the Functional Requirements Document. These should be captured in the Requirements Traceability Matrix.

## Hardware Detailed View

In this section, provide enough information for the development team to build and/or procure the system’s hardware.

**NOTE:** If this section becomes too lengthy, consider placing it in the Appendix or reference it in a separate document. Add additional diagrams, if necessary, to describe each component and its functions.

Include the following information (as applicable):

* Cable type(s) and length(s)
* Connector specifications
* Details of hardware items, such as monitors, printers, servers, I/O devices, and the relationship to each other
* Hard drive/floppy drive/CD-ROM requirements
* Memory and/or storage space requirements
* Monitor resolution
* Power input requirements for each component
* Processor requirements
* Signal impedances and logic states

## Software Detailed View

This section covers the detailed specific details of the 4 main views (logical view, process view, deployment view and implementation view) of the system besides the critical use case view.

The following sections would identify the major software components of the system. In this section, provide enough detailed information for the development team to further outline the high-level design of the components and low-level design of the components.

For each Component, the following aspects of the information can be pondered upon:

* Narrative introduction to each module, its function(s), the conditions under which it is used (called or scheduled), processing, logic, interfaces to other modules, interfaces to external systems, security requirements, etc.
* Leveraging the Behavioral diagrams (or using charts, diagrams, flowcharts as appropriate) representation for outlining the Component processing, high level logic, flow of control, and algorithms
* Data elements, record structures, and file structures associated with component input and output as per the applicability
* Report layout as per the applicability
* For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages, such as DLLs (Dynamic Link Libraries) as per the applicability.

### Use-Case View

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.

#### Use case diagrams

Provide a detailed insights on the Use cases for the proposed system.

*For example:*

Diagram

Description automatically generated

#### Use case realization

Describe the use case realization based on the user journey.

The below sub sections can repeated based on the numbers of persona user journeys.

##### Persona A - Use Cases

*The below the basic template to capture the critical use cases based on the personas.*

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | <UC\_XXX\_YYY>: Use an active verb phrase to describe this scenario | | |
| Goal | Describe in one or two sentences the scope and content of the use case. Do not describe the flow of events, business or data validation rules | | |
| Business event | These are triggers that simulate activity within the business. They prompt the business to act, for example, at the interface point between the business and an external entity that it interacts with. Events may be internal triggers based on conditions or predefined time intervals. Events must be atomic (i.e. cannot be decomposed into two or more events) and observable. | | |
| Primary Actor(s) | Identify the actor initiating the use case | | |
| Actor(s) | Identify the secondary actor | | |
| Pre-conditions | Identify pre-conditions that must be met for the use case to be executed. For example, the use cases can start only when the system is in a certain state. | | |
| Postconditons | Describe how the use case is successfully completed. Discuss alternative ways that the use case may terminate successfully. | | |
| Failure Outcomes | Failure | Outcome | Condition leading to outcome |
|  | <Failure 1> Describe why the use case may terminate. |  | Describe the condition conditions under which the termination outcome occurs. |
|  | <Failure 2> |  |  |
| Flow of Events | Describe what the actor does and how the system responds.  The use case flow of events starts when the actor performs an action. An actor always initiates use cases. The use case describes what the actor does and what the system does in response. | | |
| Alternative Scenarios | Describe the series of events that should occur for the failure outcomes. | | |
| Business Rules | Identify business rules captured or referred to in this use case. | | |
| Traceability | Identify work products, models or documents that this use case is traceable to, for example, business rules, data validation rules, functional requirements, mockups, prototypes etc. | | |
| Inputs Summary | Identify data input by the actor | | |
| Output Summary | Identify data output by the system. | | |

##### Persona B - Use Cases

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | <UC\_XXX\_YYY>: Use an active verb phrase to describe this scenario | | |
| Goal | Describe in one or two sentences the scope and content of the use case. Do not describe the flow of events, business or data validation rules | | |
| Business event | These are triggers that simulate activity within the business. They prompt the business to act, for example, at the interface point between the business and an external entity that it interacts with. Events may be internal triggers based on conditions or predefined time intervals. Events must be atomic (i.e. cannot be decomposed into two or more events) and observable. | | |
| Primary Actor(s) | Identify the actor initiating the use case | | |
| Actor(s) | Identify the secondary actor | | |
| Pre-conditions | Identify pre-conditions that must be met for the use case to be executed. For example, the use cases can start only when the system is in a certain state. | | |
| Postconditons | Describe how the use case is successfully completed. Discuss alternative ways that the use case may terminate successfully. | | |
| Failure Outcomes | Failure | Outcome | Condition leading to outcome |
|  | <Failure 1> Describe why the use case may terminate. |  | Describe the condition conditions under which the termination outcome occurs. |
|  | <Failure 2> |  |  |
| Flow of Events | Describe what the actor does and how the system responds.  The use case flow of events starts when the actor performs an action. An actor always initiates use cases. The use case describes what the actor does and what the system does in response. | | |
| Alternative Scenarios | Describe the series of events that should occur for the failure outcomes. | | |
| Business Rules | Identify business rules captured or referred to in this use case. | | |
| Traceability | Identify work products, models or documents that this use case is traceable to, for example, business rules, data validation rules, functional requirements, mockups, prototypes etc. | | |
| Inputs Summary | Identify data input by the actor | | |
| Output Summary | Identify data output by the system. | | |

### Logical View

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages, and for each significant package, its decomposition into architecturally significant classes based on the applicability and describe their responsibilities as well as a few important relationships, operations, and attributes.

#### Overview

Describe the overall decomposition of the design model in terms of its subsystems and package hierarchy and layers.

##### Subsystems

Describe the overall decomposition of the design model in terms of its Subsystems and relevant functionality or use case support by the subsystem.

For Example :

This sub system provides the facilities that covers all the user management functionalities. Main use cases

that comes under this subsystem includes

 User login

 New user creation

 New Batch of students creation

 Edit profile and change password scenarios

Each user that interacts with the system is an actor of these use cases. Depending on the access levels, some actions will not be permitted.

##### Layering

Describe the overall decomposition of the design model in terms of its layers.

For Example :

Graphical user interface, text, application

Description automatically generated

#### Architecturally Significant Design Packages

Based on the applicability: For each significant package, include a subsection with its name, a brief description, and a diagram with all significant classes and packages contained within the package.

Based on the applicability: For each significant class in the package, include its name, a brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.

##### Class Diagram

For example:

Diagram, schematic

Description automatically generated

**Key insights**

 All users are derived from the super class called “user”. It’s because each user of the system has common attributes and behaviours such as Id, password, name, NIC etc.

 Each subject can have many number of lecturers and each lecturer can have many number of subjects. That the reason behind having a many to many relationship between lecturer and subject.

Both MCQ paper and the essay paper have the same characteristics except for

few attributes. Hence there exists a super class “Question” which contains the

common attributes and functionalities.

 MCQ paper consists of many number of MCQs. Hence an aggregation

relationship exists between MCQ paper and MCQ.

 Each student’s marks for any question paper is stored in a Result object.

 Login manager and Login classes are used to keep track of login times and dates of each user.

##### ER Diagram

For example:



**Key insights**

 Like in the class diagram each user has a super entity called ‘user’ that holds all

the major functionality.

 Each student has a batch parameter. Entity “batch” contains the details of a

particular batch (started date, batch id).

 Entity keywords is used to store the keywords of an essay question. These words are used to get rough statistical information about student’s behaviour towards an essay question.

 The relationship between student, MCQ paper and MCQ is a trinary relationship since the system has to keep track of answers for each question from each student.

 Each MCQ, paper, essay question has an associated “subject code” parameter.

 Active parameter in both user, subject represents whether the content is valid.

For an example if a lecturer leave the institute, the entry for that lecturer is not deleted, but marked DEACTIVATED.

 Relationship “Student Batch Eligible” between batch and subjects represents the relationship that contains all the subjects that a particular batch of students can enrol.

 Last login entity keeps track of user login details on order to make sure the

security.

### Process View

Describe the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.

#### System sequence diagrams

**NOTE:** This sub sections be repeated based on the critical system sequence diagram identified in the proposed system.

##### User logs into the system

For example:

When user inputs username and password, system checks for the validity and the matching of input values and redirects to the home page of user. (The home page depends on the user and user type). If the entered username and password is invalid, the user will be asked to enter them again. If the user has forgotten the password, he can click “forget password”.

When user clicks “forget password“, he can ask the system to send a mail to his email address, from which he can change the password to a new one. The user should be a valid user, and should have an email address that is saved in the system. If the user doesn’t have a pre-set email address, the user can’t continue this process and the user should contact the system admin and reset the password. If the user enters the wrong code, he can re attempt and get a new code.

Diagram

Description automatically generated

##### User invokes “forgot password”

For example:

Diagram

Description automatically generated

##### User changes the password

For example:

When the user goes to the “change password” section, he or she can change the password. For that the user should input the previous password again. The new password and confirm new password fields should be identical. The password should adhere to the security standards that are enforced by the application policy (more than 8 characters long, should contain alphanumeric and symbolic characters).

Or else when a user forgets the password and using the code that is sent via an email the user can access the “change password” section.

If the password doesn’t adhere to the rules of the system, it would ask for another password.

If the new password filed and “confirm new password” fields doesn’t match, it’ll ask to re-enter the passwords.

A picture containing chart

Description automatically generated

#### Activity diagrams

The activity diagrams for the main business functionalities (use cases) are as bellows.

**NOTE:** This sub sections be repeated based on the critical activity diagram identified in the proposed system.

##### Student enrolls to a subject

For example:

Diagram

Description automatically generated

##### Admin adds a new batch

For example:

Diagram

Description automatically generated

##### Admin adds a new subject

For example:

Diagram

Description automatically generated

### Implementation View

Describe the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.

#### Overview

Provide names and definitions for the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram showing the relationships between the layers.

*For example 1:*

**

*OES (online examination system) is a web application that follows the MVC architecture pattern. Main reason to use this pattern is to separate functions into layers thus improve the maintainability and reusability.*

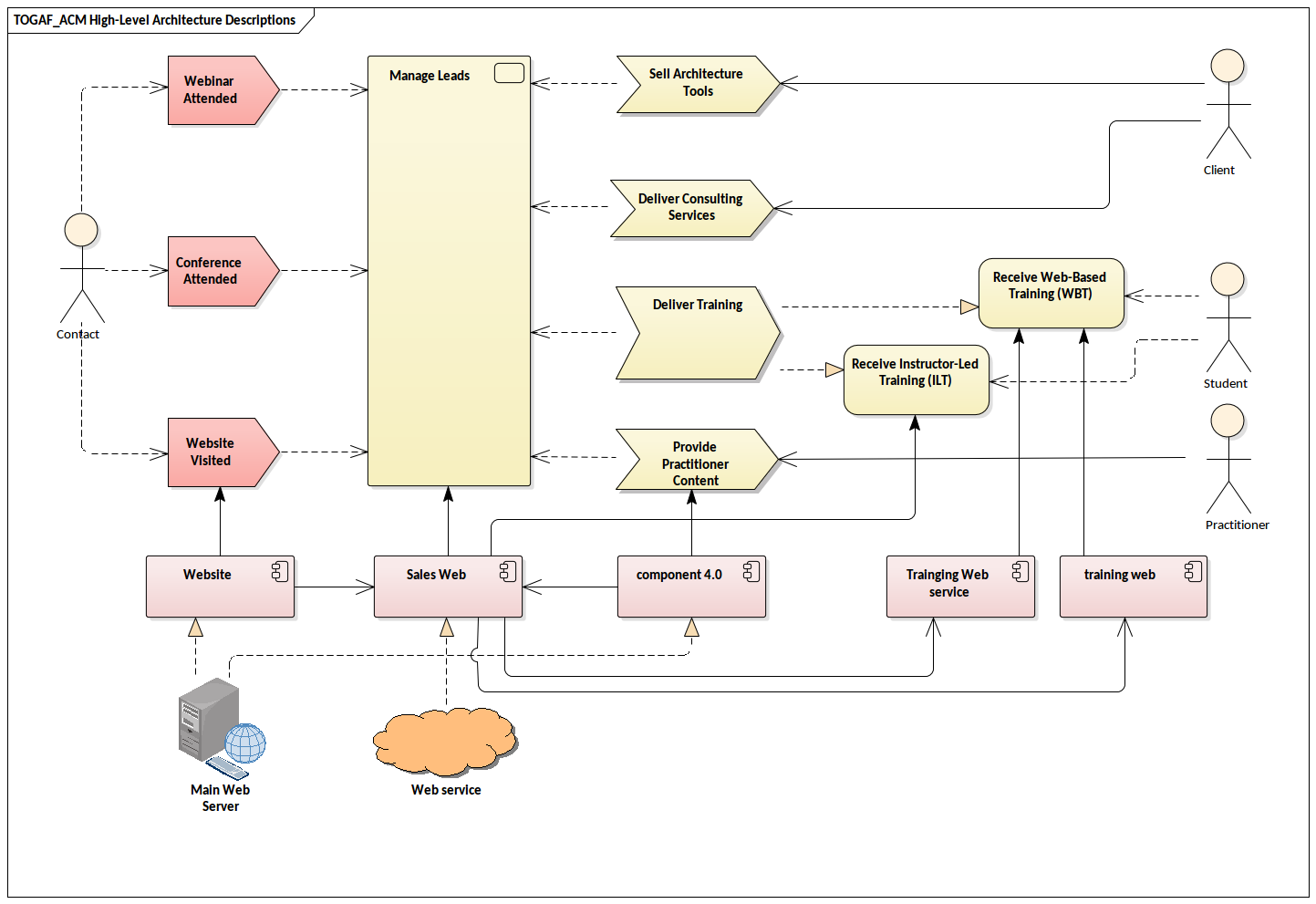
*View layer (AKA client layer) contains the graphical user interfaces (web pages). The actions of these web pages are handled by controller classes. Controller classes invokes and instantiates objects of model classes that contains the business login. Separating software packages in this manner reduces the complexity.*

*The model classes can be subdivided in to two layers. Business layer and the data access layer. Data access layer is manipulated using HIBERNATE framework. Business layer contains the main entity classes such as Student, Lecturer and etc.*

*The object relational mapping is done using the HIBERNATE framework. For each entity in the database there exists a class in the software that provides data access operations to that entity.*

*For example 2 :*

*An example of a high level architecture description diagram covering the core components of the baseline architecture of a project.*



#### Layers

For each layer, include a subsection with its name, a listing of the layer’s subsystems, and a component diagram.

*For example:*

*As mentioned above the software will be developed following the MVC pattern. For each sub layer, these 3 layers apply. Following is the representation of main subsystems of the system and their relevant main use cases.*

*Diagram

Description automatically generated*

### Deployment View

For example :

Being a web application, this system is hosted in a Cloud hosting space.

The example depicts a Google Enabled Smart Home in a serverless Internet-of-Things (IoT) environment. The architecture uses auto-scaled Messaging System implementing Streaming Analytics and  Event-Driven computation.

Graphical user interface, application, Word

Description automatically generated

## Communications Detailed View

Provide enough detailed information about the communication requirements to build and/or procure the communications components for the system.

**NOTE:** This section should provide enough detail to support the procurement of hardware for the system installation

Include the following information in the detailed designs (as appropriate):

* Details of servers and clients to be included on each area network
* Specifications for bus timing requirements and bus control
* Format(s) for data being exchanged between components
* Diagrams showing connectivity between components, data flow (if applicable), and distances between components.
* LAN topology

# External Interface View

External systems are any systems that are not within the scope of the system under development, regardless whether the other systems are managed by the development company or its client.

In this chapter, describe the interface(s) between the system under development (i.e. this system) and other external systems and/or subsystem(s).

## Interface Architecture

Describe the interface(s) between the system being designed and other systems. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide diagrams showing the communications path(s) between this system and other systems.

## Interface Detailed View

Provide sufficient detail about the interface requirements for the development team to format, transmit, and/or receive data across the interface.

Include the following information (as appropriate):

* Data format requirements; if data must be reformatted before it is transmitted or after incoming data is received. Describe the tools and/or methods for the reformat process
* Specifications for hand-shaking protocols between systems; content and format of hand-shake messages, timing for exchanging these messages, and errors handling
* Format(s) for reports exchanged between the systems
* Graphical representation of the connectivity between systems, showing the direction of data flow
* Query and response descriptions

Describe the individual data elements that the interfacing entity(s) will provide, store, send, access, and receive, such as:

1) Names/identifiers

* + - Data Element Name
    - Data Format/Length
    - Data Type
    - Definition
    - Non-Technical Name
    - Non-Technical Synonyms
    - Specifications
    - Synonyms

2) Range or enumeration of possible values (e.g. 0-99)

3) Accuracy and precision (number of significant digits)

4) Priority, timing, frequency, sequencing, and other constraints

5) Security and privacy constraints

6) Sources (setting/sending entities) and recipients (using/receiving entities)

Describe the data element assemblies (records, messages, files etc.) that the interfacing entity(s) will provide, store, and send, such as:

1) Names/identifiers

* + - Technical Name, e.g., data structure name
    - Non-technical Names, e.g. synonyms

2) Data elements

3) Medium/structure of data elements/assemblies

4) Visual characteristics (e.g. layouts, fonts, icons etc)

5) Relationships among assemblies

6) Security and privacy constraints

7) Sources and recipients

Describe the communication methods that the interfacing entity(s) will use for the interface, such as:

1) Communication links/bands/frequencies/media

2) Message formatting

3) Flow control (e.g. sequence numbering)

4) Data transfer rate

5) Routing

6) Transmission services

7) Safety

8) Security and privacy considerations

Describe characteristics of the protocols that the interfacing entity(s) will use for the interface, such as:

1) Priority/layer of the protocol

2) Packeting

3) Legality checks, error control

4) Recovery procedures

4) Synchronization

5) Status, identification, and other reporting features

Where appropriate describe other characteristics, such as physical compatibility of the interfacing entity(s) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

# Human-Machine Interface View

Describe the Human Machine interface (i.e. user interface) relative to the user. Additional information may be added if the suggested headings are inadequate.

## Interface Design Rules

Identify conventions and standards for designing the user interface.

## Inputs

Identify the input media used by the user (i.e. operator) for providing information to the system, such as data entry screens, optical character readers, bar scanners, etc.

Identify the messages associated with operator inputs, including the following:

* Form(s) if the input data is keyed or scanned for data entry
* Access restrictions
* Security considerations

## Outputs

Describe the system output design relative to the user. System outputs include reports, data display screens, query results, etc.

Identify the following, if appropriate:

* Access restrictions or security considerations
* Description of the purpose of the output
* Report requirements, including frequency for periodic reports
* Screen contents (provide a graphic representation of each layout. Define all data elements associated with the layout)

## Navigation Hierarchy

Provide a diagram of the navigation hierarchy that shows how a user moves through the user interface.

### Screen [x.1]

Provide the layout of all input data screens or graphical user interfaces. Provide a graphic representation of each interface, for example, a low-resolution screenshot. Define all data elements associated with each screen or GUI, or reference the data dictionary.

Label each data input screens and/or graphical user interface.

### Screen [x.2]

Provide a graphic representation of each interface, for example, a low-resolution screenshot. Define all data elements associated with each screen or GUI, or reference the data dictionary.

### Screen [x.3]

Provide a graphic representation of each interface, for example, a low-resolution screenshot. Define all data elements associated with each screen or GUI, or reference the data dictionary.

# Data View

This section outlines the design of the database management system (DBMS) and non-DBMS files associated with the system. For networks, detail the distribution of data and identify any changes to the logical data model that may occur due to software or hardware requirements.

Provide a Data Dictionary showing data element name, type, length, source, validation rules, maintenance, data stores, outputs, aliases, and description. Attach as an appendix.

## Database Management System

Describe how the database (RDBMS or NOSQL) will be designed, including the following information, as appropriate:

* Logical model; provide normalized table layouts, entity relationship diagrams, and other logical design information
* DBMS schemas, sub-schemas, records, sets, tables, storage page sizes, etc.
* Access methods (such as indexed, via set, sequential, random access, sorted pointer array, etc.)
* Estimate the database file size or volume of data within the file, data pages, including overhead resulting from access methods and free space
* Definition of the update frequency of the database tables, views, files, areas, records, and sets
* Estimates on the number of transactions the database may have to process

### Entity Relationship diagram - High level

*For example:*

Diagram, engineering drawing

Description automatically generated

### Implementation - High level

*For example:*

*Main design concentrations of this ER diagram is mentioned above under the section “Logical view”.*

*The database implementation for this ER diagram is done using MYSQL. The main tables are*

*1. Admin*

*2. Batch*

*3. Essay*

*4. Essay-Student*

*5. Keywords*

*6. Last-Login*

*7. Lecturer*

*8. MCQ*

*9. MCQ-paper*

*10. MCQ-paper-map*

*11. MCQ-paper-student*

*12. Student*

*13. Subject*

*14. Subject-batch-eligible*

*15. Subject-lecturer*

*16. Subject-student*

*17. User*

*Hibernate framework is used to automate the object relational mapping. From that a class is made for each*

*table in the database, that makes is possible to access the database.*

## Non-Database Management System Files

Describe all non-DBMS files including narratives on the usage of each file. Identify if the file is used for input, output, or both; identify temporary files; which modules read and write the file, etc.

* Identify record structures, record keys, indexes, and reference data elements within the records
* Define record length and blocking factors
* Define file access method, such as, index sequential, virtual sequential, random access, etc
* Estimate the file size or volume of data within the file
* Define the update frequency of the file; if appropriate, provide the estimated number of transactions per unit time, and the statistical mean, mode, and distribution of those transactions

# Size and Performance

Provide a description of the major size and performance characteristics of the system that impact the architecture, as well as the target performance constraints.

*For example :*

*Being a web application, OES will be hosted in the free web hosting space HEROKU.com. Users will not have to install any component to use this software system.*

*The size of the software is still not calculated, but it is in the range 100-200 MB. The MYSQL database will be hosted in another hosting space.*

*The client computers need to have a web browser in order to access the system. All the functionalities will be processed at the backend,*

*We also capture or describe the Cloud based aspects here and the aspects required to enable the software as per the applicability.*

# Quality

A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and others. If these characteristics have special significance, such as safety, security or privacy implications, **delineate them clearly**.

*Here we should consider the Quality aspects which are beyond the Architectures characteristics outlined in the Section 6 of this document.*

*For example.*

## Applicability to Common Services Architecture

*Describe how the application aligns with the defined Common Services Architecture. As specified by the* ***Company custom framework*** *that provides a practical interpretation thereof, the application modernization effort utilizes n-tier architecture and is based on a common, shared services approach.* ***This design maximizes standardization and re-use while minimizing maintenance impacts, technology, and database dependence.***

*Does the design ensure a separation between application logic, user interface, authorization support, and data storage? Is the system divided into areas of responsibility that are implemented as "service-based" components? Does the application delegate processing to external common services where such capabilities are available?*

# Architectural Mechanism

Describe how the system’s design supports anticipated system load, growth, availability, concurrency, and distribution requirements (including server, workstation, storage, bandwidth, and middleware).

## Analysis Mechanisms

Describe the following analysis mechanisms as they pertain to the project. **Analysis mechanisms represent a pattern that constitutes a common solution to a common problem across classes.** They can show patterns of structure, patterns of behavior, or both. **They are used during analysis to reduce the complexity of analysis, and to improve its consistency by providing designers with a shorthand representation for complex behavior.**

* **Persistency**: A means to make an element persistent (i.e. exist after the application that created it ceases to exist).
* **Distribution**: A means to distribute an element across existing nodes of the system.
* **Security**: A means to control access to an element.
* **Legacy Interface**: A means to access a legacy system with an existing interface.

## Analysis-to-Design-to-Implementation Mechanisms Map

For each of the analysis mechanisms in above section 17.1

* identify the corresponding design and implementation mechanisms.
* Design mechanisms should refine the analysis mechanisms based on the constraints imposed by the implementation environment.
* Design mechanisms provide an abstraction of the implementation mechanisms and bridge the gap between the analysis mechanisms and implementation mechanisms.
* The use of abstract architectural mechanisms during design allows consideration of the mechanisms without the details.
* Implementation mechanisms are used during the implementation process and should refine the design mechanisms given the implementation environment.
* Implementation mechanisms specify the exact implementation of the mechanism.
* They are bound to a specific technology, implementation language, vendor, etc.

|  |  |  |
| --- | --- | --- |
| Analysis Mechanisms | Design Mechanisms | Implementation Mechanisms |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

For example :

Some examples of implementation mechanisms include the actual programming language, COTS products, database, and the inter-process communication/distribution technology in use.

The following table is an example of an Analysis-to-Design-to-Implementation Mechanisms Mapping table as applied to a sample project:

|  |  |  |
| --- | --- | --- |
| Analysis Mechanisms | Design Mechanisms | Implementation Mechanisms |
| Persistency | OODMS (new data) | ObjectStore |
| Persistency | RDBMS (data from legacy database) | JDBC to Ingres |
| Distribution | Remote Method Invocation (RMI) | Java 1.1 from Sun |
| Security |  | Reverse Engineered Secure.java and UserContextRemoteObject components |
| Legacy Interface |  |  |

## Implementation Mechanisms

**[Optional]** As per the applicability, For each analysis mechanism listed in the table in Section 17.1, create a subsection that contains a Static View and Dynamic View of the design and implementation mechanisms.

### <Mechanism subsection>

#### Static View

Key Class diagram appears here followed by class descriptions as per the applicability.

#### Dynamic View

Sequence diagram

# System Integrity Controls

System designers should give consideration to integrity controls in order to restrict the loss, misuse, modification of, or unauthorized access to information that could affect the company, client, or its customers.

With this in mind, ensure that the following minimum levels of control are included:

* Ability to identify audit information by user identification, network terminal identification, date, time, and data accessed or changed
* Audit procedures to meet control, reporting, and retention period requirements
* Controls to restrict access of critical data items
* Verification processes for additions, deletions, or updates of critical data

# Conclusion

The conclusion sections describe conclusion drawn from the architecture models and the activity of architecting. It contains assessments, as well as past and current open issues.

The assessment section should attempt to assess the architecture against its requirements. The assessment section:

* highlights the advantages and known limitations of the chosen solution,
* describes how well non-functional requirements are met,
* lists any known inconsistencies within the architecture document20 (terminology, deviations from the requirements stated in Appendix A),
* optionally discusses possible directions for the evolution of the architecture,
* optionally documents any consistency analysis done across the architectural views.

For all non-functional quality and constraint requirements, the assessment section describes with which techniques and mechanisms and how well the requirement is met. For requirements that concern the behavior of the system (e.g., security, availability), the description of how the requirement is met.

Issues raised during the design of the architecture or during the review of documentation about the architecture should be recorded in the open issues section. The documentation for each issue should contain information about the origin of the issue and the tracking of it until it is resolved.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | Risk | Origin | Description | Owner | Resolution | Status |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# Appendix A

Attach any addition information that supplements the Architecture Document.

## Architecture Decision Records

∆ Provide evidence of consideration of alternatives and the rationale for the choices made.

∆ Record architecture decisions considered to be key to the architecture of <System of Interest>.

* Areas to consider to selecting key decisions include those:
* affecting key stakeholders or many stakeholders
* essential to project planning and management
* expensive to enforce or implement
* highly sensitive to changes or costly to change
* involving intricate or non-obvious reasoning
* pertaining to architecturally significant requirements
* requiring major expenditures of time or effort to make
* resulting in capital expenditures or indirect costs

∆ When recording decisions, the following information items should be considered:

* unique identifier for the decision
* statement of the decision
* correspondences or linkages concerns to which it pertains
* owner of the decision
* correspondences or linkages to affected AD elements
* rationale linked to the decision
* forces and constraints on the decision
* assumptions influencing the decision
* considered alternatives and their potential consequences

# Appendix B

Attach any addition information that supplements the Architecture Document.

## Requirements Traceability Matrix

Include a Requirements Traceability Matrix that traces components and data structures to the system requirements.

## Packaging and Installation

Outline any special considerations for software packaging and installation.

## Design Metrics

Describe all metrics to be used during the design activity.

## Glossary of Terms

Identify all terms that establish meaning within the context of the plan.

|  |  |
| --- | --- |
| Term | Meaning |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table 1 — Glossary of Terms