

**Technical Design Document**

**<Project Name>**

**Revision History**

| **Version** | **Revision Date** | **Description of Changes** | **Modifier** | **Approver** | **Approval Date** |
| --- | --- | --- | --- | --- | --- |
| 1.1 | 05/30/2019 | Initial Revision | Claude Shacklett | Corey Doyle | 06/12/2019 |
| 1.2 | 06/15/2019 | Final template changes | Larissa Khon | Claude Shacklett | 06/15/2019 |
| 1.3 | 09/29/2020 | Added Confidentiality Statement and Legal Disclaimer, updated TOC | Larissa Khon | Claude Shacklett | 09/29/2020 |
| 1.4 | 05/11/2024 | Annual Document Review | Claude Shacklett | [Corey Doyle](mailto:cdoyle@ctacorp.com) | 05/13/2024 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Table of Contents**

[**CONFIDENTIALITY STATEMENT AND LEGAL DISCLAIMER**](#_ff8w39bfosym) **5**

[**1. Title**](#_vf4yvlg6u3h) **6**

[**2. Introduction**](#_1fob9te) **6**

[**3. System Architecture**](#_3znysh7) **6**

[3.1. Physical Architecture](#_2et92p0) 6

[3.2. Logical Architecture](#_tyjcwt) 7

[3.3. User Interface](#_3dy6vkm) 8

[**4. System Details**](#_1t3h5sf) **8**

[4.1. Components](#_4d34og8) 8

[4.2. Reusable Components](#_2s8eyo1) 9

[4.2.1. User Interface Elements](#_17dp8vu) 9

[4.2.2. Database Entities](#_3rdcrjn) 9

[4.2.3. Resources](#_26in1rg) 9

[4.3. Interrelationships and interfaces between the](#_lnxbz9) Components 9

[4.4. Integration](#_35nkun2) Strategy 10

[4.4.1. Product Integration Phases](#_1ksv4uv) 10

[4.4.2. Product Integration Sequence](#_44sinio) 10

[4.4.3. Interface Requirement](#_2jxsxqh) 10

[4.4.4. Schedule](#_z337ya) 10

[4.4.5. Tools Required](#_3j2qqm3) 10

[4.4.6. Environment](#_1y810tw) 10

[**5. Data Model**](#_4i7ojhp) **10**

[5.1. Data Dictionary](#_2xcytpi) 10

[5.2. E-R Model(s)](#_1ci93xb) 10

[**6. Design Considerations**](#_3whwml4) **11**

[6.1. Assumptions](#_2bn6wsx) 11

[6.2. Constraints](#_qsh70q) 11

[6.3. Limitations](#_3as4poj) 11

[6.4. Alternate designs considered](#_1pxezwc) 11

[**7. Non-Functional Requirements**](#_49x2ik5) **11**

[7.1. Goals and Guidelines](#_2p2csry) 11

[**8. Additional Information, if any**](#_147n2zr) **11**

# CONFIDENTIALITY STATEMENT AND LEGAL DISCLAIMER

The information in this document is confidential, privileged, and only for the intended recipient's information. It may not be used, published, or redistributed without prior written consent from CTAC. If you are not the intended recipient, you may not disclose or use the information in this document in any way. Any review, retransmission, dissemination, or other use of, or taking any action in reliance upon this information by persons or entities other than the intended recipient is prohibited. If you got access to or received this document in error, please contact the owner and delete the document from all electronic devices.

# 

# 1. Title

Project Code:

Project Name: < *Project name to which this document belongs* >

System Name: < *Name of the system to which this component will be added* >

Component Name: < *Component for which High-level design applies* >

# 2. Introduction

< *Provide an introduction to the system for which this High-level design is done.*

*This can include a brief description of the application functionality, proposed environment, implicit requirements, etc. The intent is to provide a background to the proposed High-level design. Optionally, the designer can add references to the acceptance criteria that guide the High-level design.*

*Identify some of the key issues that the proposed high-level design will resolve. This can involve items like re-usability, maintainability, testability, performance, security, scalability, extensibility, high availability, fault tolerance, load balancing, backup and recovery, portability, safety, etc.*>

< *Give a brief outline of what is presented in the subsequent sections\** >

\*Note – The subsequent sections of the template outline possible sections that can be covered in High-level design. Depending on the project requirements, one or more of the sections may be not applicable

# 3. System Architecture

Any system consists of three types of architectures.

## 3.1. Physical Architecture

< *\*Present the architecture diagram for the development and deployment environments* >

Physical architecture – the layout of the hardware and software elements (external to the application) and their interaction patterns. In addition to the core application information, this architecture usually helps explain system-level features like load balancing, high availability, extensibility, etc.

**Hardware elements**

* + Identify the hardware elements that are part of the physical architecture. These can be Machine details
  + Disk and memory requirements
  + Network elements (like routers, etc.)
  + Disk sub-systems like (EMC2, NAS, etc.)

**Software elements**

Identify the software elements that are part of the physical architecture. These are items like

* + Operating systems (Windows NT, 2000, Unix, Solaris, etc.)
  + Web servers
  + Application servers
  + Database servers
  + Third-party tools (ReSharper, Crystal Reports, Install Shield, etc.)

\* Note – Often, it is not necessary that the whole application system or its full set of features be represented through a single physical architecture. If the need is felt for multiple views of the application system, then multiple architecture diagrams can be presented with the appropriate mention of hardware and software elements.

< *Present a brief note on the rationale for selecting the above-mentioned physical architecture and outline the specific requirements and/or acceptance criteria that are satisfied by the use of this architecture* >

< *If there is a possibility for alternate architectures, present the same with a mention of the possible design trade-offs* >

## 3.2. Logical Architecture

Logical architecture – also called the software component architecture or the functional architecture, outlines the various software components that constitute the system and their interactions at a macro level. In addition to the functional description of the application, this architecture usually helps explain modularity, re-usability, scalability, extensibility, etc.

Logical architecture specifies a macro-level layout of the various objects that constitute the application, emphasizing their interactions.

Logical architecture can outline high-level components

It can outline an high-level abstraction of the application (avoid implementation issues) like the Conceptual object model that describes the

* + Application Tiers
  + Application Layers
  + Interaction between the tiers
  + Interaction between the layers

< *\*Provide the logical architecture diagram and explain the various elements of the diagram* >

\*Note – Often, it is not necessary that the whole application system or its full set of features be represented through a single logical architecture diagram. If the need is felt for multiple views/ sub-views of the application elements, then multiple logical architecture diagrams in a top-down fashion can be presented.

< *Provide details of interfacing with the third-party software tools, if any, in this section* >

## 3.3. User Interface

< *If user interface elements are associated with the system, present details of those elements. User interface details can include items like:*

1. *Branding information and its location*
2. *Navigation details and location*
3. *Usability aspects like the use of personalization, registration, etc.*
4. *Security items, etc.*

*These are macro-level details and should not be delved into in terms of implementation.*

*Wherever it is appropriate, please make a mention of the explicit/ implicit requirement and/ or acceptance criteria being satisfied by items presented in this section* >

# 4. System Details

< *A logical architecture diagram in itself will not provide a detailed view of the system. It is supplemented with the following (The list is indicative)* >

**NOTE: The following four sections will repeat for every component in the application**

## 4.1. Components

< *A component is defined as any code which is in executable format. It could be a DLL, Java Class file, Java Bean, Java Package, COM DLL/EXE, .Net Assembly, and any process (EXE, etc.)* >

## 4.2. Reusable Components

< *Pertinent details on reusable components need to be provided at appropriate places during the project life cycle phases* >

| **Component Name** | **New Development** | **Reuse** | **Buy (COTS)** | **Comments** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

No. of components newly Developed:

No. of Components Reused:

No. Of Components bought (COTS):

Total No. of Components

### 4.2.1. User Interface Elements

< *User Interface elements include any type of User Interface elements used by or defined within its parent Component/ Process. These include, but are not limited to, Web Pages (HTML/ ASP/ JSP/ ASPX/ etc.), Windows Forms, etc.* >

If the UI elements are the same across the application, then define this section once.

### 4.2.2. Database Entities

< *Database entities include database-related elements like Tables, Views, Stored Procedures, SQL queries, etc.* >

Table schema for the component-related database objects.

### 4.2.3. Resources

< *Resources include file types like images, language-specific resource files containing string resources, etc.* >

Any files or other type of resources used in the application other than the database entities.

## 4.3. Interrelationships and interfaces between the components

< *A logical diagram representing the inter-relationships between the various components listed in the previous section* >

## 4.4. Integration strategy

### 4.4.1. Product Integration Phases

< *Product integration activities are often taken incrementally through the project life cycle. List here the milestones of this process* >

### 4.4.2. Product Integration Sequence

< *Describe the Product Integration sequence with its advantages and disadvantages. List the components that will be integrated in the order of the defined Integration sequence* >

### 4.4.3. Interface Requirement

< *This section describes interface requirements for integration activity and after that, i.e., during system testing, installation, etc. Component ID, Interface with, Interface category, Interface description* >

### 4.4.4. Schedule

< *This section describes the schedule for integration activity. This section also needs to include high-level milestones. A reference to the project schedule (.mpp) file can be given. The critical path could be identified. These activities could start from the requirement stage and further evolve in the integration stage. This may end at the deployment stage* >

### 4.4.5. Tools Required

< *Tool required for product integration* >

### 4.4.6. Environment

< *Define environment that is used for product integration* >

# 5. Data Model

< *Provide details of the data model in this section* >

Data model usually involves the following

## 5.1. Data Dictionary

Data dictionary gives the list of all tables used in the system and their brief description. (For new development only)

## 5.2. E-R Model(s)

< *Provide details of the data model in this section* >

# 6. Design Considerations

## 6.1. Assumptions

< *State the assumptions made while making the high-level design. These are issues that, if not found correct later in time, can impact the system design and its implementation. Mention as none if no assumptions are made* >

## 6.2. Constraints

< *State the constraints the system’s high-level design will impose on other parts of the application/environment. Constraints can be concerning elements like third-party tools, specific design patterns, etc., used for this system* >

## 6.3. Limitations

< *State the limitations that can be imposed by the system’s high-level design on the application/ environment. Usually, these are design trade-offs made while designing the system* >

## 6.4. Alternate designs considered

# 7. Non Functional Requirements

## 7.1. Goals and Guidelines

| **Goal** | **Reason for its Desirability** |
| --- | --- |
| Emphasis on speed versus memory use |  |
| Working, looking, or “feeling” like an existing product |  |
|  |  |
|  |  |

# 8. Additional Information, if any

< *If the designer has made any additional application-specific assumptions or foresees any limitations with the current design (which are outside section 6), he can note such things in this section. This section is for any item that the designer wants to mention that is important and does not fit into the template.* >