Detailed Design

For <<Project Name>>

Version x.y

Votary Softech Solutions Pvt. Ltd.

Plot No: 76, Lumbini layout,   
Near Euro school,   
Gachibowli-I (V), Hyderabad,  
Telangana - 500032,   
India.

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version (x.y) | Date of Revision | Description of Change | Reason for Change | Affected Sections | Approved By |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Approval History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version (x.y) | Prepared By | Reviewed By/Date | Approved By/Date |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Overview 4](#_Toc382397411)

[1.1 Scope of this Document 4](#_Toc382397412)

[1.2 Definitions and Acronyms 4](#_Toc382397413)

[1.3 Platform and Tools 4](#_Toc382397414)

[1.4 References 4](#_Toc382397415)

[1.5 Assumptions 4](#_Toc382397416)

[1.6 Design Constraints 4](#_Toc382397417)

[2 Design Approach/ System Overview 5](#_Toc382397418)

[3 Alternative Design Approaches Considered 5](#_Toc382397419)

[4 Module Design 5](#_Toc382397420)

[4.1 Module Description 5](#_Toc382397421)

[4.2 Class Diagram 5](#_Toc382397422)

[4.2.1 Class 1 (say, minorb.orb.Any) 6](#_Toc382397423)

[4.2.2 Class 2 6](#_Toc382397424)

[4.2.3 Class 3 7](#_Toc382397425)

[4.3 Database Design 7](#_Toc382397426)

[4.3.1 Table 1 7](#_Toc382397427)

[4.3.2 Table 2 7](#_Toc382397428)

[4.4 Interaction Diagram 7](#_Toc382397429)

[4.5 State Transition Diagram (Optional) 7](#_Toc382397430)

[4.5.1 State Transition Table 8](#_Toc382397431)

[4.6 Error & Exception Handling 8](#_Toc382397432)

[5 Unit Test Recommendations 8](#_Toc382397433)

[6 Traceability Matrix 8](#_Toc382397434)

[Appendix 1. Alternate Designs Considered 9](#_Toc382397435)

# Overview

<Provide a brief overview of the project. >

## Scope of this Document

<Describe here briefly the scope of the document and it’s purpose. Purpose of this document typically is to do the high level design by identifying the different modules and their interfaces. Here you can describe with respect to the component architecture document, which are the modules that you’re planning to describe in this document and which are out of the scope of this document. >

## Definitions and Acronyms

The terms in use in the document are explained below.

|  |  |
| --- | --- |
| Acronym | Description |
|  |  |

## Platform and Tools

<Can optionally list out any of the third party tools being used and the platform on which the system is being designed for. >

|  |  |
| --- | --- |
| Tool Name | Purpose for which the Tool is proposed to be used |
|  |  |
|  |  |

## References

The documents referred to in the preparation of this document or those on which this document is based, are as follows: <specify the documents referred to and also the relevant documents, such as SRS, any customer supplied documents, etc. >

## Assumptions

The following assumptions are made during this design:

<This gives the assumptions being made in the design of the system. These assumptions may be made due to limited functionality or may be because of implementation issues. Reasons for the assumptions must be indicated, where relevant. >

## Design Constraints

The following are the Constraints taken into account while preparing this document:

< Specify the design constraints. Design constraints could be:

External integration requirement (for open architecture)

Customer specific architecture. >

# Design Approach/ System Overview

< This section gives a brief overview of the system being designed. List here the different modules that are being dealt about in this document and their functionality in brief.

Module 1:..

Module 2:…

A good idea would be to take the module decomposition diagram from the High Level Design Document and highlight the modules that are being addressed in this document. >

# Alternative Design Approaches Considered

<Describe here the different alternatives considered. Also explain why the selected concept was selected. Describe the capabilities and limitations of the selected concept and why the limitations of the concept will not adversely affect meeting the requirements. Provide more details as required in Appendix A. Refer the same here. >

# Module Design

<Duplicate this section for each of the modules that are being addressed in this document. >

## Module Description

< Describe here what the module is, where it fits in into the overall architecture and the complete functionality of this module. Describe what are the services that are provided by this module. Here the reference to the high level design document or requirement specifications document may be included as appropriate. >

## Class Diagram

<This section includes a static class diagram of the module. All the classes comprising the module are to be shown as one or more UML class diagrams. A sample class diagram is shown below. Can specify if any of the standard design patterns have been used. >

< A class diagram is to be accompanied by a description of all the classes shown. The section below can be repeated for all the required classes. >

### Class 1 (say, minorb.orb.Any)

< This section can contain all the details of the class. It can include details like where the class fits in the module and the particular functionality it incorporates. This section also contains description of the important class attributes and operations. The operation descriptions may optionally contain pseudo code. >

<

Data Members

private java.lang.Object value :

This provides the data container for the any. All primitive data types are converted into the corresponding primitive data type classes and upcasted to value. For the constructed data types like struts, the helper classes first Company Namehal them and the value holds the Company Namehaled streams.

private org.omg.CORBA.TypeCode typeCode :

Holds the typecode of the data contained

Operations

public boolean equal(Any a) {

// return false if the typeCode.equal() returns false

// Evaluate value.equals()

}

public Object extract\_object() throws CORBA.BAD\_OPERATION {

if( typeCode.kind().value() != TCKind.tk\_objref)

throw new BAD\_OPERATION();

return (org.omg.CORBA.Object) value;

} >

### Class 2

### Class 3

## Database Design

< If the module comprises classes which interact with a database, the database specific design is to be discussed here. If for example, the back end database is a relational database, this section describes all the tables that are maintained during the process lifetime. References can be given to the particular classes that interact with each of the tables. >

### Table 1

< This Section would include a description of the table and the fields associated with the table. >

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table | Field Name | Field Type | Field Length | Remarks |
| Table 1 | Field1\* | Integer | 8 |  |
|  | Field2 | String | 30 |  |
| Field3 | Integer | 4 | To link with Table 2 |
| Table 2 | FieldA\* | Integer | 4 |  |
| FieldB | Integer | 4 | Links Table 1 |
| FieldC | Float | 3,2 |  |

<A \* means that these fields would be keys to the table. >

### Table 2

……

Or if the database being used is an object oriented database, this section may include a class diagram of the database classes, i.e. the schema of the database for this particular module.

## Interaction Diagram

< This section includes the sequence diagrams appropriate to the module. Operation invocations from other modules are to be shown through appropriate actors. The sequence diagrams essentially illustrate all the object interactions of the module. While the class diagrams give a static representation of the module, the sequence diagrams give more of a dynamic view. The sequence diagrams may depict a particular operation sequence which provides the module functionality or the series of initialization actions performed or how the threads are spawned in the module etc. >

## State Transition Diagram (Optional)

< State Transition Diagrams will be used in event-driven systems to indicate valid state transitions. >

### State Transition Table

< State Transition Tables are useful in event driven system and can be used in conjunction with State Transition Diagrams to depict State Transitions. While the diagram gives a pictorial representation and a snap-shot view of the transitions, the table would give an implementation view for a detailed look. >

## Error & Exception Handling

< In this section, describe what are the kinds of internal and external errors that can happen in the module and what is the action taken for each one of them. Also describe here the limits of input data and what action is taken even if there is a wrong input to the module. Also describe what are the different warnings (if any) generated by this module.

Also describe what are the exceptions that are handled in this module and what is propagated to the higher modules. >

|  |  |  |
| --- | --- | --- |
| Error Code | Error Description | Action taken |
| Error No | Describe the error | This can be, you return an error code to the calling module or you display an error message and stop processing or anything required to handle the error. |
|  |  |  |

< Also describe here a set of error codes that this module will be sending to other modules. >

# Unit Test Recommendations

<Designer can give test recommendations if the designer feels certain areas needs attention at the time of unit testing. >

# Traceability Matrix

<Give Reference to updated Traceability matrix. Traceability matrix to be updated now with High Level Design references >

# Appendix 1. Alternate Designs Considered

**DESIGN 1**

Document the alternate design in detail.

**DESIGN …N**

Document the alternate design in detail.

**RATIONALE FOR CHOOSING A DESIGN**

<Document the Rationale for choosing one particular design in detail. >