# Introduction

VenAqua Visualization Report application is a middleware application which interacts with CORE DB 2.0 in the existing VenAqua architecture and front-end application. VenAqua Visualization Report fetches required data from CORE DB system and manipulates it as needed to visualize in the front end application. It publishes SOAP based Web Services API, through which an internal components of a VenAqua architecture or a third party applications can communicate to VenAqua Visualization Report application to get a desired data.

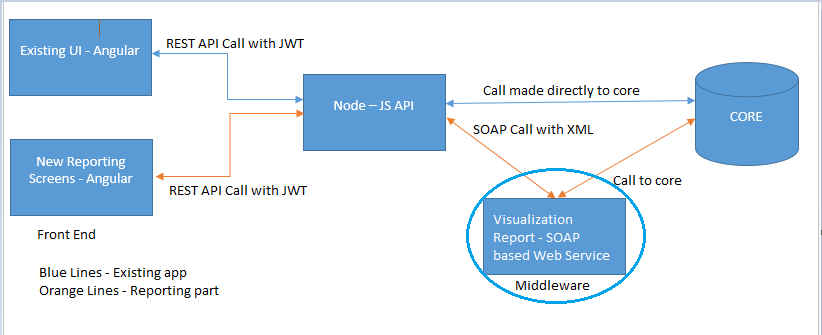
VenAqua Visualization Report application collects and manipulates data from the CORE DB system, such as

* Water Consumption from different sources in a site
* Water Consumption of individual house
* Various water types and their demand in a site
* Pump’s ON/OFF states and their yields
* Various water sources and its performance

And etc.

# Communication Architecture

The below diagram shows how the Visualization Report application, can be invoked from the front end application.



In this architecture, the third party applications must be authenticated by the middleware application in order to communicate further. In the subsequent calls, the third party application must send their access token which is received in the authentication call.

# Visualization Report Architecture

Visualization Report application has divided into multiple small modules and each module have their own roles to prepare data as requested by the front-end or third party applications.

**General Utility**

**SOAP  
Web Services**

**CORE ENGINE**

**DB  
Handler**

**Web Services**

**CORE DB  
2.0**

**Token   
Manager**

**Authentication  
Handler**

**Data Preparer**

**Error Handler**

## General Utility

General utility module contains classes which are related to utility purpose. These classes will be used across all the modules in this architecture.

## SOAP Web Services

This module publishes SOAP based web service API with multiple endpoints. Each endpoint is used for different purpose.

**<host>/report** – is a service path for this API.

**<host>/report?wsdl –** is a path for API document in WSDL format.

Service provides multiple operations to retrieve various types of data. Users can fetch API document to know all the available operations and service related details. The supported operations are,

1. **Login** – is an authentication operation which validates user credentials and returns access token for further communication.
2. **PumpYield** – To retrieve available pumps ON / OFF duration with its yield value for the requested period.
3. **SiteUsageByWaterSource** – Operation returns site’s water consumption from various water sources for the requested period.
4. **SiteWaterMap** – Returns water consumption of all the houses in a site for the requested date range and time period. Consumer of this API can request data for a specific house for a day.
5. **HighUsers** – Clients can use this operation to determine the high users in a site for a date range.
6. **SiteTrendByWaterSource** – Compares all the water sources and their performances for the given date range in terms of their individual yields.
7. **SiteUsageByBlockLevel** –Clients can fetch the individual residence consumption in their respective blocks for the given date range.
8. **SiteDemandByWaterType –** Clients can get different varieties of water with their demand on an hourly basis for a given date range.

This module transfer control to Core Engine module to fetch required data from Core DB2.0 system for all requests.

## Core Engine

Core Engine is a centralized place which communicates with all other modules for different actions.

* Once the request is received, the core engine module validates user credentials or access token with the help of Authentication handler module.
* Uses DB Handler module to interact with the Core DB 2.0 system to fetch requested data and prepares response data by using Data Preparer module.
* Error Handler module will be used to send error responses, in case of any exception while processing client request.

## Authentication Handler

The Authentication handler validates the user credentials during the login call and also checks whether the access token is valid and its expiration with the help of Token manager module. The authentication handler acknowledges back to the core engine module, to process further for a particular request. Authentication failed response will be sent if the user credentials or the access tokens are not valid.

Authentication handler will expose java interface, application can use this interface to extend or modify default behavior of Authentication handler module. This will handy for future changes in terms of authentication process.

## Token Manager

It manages the access tokens for a user by interacting with the core db2.0 module. This module mainly invoked by the Authentication handler module to complete the authentication process.

Token manager will be invoked only for the login request to create a fresh token. For all other scenarios, token passed along with the request will be validated.

For successful token validation, the control will be transferred to core module from the authentication module.

## DB Handler

It uses JDBC driver libraries to communicate with the Core DB2.0 system. Major role for this module is manages connection with the database and executes SQL queries to fetch the data. Core Engine module uses this to prepare desired data for a request.

## Data Preparer

It handles data fetched with the help of DB handler module to prepare response content. It handles both data fetched from DB handler module as well as Error handler module in case of any exception in the process of request.

## Error Handler

In all error scenarios, the control will be transferred to this module and its take care of handling the exceptions. It categories exceptions into multiple levels and provides error codes based on the functional and category aspect. Finally, it sends error response content to core module to send it to client.

Application provides external hooks to modify error response logic defined in the default error handling logic.

# Milestones and Deadlines

|  |  |  |
| --- | --- | --- |
| Milestones | Expected Duration | Delivered On |
| SOAP mock server which exposes above mentioned services with sample data | 4-5 Days |  |
| Node JS based mock server   publishes REST API and interacts with SOAP server | 5-6 Days |  |
| Base Framework and Pie Chart  Base Framework contains modules like core, DB handler and data preparer also provides real data for Pie Chart. | 7-10 Days |  |
| Tree, Bubble & Usage Chart provides real data for Tree, Bubble and Usage Chart. | 3-4 Days |  |
| Gantt & Sparkline Chart provides real data for Gantt and Sparkline Chart. | 4-5 Days |  |
| Water Map Charts provides real data for Heat Map and Histogram Chart. | 2-3 Days |  |
| Authentication and Token Management with login service and access token validation | 3-4 Days |  |
| Overall | 28-37 Days |  |

**Price estimation**: 2000 per day