**ECOMMERCE - REPORT GENERATION SYSTEM**

**Technical Architecture and Solution**

**Document:** ECOMMERCE - REPORT GENERATION SYSTEM- Technical Architecture and Solution

**Author:** Srimanikandan

**Version:** 0.1

**Date:** 20-09-2023

|  |  |
| --- | --- |
| **Document Summary** | |
| **Author** | Srimanikandan |
| **Creation Date** | Sep 20, 2023 |
| **Version** | 0.1 |
| **Release Date** |  |

**Document Reference**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Change Reference** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Distribution List**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Company** | **Project Role** | **Document Role** |
| X | AAA | Vice President Engineering | Approver |
| Y | Zuci Systems | Technical Manager | Reviewer |
| Z | Zuci Systems | Technical Lead | Author |

Table of Contents

[1. Introduction 4](#_Toc14493)

[1.1 Architecture Components Overview 4](#_Toc31814)

[1.2 Design Principles 5](#_Toc14840)

[2. ABC Technical Design 5](#_Toc18573)

[2.1 Architecture 5](#_Toc24767)

[2.2 Architecture Description 6](#_Toc15235)

[2.3 Component Level Design 7](#_Toc13210)

[2.3.1 Data Stores 7](#_Toc6879)

[2.3.2 Presentation layer 8](#_Toc31845)

[2.3.3 Service Layer 8](#_Toc16386)

[2.3.4 Business Service Layer 8](#_Toc29198)

[2.3.5 Data Layer 8](#_Toc1264)

[3. Technical Stack Layout 8](#_Toc12847)

[4. Functional Layout 9](#_Toc5618)

[4.1. Class Diagram 10](#_Toc4250)

[4.2. Entity Relationship Diagram 13](#_Toc13691)

[5. Non-Functional Requirements 14](#_Toc13371)

[5.1 Security 14](#_Toc11271)

[5.2 Performance 14](#_Toc29678)

[5.3 Multilingual 14](#_Toc11269)

[5.4 Audit Trail 14](#_Toc15176)

[6. Glossary 15](#_Toc29204)

# **Introduction**

Ecommerce - Report Generation System is a project that deals with the reports of the Online Shopping company SportsZone which is providing good quality products like accessories, equipments, clothings, nutritions etc,. related to sports.

This project includes two use cases namely the customer and the admin. Customer can browse, view, buy products which are all available in the inventory. Customer will be able to buy products via cart option. Admin can update the order status and customer details. Admin can view the various reports i.e Customer reports, Product reports, Order reports.

Reports are generated with real time data which will be stored on to database by integrating with the customer via webapplication.

The purpose of this document is to add the necessary detail to the current business requirement to represent a suitable architecture and design for coding. This document is also intended to provide a high-level design prior to coding and can be used as a reference manual for how the modules interact at a high level.

## Architecture Components Overview

This section of the document describes the technical components in Ecommerce - Reports Generation system for AAAA. The XYZ solution presents all the design aspects of the application architecture and the details of the business components. Apart from the proposed solution, this document also includes deployment options and the non-functional requirements identified for this project. The architecture components considered have been listed as below.

| **No** | **Component Name** | **Description** |
| --- | --- | --- |
| 1 | Application Architecture | This section explains the basic layout of the architectural design proposed for this application. |
| 2 | Technical Stack Layout | This section explains the technology stack proposed for the various layers of the application. |
| 3 | Functional Layout | This section explains the functional layout for the various components of the application and details on how they interact with each other. |
| 4 | Process Layout | This section of the document explains the development process during the various stages of the application development. |
| 5 | Non-Functional Requirements | This section of the document lists the identified non-functional requirements which are essential for the seamless working of the application. |

## Design Principles

These principles will help us to create a system architecture that adheres to the proven principles, minimizes the cost, helps in incorporating any changes in the requirements and promotes re-usability and scaling the application.

* Separation of concerns
* Single Responsibility principle
* Principle of Least Knowledge
* Do not Repeat Yourself (DRY) Principle
* Minimize upfront design

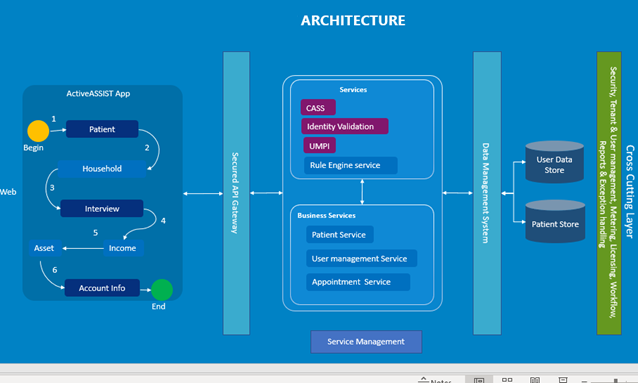
The ability to reuse the components is one of the key principles followed in this approach. In this design, we have identified the components which can act as a Platform for this application and provides an ability to extend it, for different devices considering the future requirements to extend the application.

# **SportsZone Technical Design**

This section briefs on the solution architecture which forms the core of the design. It depicts the global organization of the software system considered for building the Ecommerce - Report generation system - SportsZone. It is a set of high-level decisions that will strongly influence the integrity and structure of the system. The proposed high-level architecture of the SportsZone solution is as shown and explained below.

## Architecture

This section of the document gives the main objectives of the SportsZone application to develop a scalable and highly configurable application. The components have been designed to provide an application platform for enabling scalability to build extendable features.



## Architecture Description

This representation provides the logical view of various components which are involved in the ABC system.

ABC UI enables the user for financial assistance screening and workflow management. The solution is designed to identify alternative funding sources for patients, support patient advocacy through counseling and assistance, and ensure that self-pay patients are managed consistently.

The business services like patient service, User management service, Appointment Service hold the core functionalities of the ABC application. The existing third-party API services like CASS, UMPI and Identity Validation are used for validating patient details.

Processing Service will validate the deliverability of the address and standardize it according to USPS standards.

**Identity Validation service** is an optional feature that is turned on for certain clients. It would be configured based on the client preferences and permissions. This involves calling a 3rd party service to validate the patient’s residency at the address provided, their DOB, SSN, and phone number.

**UMPI Service** is a Universal Master Patient index provided through a 3rd party service. This value is assigned to the patient and used to uniquely identify them across all the client sites. If done inside of ABC, we will provide a service or API to call to generate this value for the new patient record.

The Secured API gateway will authenticate the valid users to access the service from the User Interface. This service is responsible to handle each request from the user interface and provide the necessary response for them. This layer will communicate with the business service layer to perform certain operations/functionality.

The Rule Engine service will provide the eligibility criteria based on the user inputs to determine the financial assistance.

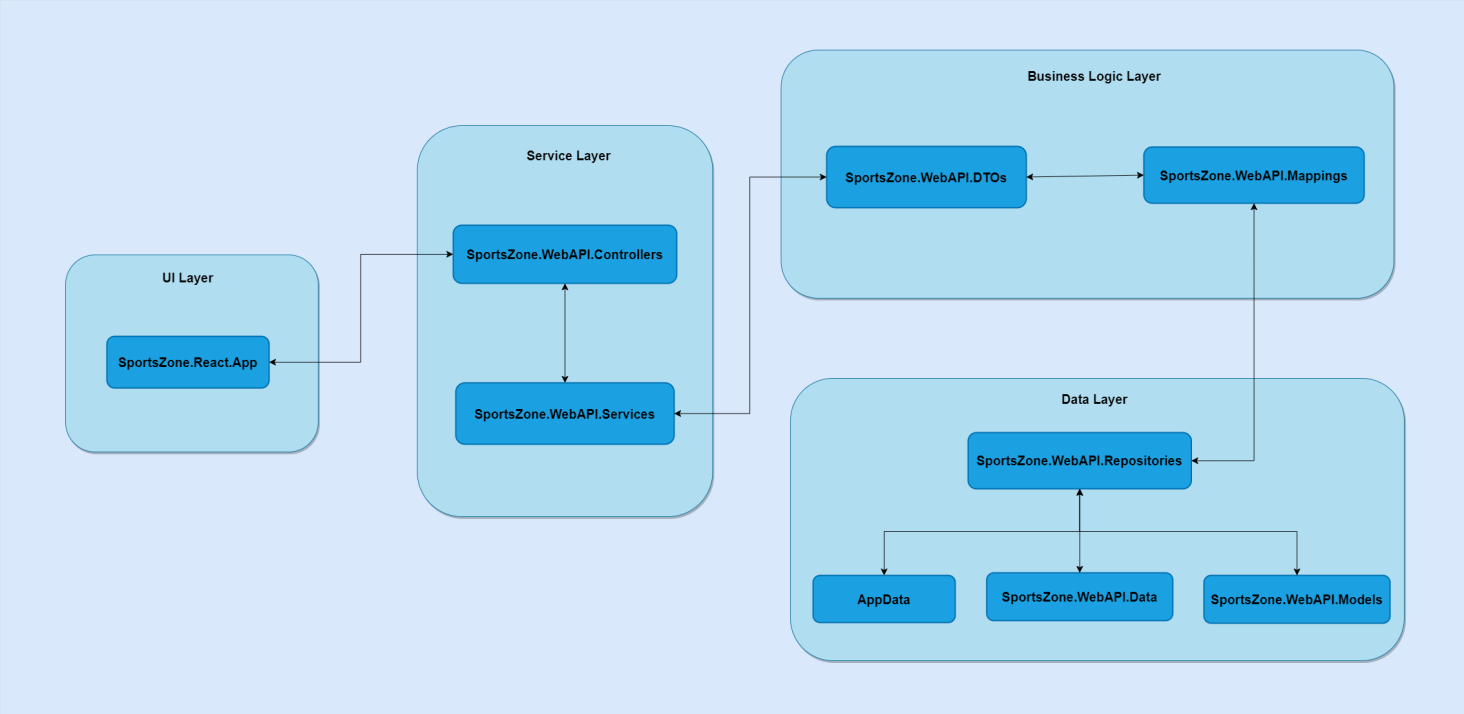
The Data management system connects the Service Layer to Data store.

The cross-cutting layer consist of Security, User management, Metering, Licensing, Workflow, Reports & Exception handling.

The Audit trail and Login activity should be implemented in Database layer.

## 2.3 Component Level Design

This section will provide the details of the various components which will form the ABC in the application architecture.



### 2.3.1 Data Stores

| **No** | **Datastore Name** | **Description** |
| --- | --- | --- |
| 1 | KLMDB | This database captures the details of all transactions used while the patients are applying for financial assistance. |
| 2 | User Management DB | This database captures the details of user creation and administration. |

### 2.3.2 Presentation layer

This layer represents the man machine interface. This layer is responsible for presentation, user interaction, client-side validations, and interaction with the process layer. Responsive web design (RWD) approach has been adapted for the web design aimed at crafting sites to provide an optimal viewing and interaction experience for easy reading and user navigation with a minimum of resizing, panning, and scrolling. This design would also aim to have a provision to extend support to other devices such as tablet.

### 2.3.3 Service Layer

This layer provides access to service, and it contains the implementations of service interaction. The Client will interact with the services of the ABC to populate the data from the CASS API / Core API / database. All data interactions are handled by REST API calls. It contains the necessary base use cases for the service. It includes the Authentication, Authorization, Logging, Exception handling, Security, Handlers, Dependency Injection., etc. It contains the request and response entities of client interface. The request schema from the client interface should be equivalent to corresponding request entity in this layer. The services should handle the request from the client interface and provide the response for it. The service traverse through the Service layer which is used to manage the database operation and XYZ Service. The service authentication must be evaluated in this layer. This enables us to access the services/endpoints in the system.

### 2.3.4 Business Service Layer

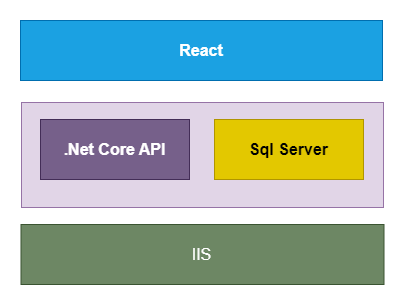
This layer is responsible to implement the business logic and it would take care of external layer communication like Database operation and SportsZone.WebAPI Service. It would map the view entity and business entity for manipulating the data between different layers.

### 2.3.5 Data Layer

This layer has entities that are used to persist the data into database and compute the logics in business service layer. The database connection/context should be established in this layer. Also, it contains the unit of work and field mapping.

# Technical Stack Layout

This section comprises the list of technologies which has been proposed for the ABC. The technical stack consists of UI as Angular v9.0 which is based on a powerful and flexible component model for building rich interactive web UI. Angular UI components using a TypeScript: Angular components can seamlessly handle UI events, bind to user input, and efficiently render UI updates. .NET Core 3.1 is a free and open-source, managed computer software framework for Windows, Linux, and macOS operating systems. It is a cross-platform successor to .NET Framework.

Tech Stack LayoutTech Stack LayoutTech Stack LayoutTech Stack Layout

# Functional Layout

This section describes the interaction between the various functional units of the ABC application. This layout depicts use activity, entity relationship and class diagram of the application. This gives a detailed perspective of each unit in the system.

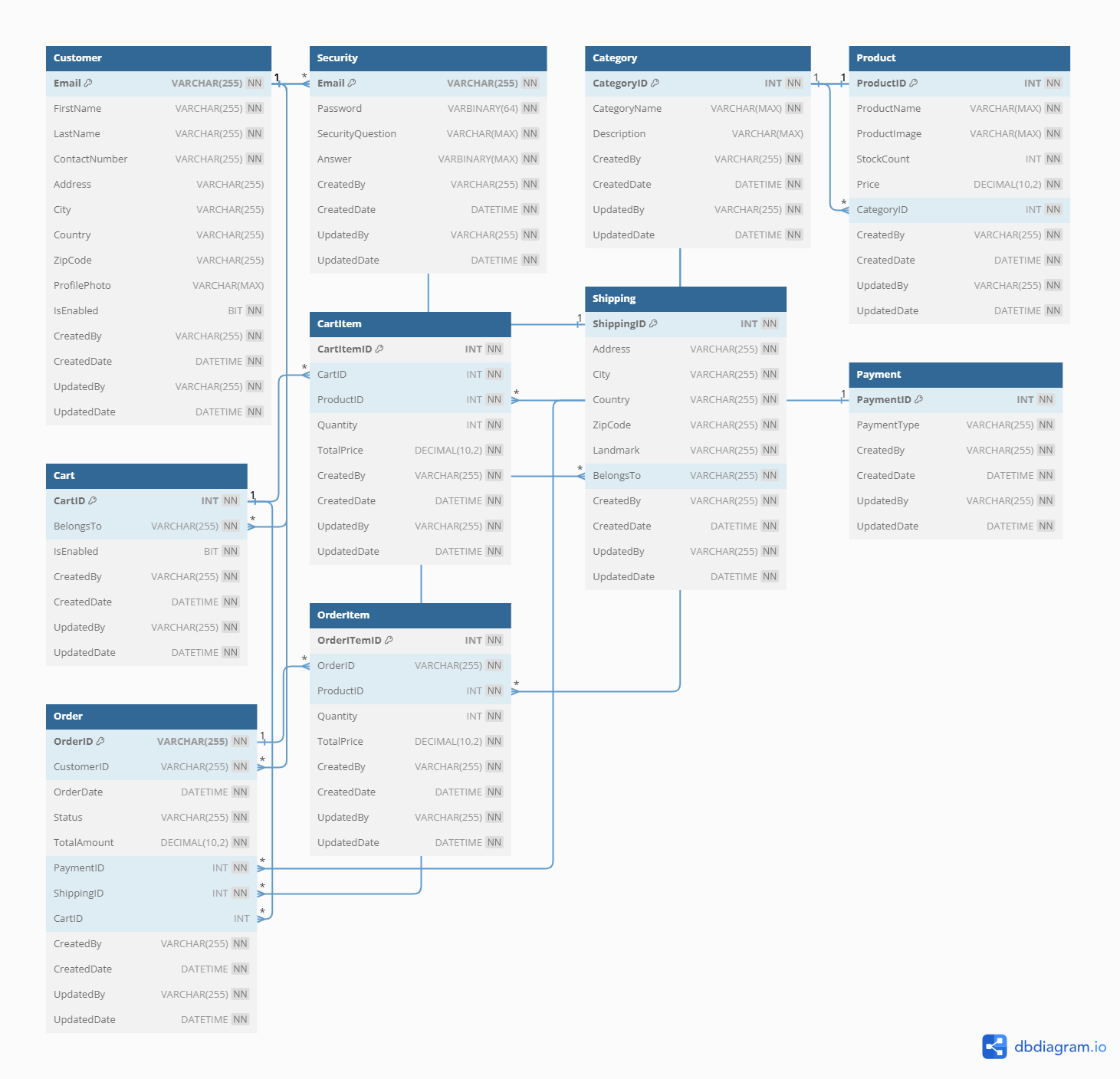
## 4.1. Class Diagram

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

## 4.2. Entity Relationship Diagram



# Non-Functional Requirements

This section will describe the non-functional attributes. These attributes captured here are the possible non-functional requirements identified so far as part of the requirements analysis. This section would be expanded based on the needs of the application.

## Security

The security can be further enhanced through REST service calls over https, SSL, Data encryption and through Data isolation. Based on the requirements, on what levels of security is required for the application and which kind of the client’s data are sensitive in nature, the implementation details could be added to this section. The application framework will be used for authentication and authorization of system.

## Performance

Application performance benchmarks like response time, transaction processing time, throughput, volume data processing/Sec and other Non-Functional requirements will be adhered as per the understanding arrived during the analysis of the requirements.

## Multilingual

Multilingual support will be provided by the application design. Our system will support multiple languages (data field and labels) using the property information for the language. The system maintains translated text in property files. However, the language will be defaulted to English for the application’s initial release.

## Audit Trail

The application’s framework provides a utility for audit trails. The framework provides a provision on the configurable entities that we need to audit. This will enable to track all the transitional data entries for that particular entity.

# Glossary

Section: Glossary

| **Term** | **Definition** |
| --- | --- |
| API | The Application Programming Interface is a service-based interface which contains all the methods used for ABC. |
| QA | Quality Assurance |
| OPS | Development Operations |
| AA | Active ASSIST |
| DB | Database |
| HW | XYZ |