Surety modernization Application Architecture

|  |  |
| --- | --- |
| **Date:** | 5th January 2021 |
| **Version:** | 0.1 |
| **Authors:** | Jaish Mathews |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 1/4/2021 | 0.1 | First Draft | Jaish Mathews (DXC) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Agenda 4](#_Toc60741706)

[2 Architecture Participants 5](#_Toc60741707)

[2.1 UI Tier 5](#_Toc60741708)

[2.2 API Tier 6](#_Toc60741709)

[2.3 Data Tier 6](#_Toc60741710)

# Agenda

Here describing the proposed Application Architecture Components and Layers for Surety Modernization project.

# Architecture Participants

Below diagram defines the Application Architecture participants.

Graphical user interface, application, PowerPoint

Description automatically generated

## UI Tier

This tier will be made with Blazor Web Assembly model. Main participants of this Tier are as follows.

### Blazor Pages

The main pages of the UI are mentioned as Blazor Pages. Blazor Pages contains various UI functionalities and can host multiple Blazor Components. Each of the Blazor Page functionalities needs to be created as separate Blazor Components and needs to be hosted inside Blazor Pages. Below is the Blazor Pages and Blazor Component relation.

**Blazor Page**

Blazor Pages has an extension of .razor and residing inside “Pages” folder.

### Blazor Components

Blazor Components are hosting inside the Blazor Pages. A Blazor Page will be divided in to multiple Blazor Components based on the different functionalities. If a Blazor Page has a single functionality, then there is no need of a separate Blazor Component, but the Blazer Page itself is the Blazor Component.

Blazor Components has an extension of .razor and residing inside “Components” folder.

### DTO Classes

DTO Classes would be used for external data transfer and for external data receive as well. No other objects would be used for data transfer and receive. These are POCO classes with needed Data Annotation attributes for validation. DTO Classes also serves as a layer of separation during communication with actual database objects through APIs. All DTO Classes should be in a separate project and to be shared across both UI and API tiers.

## API Tier

## Data Tier