**LEAN TECHNICAL DOCUMENTATION**

**[SE Egbokhare]**

[37469878]

**Date:** *2024-09-29*

**Document Version:** *V1.0*

Table of contents

[1 Introduction 1](#_Toc171508441)

[2 Solution Design 1](#_Toc171508442)

[2.1 Detailed Solution Design 1](#_Toc171508443)

[2.2 Data Design 1](#_Toc171508444)

[2.3 Technical Assumptions 1](#_Toc171508445)

[2.4 Technical Caveats 1](#_Toc171508446)

[2.5 Wireframes 1](#_Toc171508447)

[3 Errors & Exceptions 2](#_Toc171508448)

[3.1 Business Exceptions 2](#_Toc171508449)

[3.2 Application Errors 2](#_Toc171508450)

[4 Environment Details 2](#_Toc171508451)

# Introduction

This GitHub project aims to develop and track the completion of various tasks related to multiple CMPG 323 projects over the semester. The projects are designed to solve common challenges in the software development lifecycle, such as agile project management, secure web API development, web application deployment, robotic process automation, and data visualization.

**Problem(s) to Solve**:

* Ensure smooth management and tracking of software development projects with clear timelines, roles, and progress metrics.
* Automate repetitive tasks using RPA technologies.
* Develop secure web services and cloud-hosted applications with robust authentication.
* Implement and visualize data analysis for effective decision-making using AI and Power BI.

**Solutions to be Developed**:

* A **Web API** solution for secure data management and processing (Project 2).
* A **Web Application** leveraging modern frameworks for user interactions (Project 3).
* A **Robotic Process Automation (RPA)** solution for automating business processes (Project 4).
* A **Data Visualization** solution for generating insightful reports using Power BI (Project 5).

# Solution Design

## Detailed Solution Design

#### 2.1 Detailed Solution Design

This solution uses a variety of technologies that interact in a cloud-based environment, forming the backbone of the system.

* **GitHub Project**: Used for task management and tracking project progress.
* **.NET Web API (C#)**: A secure API for managing telemetry data, protected with JWT authentication.
* **Blazor Web App**: Implements user interfaces and leverages the repository pattern for data operations (SOLID principles).
* **UiPath RPA**: Automates routine tasks, such as data entry and report generation.
* **Power BI**: Visualizes data insights, allowing stakeholders to make informed decision

Figure 2‑1: Context Diagram

## Data Design

The context diagram is broken down into more detail to show how the different technologies will interact with one another within the developed solution. The transportation of data, across the solution, is detailed in the data flow diagram below.

Figure 2‑2: Data Flow Diagram

* The API communicates with the database, receiving requests from the web app and passing data to the RPA solution.
* Power BI connects to the data repository for visualization and analysis.

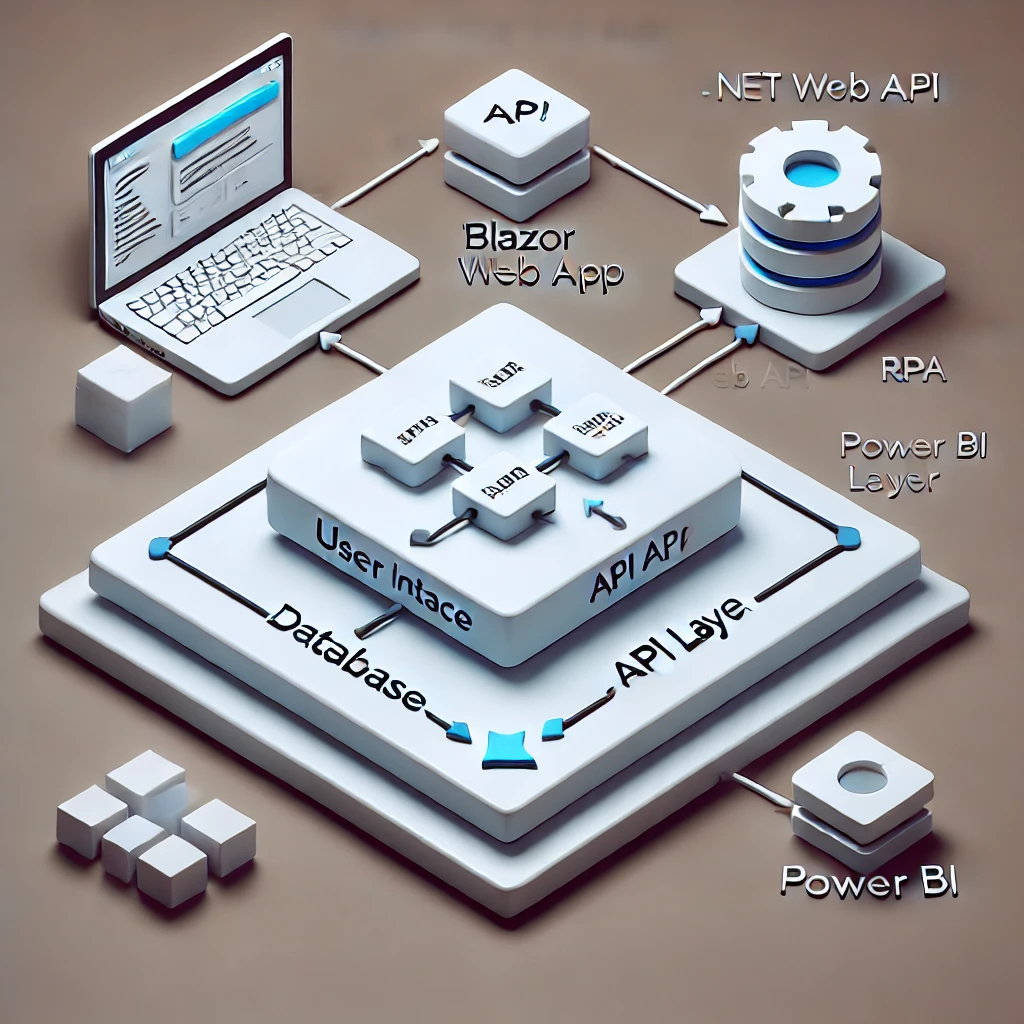


Figure 2‑3: Data Design

## Technical Assumptions

* The following assumptions have been made while designing the solution:
* All license allocations will be done before development commences.
* <List all other assumptions, especially relating to business rules>
* All relevant licenses (e.g., UiPath, Azure hosting) will be provided before development.
* The project team will have access to all necessary development tools and environments.
* The system will handle data up to a certain volume, in line with the expected user base.

## Technical Caveats

The following caveats have been raised as part of the solution design. These caveats would need to be addressed and may have an impact on the design.

* Any changes to cloud service providers (e.g., Azure) may impact deployment timelines.
* Data security and encryption will depend on third-party services.
* RPA integrations may require specialized handling for unexpected changes in web page layouts.

## Wireframes

All prototypes for the reports can be found below:

# Errors & Exceptions

## Business Exceptions

The following business exceptions should be built into the solution:

| Exception Name | Step | Parameters | Action To Be Taken |
| --- | --- | --- | --- |
| InvalidClientData | Data Validation | Client Name, Email | Log the error, notify the user, and reject the entry. |

Table 1: Business Exceptions

## Application Errors

The following application (unknown) errors may occur as part of the solution:

| Exception Name | Step | Parameters | Action To Be Taken |
| --- | --- | --- | --- |
| DatabaseConnectionFail | API Initialization | Database URL | Log the error and retry the connection after a set interval. Notify the user if the issue persists. |

Table 2: Business Exceptions

# Environment Details

The development of the solution would need to be executed as per the designated development strategy. The information below represents the solution and the appropriate environment(s) that will be used to implement the overall solution:

| Item | Description |
| --- | --- |
| Environment Type | Development  Testing  Production |
| Credentials Needed | Admin and developer credentials for development, test accounts for QA, and live credentials for production |
| Development Technologies Used | .NET Core (API), Blazor (Web App), UiPath (RPA), Azure (Cloud Hosting), Power BI (Data Visualization) |
| Deployment Technologies Used | Azure DevOps for continuous integration/continuous deployment (CI/CD), GitHub for version control |
| Scalable | Yes, designed to scale horizontally with cloud hosting |

Table 4‑1: Project Details