**LEAN TECHNICAL DOCUMENTATION**

**DK Khoza]**

32593392

**Date:** *2024-07-29*

**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 3](#_Toc171508444)

[2.3 Technical Assumptions 5](#_Toc171508445)

[2.4 Technical Caveats 5](#_Toc171508446)

[2.5 Wireframes 6](#_Toc171508447)

[3 Errors & Exceptions 6](#_Toc171508448)

[3.1 Business Exceptions 6](#_Toc171508449)

[3.2 Application Errors 7](#_Toc171508450)

[4 Environment Details 7](#_Toc171508451)

# Introduction

Throughout the semester, the NWU Tech Trends project will tackle a series of challenges through the completion of five distinct projects, each building on the foundation of the previous one. NWU Tech Trends, a consulting firm specializing in cutting-edge software development with a focus on Data Integration, Generative AI, and Automation, faces a key issue: how to accurately quantify the benefits they bring to their clients. These benefits often come in the form of time and cost savings, but demonstrating and tracking these results is challenging.

The solution to this problem will be developed incrementally, with each project addressing different aspects of the problem. Projects 1 and 2 will concentrate on the creation of a Web API to capture and store telemetry data associated with clients and their respective projects. This data will serve as the backbone of the solution, providing a comprehensive view of time, cost, and performance metrics. Project 3 will build on this by creating a user-friendly web application that allows for the management of clients, projects, and the associated telemetry data. Moving into Project 4, Robotic Process Automation (RPA) will be implemented to test the functionality of the web application, ensuring it meets both performance and reliability standards. Finally, Project 5 will focus on the development of a reporting tool that visualizes time and cost savings for both clients and projects, giving NWU Tech Trends a clear picture of their impact.

This comprehensive, multi-phase approach will ultimately deliver a robust solution that allows NWU Tech Trends to track and report on the value they provide to their clients, while also streamlining the internal management of client data, project metrics, and automation efforts.

# Solution Design

## Detailed Solution Design

To address NWU Tech Trends' challenge of quantifying benefits like time and cost savings, the solution will integrate various technologies to create a cohesive, scalable system. Each component will contribute to the overall functionality and efficiency of the system.

**1. Web API:**

* **Technology**: ASP.NET Core (or alternatively Node.js).
* **Function**: The Web API will serve as the backbone for data storage and retrieval. It will handle CRUD operations on telemetry data, client information, and project data.
* **Interaction**: The API will communicate with a database (e.g., SQL Server, or NoSQL options like MongoDB) to store the collected telemetry data. It will also interface with the web application for user interactions and the RPA bot for automated testing data.

**2. Web Application:**

* **Technology**: React.js (or other frameworks like Angular or Vue.js).
* **Function**: This component will provide an interface for managing clients, projects, and telemetry data. It will allow users to create, edit, and delete project information and monitor data collected in real time.
* **Interaction**: The web app will consume the Web API, allowing for the seamless management of telemetry data. It serves as the primary interface for users to interact with the system.

**3. Robotic Process Automation (RPA):**

* **Technology**: UiPath (or alternatives like Blue Prism or Automation Anywhere).
* **Function**: The RPA bot will simulate user interactions with the web application, performing end-to-end testing. It will ensure that the system behaves as expected and log any deviations.
* **Interaction**: By interacting directly with the web application, the RPA bot will automate user actions such as data input and validation. The results of these tests will be recorded in the database for reporting purposes.

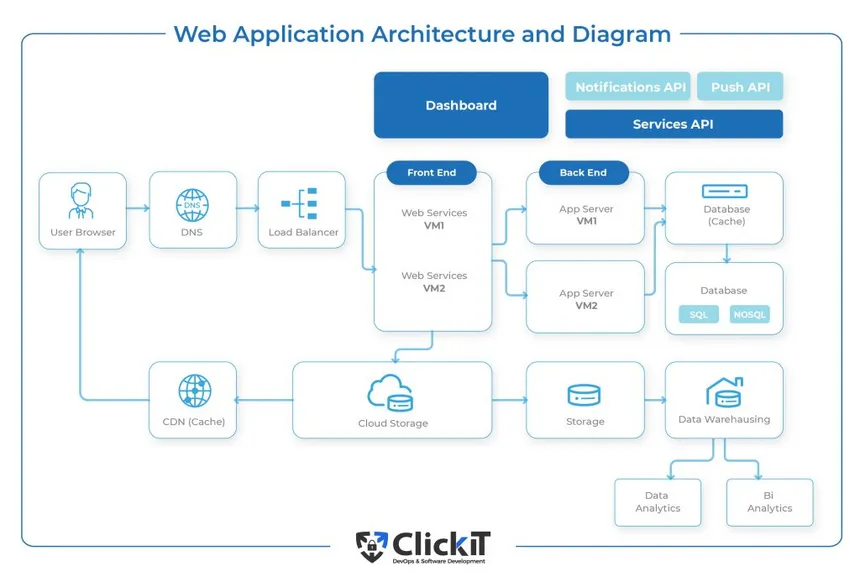
**4. Reporting Component:**

* **Technology**: Power BI (or alternatives like Tableau).
* **Function**: This component will generate visual reports on the time and cost savings achieved for each project and client. These insights will be crucial for demonstrating the value of NWU Tech Trends' services.
* **Interaction**: Power BI will pull data from the Web API, enabling real-time reporting and the generation of dashboards that summarize client benefits.

**5. Cloud Hosting:**

* **Technology**: Microsoft Azure (or AWS/GCP depending on preferences).
* **Function**: The solution will be deployed on the cloud to ensure scalability, security, and accessibility from anywhere.
* **Interaction**: All components (Web API, web application, RPA bot, and reporting tool) will be hosted and managed in the cloud, facilitating seamless integration and deployment.

This multi-layered architecture ensures that each component works efficiently with the others, providing NWU Tech Trends with a scalable, reliable, and insightful solution. Each technology plays a critical role in tracking, testing, and reporting on client outcomes.

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.

2.2.1 Data Flow Diagram Explanation:

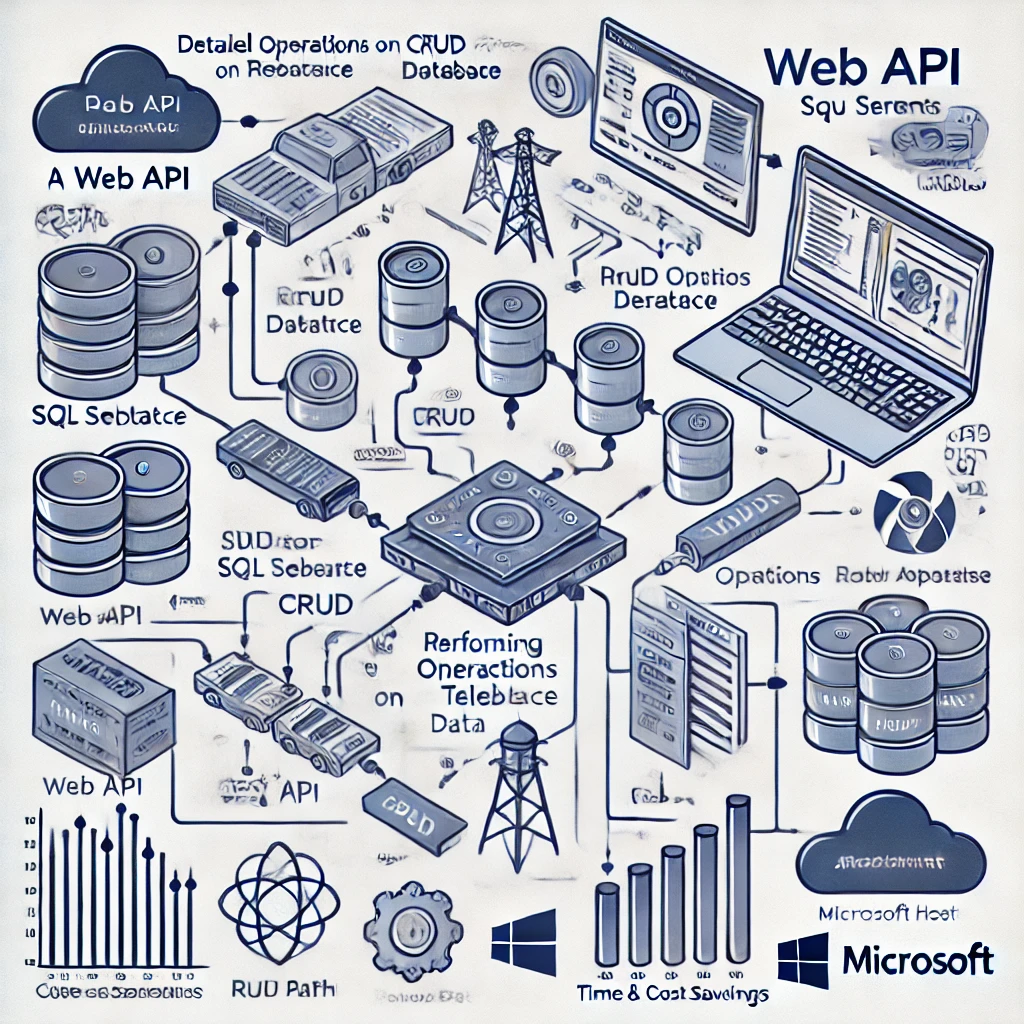
The data flow diagram (DFD) outlines the movement and processing of data throughout the NWU Tech Trends solution. It helps visualize how data is exchanged between the various components of the system, ensuring that each process is defined and the flow of information is clear.

1. **External Users (Clients/Administrators)** interact with the system through the **Web Application**. This web interface captures inputs like project and client information, allowing CRUD operations on telemetry data.
2. The **Web Application** communicates with the **Web API** via HTTP requests. It sends the user input (client data, project data, telemetry data) for processing, which is then relayed to the **Database**.
3. The **Web API** (ASP.NET Core) functions as a middle layer, receiving requests from the web application. It manages data handling operations by interacting with the **SQL Server database**. The API executes CRUD operations, storing telemetry data, and fetching project or client data as needed.
4. The **SQL Server Database** stores structured data, such as client information, project details, telemetry records, and any associated metrics (time and cost savings).
5. The **RPA Bot (UiPath)** interacts with the **Web Application** for automated User Acceptance Testing (UAT). The bot mimics user behaviour, testing the application's features, and records results back to the system.
6. Finally, the **Reporting Component (Power BI)** pulls data from the **Database** (via the Web API) to generate reports. These reports visualize time and cost savings, providing actionable insights on client projects and automation benefits.

**2.2.2 Data Design:**

* **Clients Table**: Stores client-specific information like client name, industry, and contact details.
* **Projects Table**: Contains project-specific data such as project name, start and end dates, and associated clients.
* **Telemetry Data Table**: Records metrics related to automation, including time saved, cost savings, and the specific project/client it relates to.

. Figure 2‑1: Data Flow Diagram

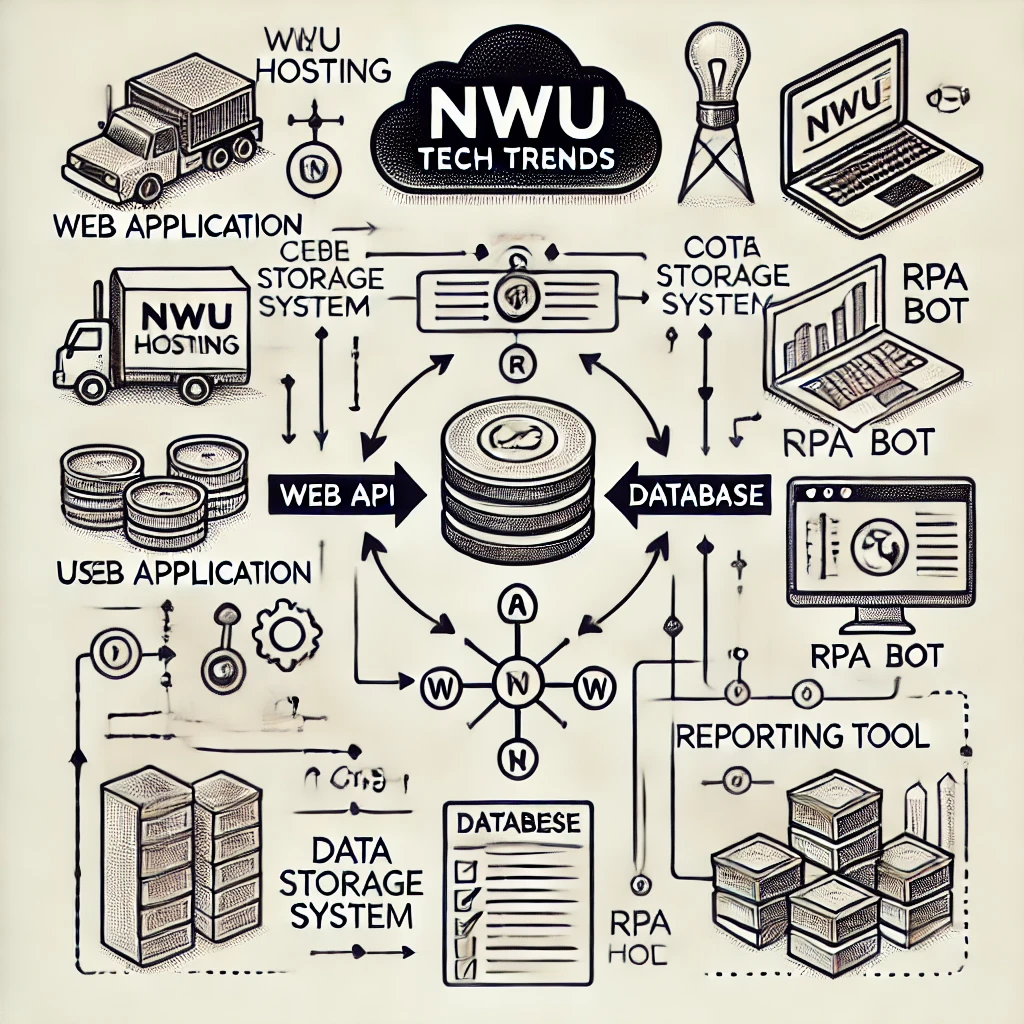


## Technical Assumptions

The following assumptions have been made while designing the solution:

* **License Allocations**: All necessary software licenses (for cloud hosting, APIs, databases, RPA, and reporting tools) will be allocated and configured before development begins to avoid interruptions.
* **Data Integrity**: It is assumed that the data provided by the clients, such as project details and telemetry data, will be accurate and consistent. There will be no significant data validation issues or discrepancies that could affect system functionality.
* **Stable Internet Connectivity**: The cloud-hosted environment assumes continuous and stable internet connectivity for users accessing the web application, APIs, and RPA systems, as well as for the automation processes and reporting components.
* **User Roles and Permissions**: Pre-defined user roles (e.g., administrator, client) will exist, and access control to different parts of the system will be managed accordingly without the need for role customization.
* **Standard Business Rules**: The business processes related to tracking time and cost savings through automation will follow standard practices and predefined metrics. No unique, client-specific business rules will necessitate major deviations from the core system design.
* **Technology Compatibility**: The chosen technologies (ASP.NET Core, SQL Server, UiPath, Power BI, etc.) are fully compatible and integrate seamlessly without significant configuration challenges.
* **Client System Requirements**: Client environments are expected to have the necessary infrastructure and software to support interaction with the web application and automation tools, particularly for remote access and API interactions.

Figure 2‑2: Data Design



## Technical Caveats

The following caveats have been identified during the solution design phase. These potential limitations may need to be addressed and could impact the overall implementation:

* **API Rate Limits**: The Web API may face restrictions related to API rate limits when scaling, particularly if external services are integrated. This could slow down the performance of the system if not optimized.
* **Data Latency**: There could be delays in data retrieval from cloud-hosted databases, especially if the solution handles large volumes of telemetry data. Optimizing database queries and using caching mechanisms may be necessary to address this.
* **Automation Complexity**: The RPA solution might encounter challenges with unexpected changes in the web application's UI or workflow. Frequent updates to the web application could require constant RPA script maintenance, leading to potential delays in automation testing.
* **Cross-Platform Compatibility**: Ensuring that the web application works seamlessly across different browsers and devices (mobile, desktop) could require additional testing and front-end adjustments, which may extend development timelines.
* **Cloud Hosting Costs**: While cloud hosting offers scalability, the costs associated with hosting and scaling the solution in environments like Microsoft Azure may exceed initial estimates, especially with high traffic or complex reporting requirements.
* **Data Privacy and Security**: Managing sensitive client data (telemetry, project details) in a cloud environment raises concerns regarding data security and compliance with regulations like GDPR. Ensuring the right encryption, access control, and security protocols are in place may increase complexity.
* **Third-Party Integration Reliability**: Integration with third-party services such as Power BI for reporting or UiPath for RPA may introduce dependencies that could affect the system's reliability if those services experience downtime or changes in their APIs.
* **Performance Under Load**: The system’s performance could degrade under high load conditions, especially when multiple clients are simultaneously running RPA bots or generating large reports. Load balancing or additional server resources may be required to maintain performance.

## 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 |
| --- | --- | --- | --- |
|  |  |  |  |

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 |
| --- | --- | --- | --- |
|  |  |  |  |

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 | GitHub, API access |
| Development Technologies Used | GitHub, VSCode, Azure(optional) |
| Deployment Technologies Used | GitHub, VSCode, Azure(optional) |
| Scalable | Yes |

Table 4‑1: Project Details