Process Design Document (PDD)

DHL Leave Management System

DIGITAL AUTOMATION CHALLENGE 2.0

DHL APSSC

## Zuhayer Adnan Siddique

## Bachelor of Computer Science (Software Engineering)

## Universiti Teknologi Malaysia (UTM)



# Table of Contents

[**1. Introduction 3**](#_6qy8cn5qfzpo)

[1.1 Purpose 3](#_9wdt51nt0a2x)

[1.2 Scope 3](#_abgwoqh8vmy0)

[1.3 References 3](#_kvmc54sujzmk)

[**2. How to execute the web solution & script 4**](#_83rzop82flbm)

[**Prerequisites 4**](#_y9oab55k5bnk)

[**3. System Overview 5**](#_xoce82z6kvoq)

[1. Web Application (ReactJS + NodeJS) 5](#_apcdm0im3o38)

[2. Automation (UiPath Studio) 5](#_a2lusddo9zoa)

[**4. Key Features 6**](#_1j5fic8xpnts)

[**5. Automation Solution 8**](#_58qppc8cyjnj)

[5.1. Workflow Design Strategy 9](#_1ntfw2ocrjff)

[5.2. Detailed Workflow 9](#_vh9gbe84q4gv)

[**6. Web Solution 12**](#_nq4od751wgb7)

[6.1 High Level Overview of Main Components 12](#_33k9o8f3alej)

[**7. Testing 13**](#_hwzg156xj8k7)

[**8. Possible Future Improvements 14**](#_brfrevcrs0f5)

[**9. Summary 14**](#_vp53fl4r9bml)

# 1. Introduction

## 1.1 Purpose

The Process Design Document (PDD) serves as an overview for the RPA-Powered Leave Management System for DHL APSSC’s Digital Automation Challenge 2.0. This document is intended for Project Managers, Software Engineers, Testers, and anyone else who will be

involved in the implementation of the system.

## 1.2 Scope

1. **Web Solution**: Built with **ReactJS** and **NodeJS**, this component provides the frontend and backend infrastructure for the leave management web application, enabling HR to input and manage leave applications manually.
2. **Automation Solution**: Developed using **UiPath Studio**, this component automates the data entry process by validating leave records from an Excel file and inputting them into the web system.  
   It follows a **Transactional Business Process with Row-Level Fault Tolerance** inspired by REFramework, which ensures:

* Each row is treated as an independent transaction
* Automatic retry is performed for transient input or validation failures
* Failed rows are safely logged without interrupting the rest of the process
* Browser-level crashes are caught and reported as partial or failed executions
* A summary report is generated via email and API log submission at the end of the run, with a log of failed rows and the reason for failure are embedded in the email

Each of the components will have a high level overview of its critical elements in this document below.

## 1.3 References

* <https://www.dhl.com/global-en/delivered/innovation/logistics-automation.html>
* <https://www.uipath.com/resources/automation-case-studies/dhl-global-forwarding-freight>
* [https://briohr.com](https://briohr.com/leave-and-time-off/?utm_term=leave%20management%20system&utm_campaign=(MY)+Generic&utm_source=adwords&utm_medium=ppc&hsa_acc=1726758884&hsa_cam=16065629989&hsa_grp=138542166651&hsa_ad=676918495907&hsa_src=g&hsa_tgt=kwd-96800742&hsa_kw=leave%20management%20system&hsa_mt=b&hsa_net=adwords&hsa_ver=3&gad_source=1&gad_campaignid=16065629989&gbraid=0AAAAACnfZELnGwhOk8TxAUV9gFYo7wJSL&gclid=Cj0KCQjw8vvABhCcARIsAOCfwwoE8cDfEiWMhtzzGEAEv7fvyPDGmeftWbqf_LZlkzBIifofn6IR8OkaAvmoEALw_wcB)
* <https://www.youtube.com/watch?v=b-4228rB0vI>
* <https://docs.uipath.com/task-capture/standalone/2022.4/user-guide/details-about-the-pdd>

# 2. How to execute the web solution & script

# **Prerequisites**

* Ensure the following software is installed on your system:
  + **Node.js**
  + **UiPath Studio (and any 3rd party libraries UiPath requires)**
  + **Google Chrome browser**
  + **UiPath Browser Automation extension for Chrome**

**Running the Web Solution**

1. Open a terminal window.
2. Navigate to the **project root directory**.

* Run the following command to install all frontend and backend dependencies:  
  npm install
* Once installation is complete, start the application with:  
  npm run dev

1. This will start both the **ReactJS frontend** and the **NodeJS backend** concurrently
2. Visit <http://localhost:8080/> to access the web application.
3. The backend is hosted at [http://localhost:5000/](http://localhost:8080/)

**In case of Step 2 failing, try the following:**

* Open terminal as administrator
* Navigate to to lms-frontend and lms-backend and run npm install on both folders
* Once installation is complete, start the application with: npm run dev

**Launching the Automation Script**

There are two options to execute the automation workflow:

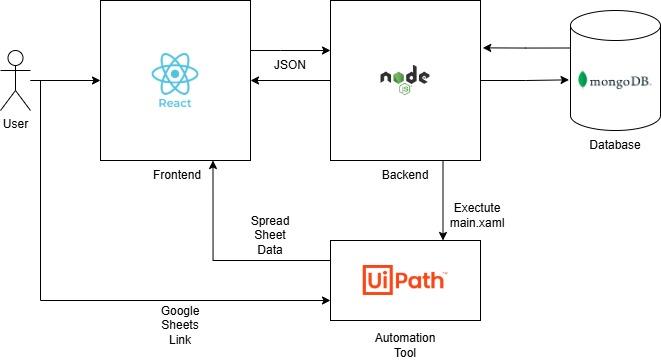
Option 1: Manual Execution

* Navigate to the LeaveAutomation folder inside the project directory.
* Open Main.xaml using **UiPath Studio** and clicking **Run**

Option 2: Via Web Application

* Open the web application in your browser.
* Log in using your credentials (provided in the login page).
* On the **Dashboard**, click the **Floating Action Button (FAB)** to Open Main.xaml using **UiPath Studio**.

# 3. System Overview



*Figure 2.1 - High-level system architecture*

#### **1. Web Application (ReactJS + NodeJS)**

* Acts as the user interface for the Leave Management System.
* Users can launch the automation via a floating action button.
* Backend APIs trigger the bot and handle logs and results.

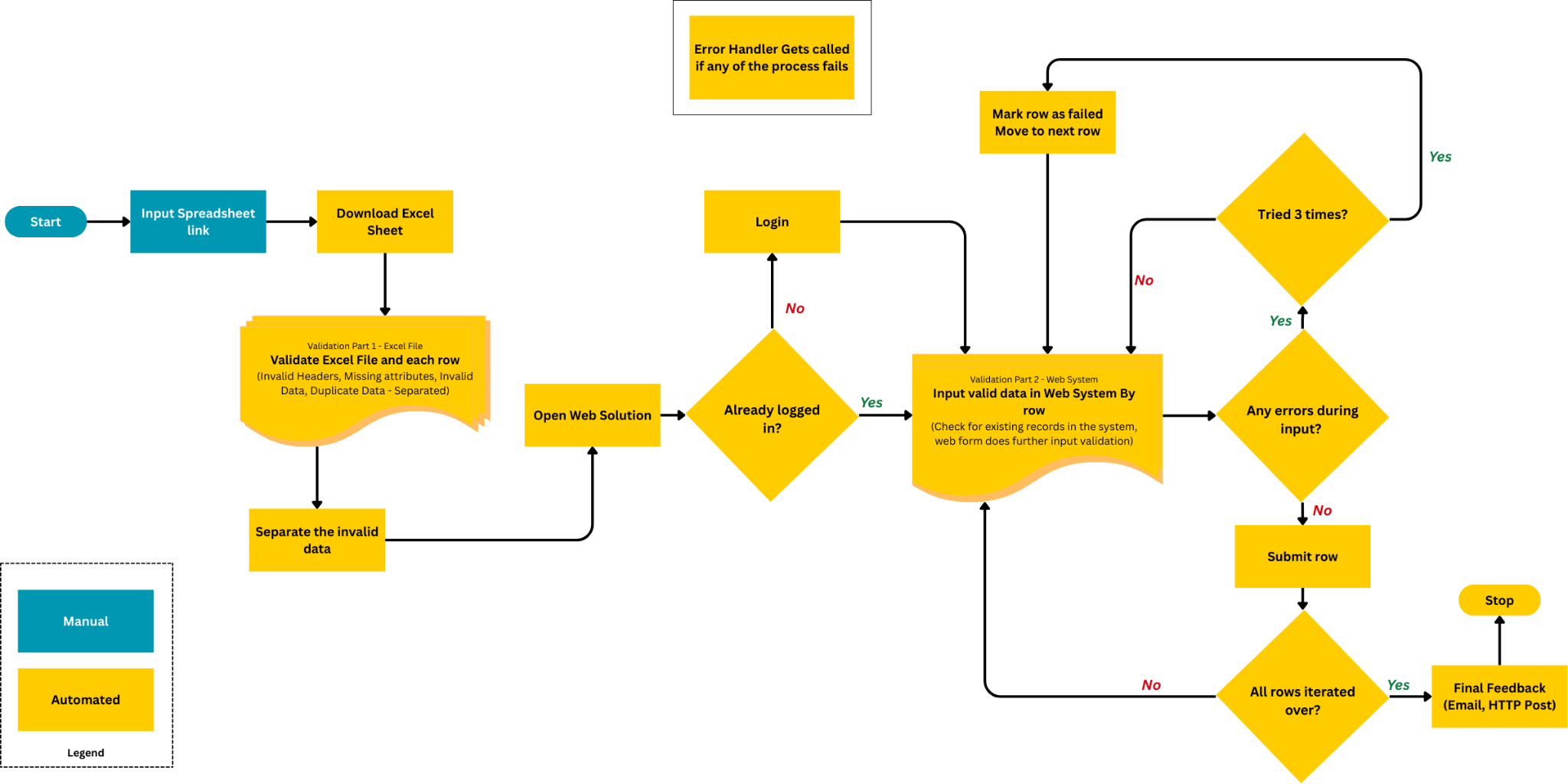
#### **2. Automation (UiPath Studio)**

* Runs from the web app or file explorer.
* Automates the process of:
  + Logging in (if not already logged in)
  + Downloading the Spreadsheet
  + Validating data and Submitting valid entries
  + Custom retry and error handling logic in case of any failure during input in the web system
  + Sending a results/errors summary email

# 4. Key Features

| *Figure 4.1 Validation Check - 1* | 1. UiPath processes the excel data to find invalid data  - Invalid Headers (will terminate process)  - Duplicate Rows  - Missing attributes  - Invalid Type of attributes  - Dates are invalid  All within the **excel file**  - Creates a separate DT to store all the rows that have been removed for logging purposes. |
| --- | --- |
| *Figure 4.2 Validation Check - 2* | 2. Web App  - Validates data based on **Employee ID** and **Leave Dates** both in frontend and backend.  - Proceeds if valid and non-duplicate  - Skips and clears the form if duplicate or invalid. |
| *Figure 4.3 Retry Mechanism* | 3. UiPath will attempt to input each row up to **three times** in case of a failure to submit or clear the form.  Such errors may occur due to backend latency—for example, when the previous request is still being processed and the new data has already finished being entered, resulting in the submit button remaining disabled. |
| *Figure 4.4 Tabular Data with Search and Filter* | 4. Allows HR to efficiently search and filtering leave records using parameters such as Employee ID, Name, and custom date ranges.  Additionally, it clearly separates and categorizes completed leave applications—those with end dates earlier than the current date—ensuring easier tracking and historical referencing. |
| *Figure 4.5 Input Tracking* | 5. UiPath sets a browser **localStorage** flag **(isAutomated = true)** when submitting data.  This helps distinguish between **user-submitted** and **automation-submitted** entries inside the web system. |
| *Figure 4.5 Email Feedback* | 6. Automated email notifications are sent to the designated recipient (**sivanesan.letchumanan@dhl.com**) based on the outcome of the process—success, failure, or partial completion. This is handled via **FinalFeedbackProcess.xaml** or **ErrorHandler.xaml**, and includes:   * Count of successful and failed uploads * Detailed breakdown of failed records (row-level) |
| *Figure 4.6 Automation Logs* | 7. Log of whether the execution of an automation script was **complete**, **partially complete** or completely **failed**. |
| *Figure 4.7 Data Visualization* | 8. Features that present data insights through graphs, charts, tables, sorting, and grouping - helping users understand trends, statuses, and key metrics. |

# 5. Automation Solution



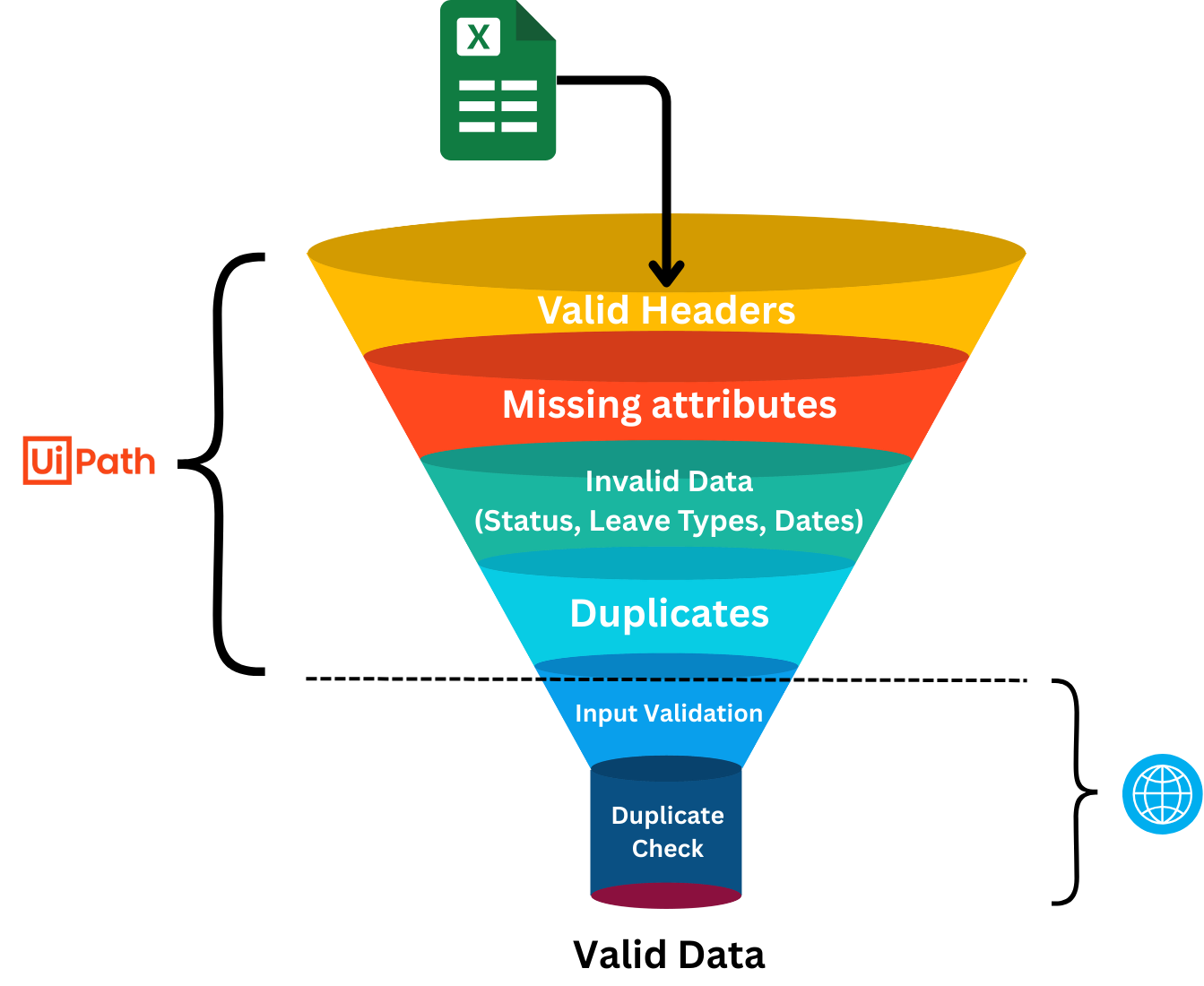
*Figure 5.1 - UiPath Application Flow*

*Detailed flow Draw.io link:* [*https://drive.google.com/file/d/1ClgoDdoAncbwPI55lczY1Z57x72AjMKn/view?usp=sharing*](https://drive.google.com/file/d/1ClgoDdoAncbwPI55lczY1Z57x72AjMKn/view?usp=sharing)

## 5.1. Workflow Design Strategy

A custom workflow architecture was chosen for the Automation Solution. More details can be found in the section below. The process implements **two levels of data validation**:

* **Pre-validation** within the Excel file to check for structural and logical correctness.
* **Runtime validation** during web form submission to detect duplicates, overlaps, and invalid entries in the system and also in the backend.



*Figure 5.2 - Data Validation Flow*

The design further drew inspiration from **REFramework** principles such as modularization, retry logic, and centralized error handling, adapted into a simpler format to implement a **Transactional Business Process with Row-Level Fault Tolerance**:

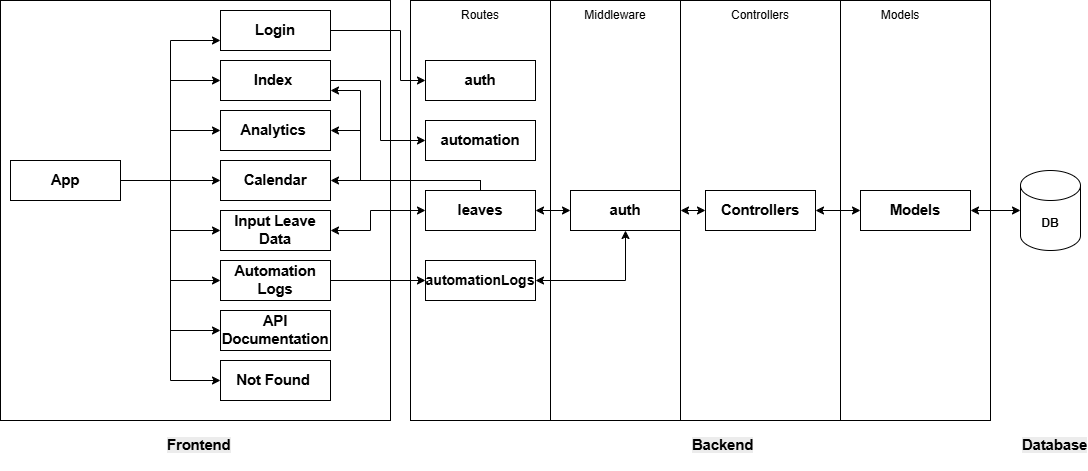
* **Row-by-Row Processing:** Each Excel row is handled as an independent transaction for clarity and traceability.
* **Retry Mechanism:** Submissions are retried up to three times to manage transient issues.
* **Custom Fault Tolerance Logic:** The workflow is designed so that a failure in processing one row does not interrupt the automation flow; errors are logged, and the process continues with the next row.
* **Custom Error Handling:** Exceptions trigger screenshot capture and detailed error reporting.

## 

## 5.2. Detailed Workflow

1. **Main.xaml**: Entry point of the Automation application.
   1. **GetUserInput.xaml** **-** 
      1. Prompts the user to get **input** of the spreadsheet link to be processed.
      2. Checks whether the input is empty using an **if** statement.
      3. Checks whether the input is a valid spreadsheet using an **if** statement.
      4. Extract a downloadable link of the spreadsheet using **assign** statements .and **regex**
   2. **DownloadExcel.xaml -** 
      1. **HTTP client** calls **GET** method to the downloadable link that was created.
      2. Get file path of the downloaded spreadsheet using an **assign** statement.
   3. **ValidateExcelFile.xaml** 
      1. **Get Workbook Sheet** gets the first page of the spreadsheet
      2. **Read Range Workbook** iterates through the spreadsheet and stores it to a datatable
      3. Checks whether the files has missing headers using an **if** statement
      4. Initialize **InvalidRowsDT** and **ValidRowsDT** datatables
      5. **Data Validation Step-1** (Within Excel File)
         1. **For Each Loop** through ExcelLeaveRecords
            1. **If** data has any missing attributes **Add Data Row** to **InvalidRowsDT**.
            2. **If** data has invalid data (date format, End Date < Start Date, Status types, Leave Types) **Add Data Row** to **InvalidRowsDT**.
            3. **Else Add Data Row** to **ValidRowsDT**
         2. **For Each Loop** again but through **ValidRowsDT**.
            1. Use **Staff ID, Start Date, End Date** to generate a unique key for that row
            2. Check the **ValidRowsDT** whether the same data exists (check for duplicate data)
            3. **If** duplicate is found, add it to **DuplicateRows. Else** add it to UniqueRows.
      6. **Merge DuplicateRows** and **InvalidRowsDT** for email report.
      7. **Delete** our downloaded excel file (Clean Up)
   4. **Login.xaml**
      1. **Open Browser**
      2. **Check App State** to verify login state by detecting the presence of authenticated UI elements
      3. If **Target Appears** - Skip Login.
      4. If **Target Does not Appear**
         1. **Type Into** activity inputs credentials into the Login Form and Submits
      5. **Click** Input Leave Data button
   5. **InputExcelData.xaml**
      1. **For Each Loop** through ValidRowsDT that was passed by **ValidateExcelFile.xaml**
      2. **Type Into** to Input Leave Data form
      3. **Inject JS Script** to set isAutomated to true (to indicate that value was submitted by the automation workflow)
      4. **Data Validation Step-2** (Within the Web System)
         1. **System** checks for duplicate data based on **Staff ID, Start Date, End Date**
         2. **Check App State** to check feedback from the Web System for “No Duplicate Data Found” Message
         3. If **Target Appears - Click “**Submit Button”
         4. If **Target Does not Appear -** 
            1. **Click “**Clear Fields Button”
            2. **Add Data** to **InvalidRowsDT** that was created earlier
         5. System will try **3 times** to input this row and will move to the next one if it fails to submit.
   6. **FinalFeedbackProcess.xaml**
      1. **Send SMTP Email**
         1. Successful Data Entries Count
         2. Failed Data Entries Count
         3. Failed Data Entries Rows
         4. Reason for failure
         5. Start and End Time of process
      2. **HTTP Request - POST** to /api/automation-logs
2. **ErrorHandler.xaml:** Runs in case any try-catch blocks fail, and grabs screenshots and sends an email and posts the log in our system.

# 6. Web Solution



*Figure 6.1 - Web Solution High Level Overview*

The system uses a modern web application architecture, specifically a **client-server (multi-tier)** model with a React frontend and a Node.js/Express backend, communicating via RESTful APIs.

## 6.1 High Level Overview of Main Components

* **Index** - Main landing page showing overview of leave management system with a Leave Table component to search, filter and group data
* **Analytics** - Data visualization and statistics about leave patterns and usage
* **Input Leave Data** - Form for submitting new leave requests and checking for duplicates within the system
* **Calendar** - Visual calendar view of scheduled leaves
* **Automation Logs** - History and status of automated leave processing tasks
* **API Documentation** - Reference guide for the system's API endpoints
* **Login** - Authentication page for user access
* **Not Found** - Error page for invalid routes

# 7. Testing

Extensive testing was conducted to ensure the system meets the challenge’s functional and performance requirements across both the web application and UiPath automation components:

**Web Application**

* **Leave Data Input**:  
  Conducted both manual and automated submissions using valid and invalid data, including missing fields, incorrect date formats, duplicate entries, and overlapping leave periods. Confirmed system responses such as “No Duplicate Data Found” were displayed as expected.
* **Login Functionality**:  
  Verified system behavior using correct and incorrect credentials. Confirmed that the Login.xaml automation executed successfully upon valid input.
* **Search & Filter**:  
  Tested the ability to query leave records by Employee ID and date range. Validated the accuracy and completeness of returned data.
* **Performance & Scalability**:  
  Uploaded 100+ leave records to assess system performance under moderate load. Observed stable behavior with consistent response times.

**UiPath Automation**

* **Validation Logic**:  
  Tested various Excel files containing duplicate entries, overlapping dates, missing headers, and logically invalid data (e.g., end date preceding start date). The automation successfully identified and rejected faulty records.
* **Automated Submission**:  
  Validated the form submission process through automation, ensuring proper handling of retries (up to 3 attempts) and response parsing.
* **Email Reporting**:  
  Confirmed the generation and delivery of summary emails for successful and failed uploads.
* **Exception Handling**:  
  Simulated network disruptions and invalid element selectors to confirm error logging mechanisms. Verified that screenshots were captured and error notification emails were generated appropriately.

**Cross-Device Testing**

* Environment Compatibility:
  + Automation was tested on multiple systems, including fresh Windows installations via VMware.
  + Issues such as browser dependency, password activity compatibility, and blocked SMTP ports were diagnosed.
  + These findings emphasized that certain UiPath activities are environment-specific and may require adjustments on new devices and can be resolved by hosting the system on a server inside a container and using unattended robots.

# 8. Possible Future Improvements

* **Deploy with Orchestrator & Host Web App**

Move the automation to UiPath Orchestrator and deploy the web application on a server to eliminate dependency on a local/host machine to create a totally unattended version of the system.

* + This enables centralized management, scheduling, and monitoring of the automation process.
  + It also opens doors for parallel processing during the input of data. Which would result in faster completion times.
* **Implement a Dedicated Employee Table**Introduce a structured employee database to store staff profiles:
  + Link leave records to employee IDs for faster autofill during application input.
  + Enable lookup of historical leave applications for each employee to support smarter approval/rejection decisions by HR or supervisors.
  + Enhance reporting and validation accuracy by referencing persistent employee data.
* **Introduce Role-Based Access Control (RBAC)**
  + Assign roles (e.g., Admin, HR, Viewer) to manage permissions across the system. Employees can view the status of their leave applications.
  + Prevent unauthorized data access or accidental modifications.

# 9. Summary

The DHL Leave Management Automation Solution was designed to be lightweight, modular, and reliable, effectively handling validation, data entry, and exception management. While built on a custom workflow structure, it incorporates key REFramework-inspired practices such as row-level transaction processing, retry logic, and centralized error handling. The solution ensures accurate submissions, clear reporting, and easy maintainability—laying a strong foundation for future scalability and integration with enterprise-grade automation platforms like UiPath Orchestrator.