Introduction

Indus Valley Partners is **a technology based consulting firm** focused on the **capital market domain** since 2000. With its three main segments of business- IVP Products, IVP Managed Services and IVP Consultancy, the company helps its clients stay a step ahead in the capital market domain.

The **leadership team** are as follows:

- **Gurvinder Singh** - CEO & Managing Director (https://www.linkedin.com/in/ivpgsingh/)
  Expertise: New Fund Launches, Hedge Fund Trading Strategies, Fund Administration

- **Gaurav Aggarwal** - CCO
  Expertise: Portfolio Accounting, Performance Attribution, Regulatory Reporting

- Deepak Sawardekar - Managing Director
  Expertise: Technology Architecture, Data Warehousing

- **Nikhil Tyagi** - Managing Director
  Expertise: Private Equity, Order/Execution Management

- **Sandeep Malhotra** - Managing Director
  Expertise: Cloud Data Warehousing, MDM

- Piyush Singhi - Managing Director
  Expertise: Credit, Private Funds

- **Anuj Gandhi** - Managing Director
  Expertise: Hedge Fund Consulting, Pricing and Valuation

- Kevin Ronayne - Managing Director
  Expertise: Sales

Accomplishments

4 $TN+ **AUM managed** using IVP Technology	**1.5 $TN+** **AUM reported** through Regulatory Reporting Platform	**425+** **Strategy & Bespoke** Projects delivered
160+ **Fund managers** using IVP Solutions	**90+** **Fund Managers** using IVP "Digital First" Managed Services	**35%** **Global Hedge fund AUM** running through IVP products

Clients

Company's products and services are divided into two main segments. Front Office and Middle/Back Office

Front Office Products

- Enterprise Data Management
- Security and Reference Master
- Data Warehouse
- Portfolio Solution
- Decision Science
- Order Management System

Middle/Back Office Products

- Reconciliation Solution
- Regulatory Reporting
- Treasury Management
- Pricing and Valuation Automation Solution
- Cash Management Solution
- Expense Allocation Solution
- NAV Solution
- ALT Data Analyzer
- ESG Management

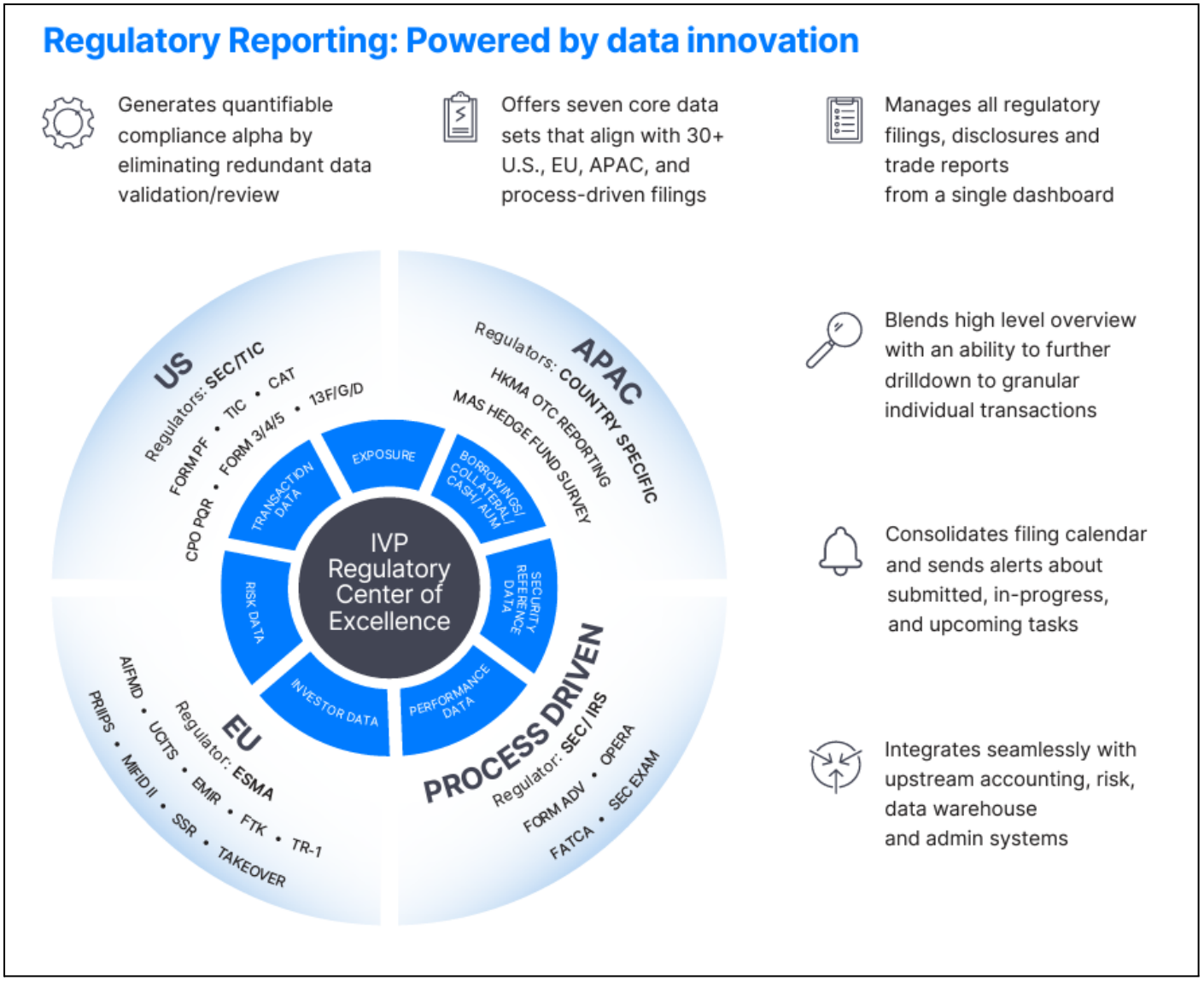
For more details about these products, please visit company website: (https://www.ivp.in/products/)

Awards and Achievements

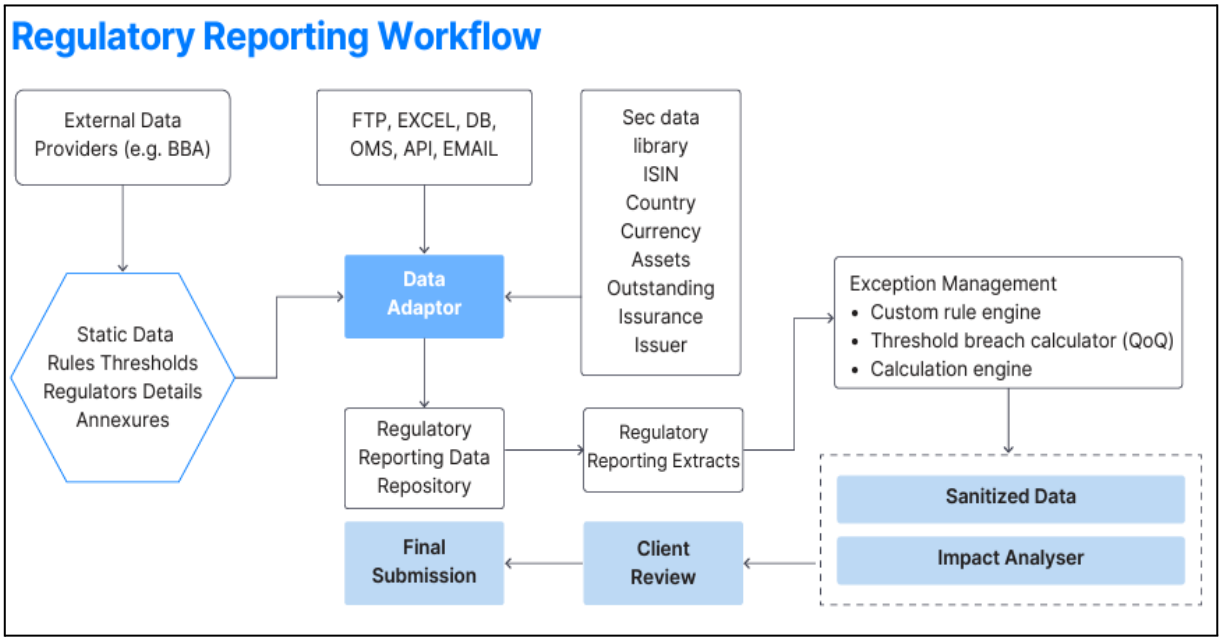
- Indus Valley Partners has been shortlisted in this year's FTF News Technology Innovation Awards 2023 for Best Middle to Back Office Integration Solution and Best Operational Risk Management Solution (Visit Link to know more)
- Indus Valley Partners has been shortlisted in this year's A-Team TradingTech USA Awards 2023 for Buy-Side OMS(Visit Link to know more)
- Indus Valley Partners Wins 'Data Management Insight Awards Europe 2022' for Best Buy-Side Managed Services Platform.(Visit Link to know more)
- Indus Valley Partners has been shortlisted in the 2022 With Intelligence HFM US Services Awards for Best Treasury Management Solution and Best Data Management Solution.(Visit Link to know more)

About the Assigned Product "RAPTOR"

"Raptor" is the core product for Regulatory Reporting Solutions provided by Indus Valley Partners. Regulatory Reporting brings automation and data expertise to the regulatory technology ecosystem, offering seven core data sets that can drive more than 30 global filings with a single, streamlined workflow. Together with the company's digital-first managed services, Raptor uses a cloud-based deployment model to bring enormous economies of scale to compliance reporting. The summary highlights for the regulatory reporting solutions is shown below:



Regulatory Reporting includes comprehensive support for global regulatory filings, removing the need for duplicate effort for these documents and associated investor reporting. Regulatory Reporting includes an easy, high-level visualization of all filings with customized dashboards that allow users to drill down to the most granular details to simplify the process.



Above workflow is maintained and executed by ‘Raptor’ as a core product along with the other IVP products like- Polaris: Data Warehouse Management Tool & Security Master: Security and Reference Management Tool, team of Operations Analysts, Managed Services, Business Analysts and Dev at Support team.

About the roles and responsibilities

Assigned the role as **Associate Software Engineer**(working as a Full Stack Web Development Engineer) and during this six month internship period, assigned the responsibility to complete certain tasks-

- R&D of the Question UI Component to optimize its performance and tackle down its rendering within a second (Component rendering in about 4-5 minutes due to the internal execution in exponential times).
- Develop a Question Preview UI Screen as per the existing application(built in ASP.NET, C# and SQL).
- Develop a Question Review Page as per the existing application(built in ASP.NET, C# and SQL).
- Develop the Question Footer Utilities for the Question UI Component as per the existing application.
- Collaborate with the UI/UX Engineers and fix the product’s UI/UX component in React application.
- Generate the Consolidated Scripts for the client requirements of the Annex4 Form’s FAQs and Glossary.

About the technology used

- **React (Javascript + Typescript)**
  Raptor is building on the solid foundation of React Library. Raptor uses the Redux implemented with Redux Saga Library to manage the application side effects i.e., asynchronous data fetching from APIs, easier to handle. It also integrates the Typescript to efficiently manage the states in React. Unit Testing in Raptor is done with the Jest Framework of Javascript.

- **C# based Web Service**
  C# is used throughout the project to develop the WebService to extract the Procedures created in the SQL Server Management Studio. C# enables one to create a secure gateway for accessing the data in an abstracted way. There are several advantages of using Web Service - independent of language, protocols, platforms, follows stateless Architecture, based on XML.

- **SQL(SQL Server Management Studio)**
  SQL (Structured Query Language) is one of the most popular languages for storing, manipulating and retrieving data from databases. In Raptor, we store almost everything in the SQL powered databases, whether it is forms details, Quarter over Quarter filings, clients information, user permissions, security configurations, form configurations and many more.Current project under development maintains about 380 tables in the database, whereas deployed project manages about 600+ tables with approximately 85.6 Gigabytes of internal information of its clients and assets.

- **ASP.Net Web Framework**
  ASP.Net used in the currently deployed code of the Raptor. ASP.Net is the modern framework that uses the C# to access the services, generate the HTML based code passed to the ASP.Net with the help of javascript, which later added the generated components dynamically to the HTML and JS DOM.

- **Javascript, JQuery and LINQ**
  Javascript is the core of the Raptor, in the deployed application, it is responsible for building dynamic components with the help of JQuery and LINQ. In the development build, i.e. React application, the React library is itself the javascript library.

About the project architecture

Raptor started in the year '2012' with the objective to automate the assets filling with lots of auto validations, corrections and access the QoQ and keep aware its user of the financial terminologies and policies that are specific to country to country.

Raptor deployed build uses the SQL + C# Web Service + ASP.Net, whereas the current version under development uses SQL + C# Web Service + React'18 + Redux Framework. C# Web Service uses the MVC Architecture and React follows the Functional Components with global store implemented using the Redux with Saga middleware.

In the above project, I have been assigned to the React app development team, where I have assigned several tickets(tasks) planned to be assigned to the upcoming fresher.

- **Bottleneck of the QuestionContent Component:** Initially, Raptor uses the wrappers of the MUI components developed by another team, named, RAD, which is responsible for the providing the components with all the possible typescript integrated properties using Interfaces and Classes in core MUI components(Visit to know more about MUI Components: Link).

  In order to let you explain the above case study, you can think of the user customized 'text-field' taking around a second to render on the screen holding the data fetched from an api from the web service. Now, imagine such 560 text-fields are rendering synchronously in a single component holding data provided by O(cubic time) complexity.

  This scenario together renders in about 4-5 minutes, during this period the whole browser window freezes for about 2-3 minutes, this leads to a serious bottleneck of the application.

- **Develop a Question Preview Page:** Question Preview page enables the user to view the current status of this form application with privileges of read only status.

- **Develop a Question Review Page:** Question Review page multiple functionalities, like, Question UI Component displays sections wise data, but in our scenario, we can have multiple sections having similar questions with different values answered to them. In order to develop this page, proper understanding of the data schemas and data inputs is mandatory.

- **Develop Question Footer Utilities for Question UI Component:** Several question footer utilities need to be developed. Following are the utilities:
  - *Comment and Attachment:* Enable the user/client/Operations Executive to add a comment and Attachment wherever required.
  - *QoQ:* Enable the user/client/Operations Executive to check for the previous quarters filing for the current question.
  - *Calculation Tree:* Define the calculations for the related question when requested by user/client/Operations Executive.
  - *Related Questions:* There are few questions that hold the relations with other questions. To make the user/client/Operations Executive aware of the question dependency, this utility displays the list of related questions.
  - *FAQ:* Utility displays the list of frequently asked questions explained by the form publisher regulatory agency.
  - *Glossary:* Utility displays the list of Glossary of the keywords explained by the form publisher regulatory agency.
  - *Audit:* Enable the user/client/Operations Executive to view the complete Audit of the current question in the grid/tabular format.

- **Develop Bulk Actions Functionality for Question Ui/Question Review Component:** Bulk Actions is a drop down based button utility designed to enable users to perform bulk actions to their forms. Actions clubbed inside the Bulk Actions button for the question ui are:
    - Approve Level 1
    - Approve Level 2
    - Revoke Level 1
    - Revoke Level 2
    - Save All
    - Refresh
    - Question Preview
    - Question Review
    - Form Specific Functionalities, and

  Actions clubbed inside the Bulk Actions button for the Question Review Component are:
    - Approve Level 1
    - Approve Level 2
    - Revoke Level 1
    - Revoke Level 2
    - Save All
    - Refresh
    - Print Page

About the training period

Industrial training starts from January 10, 2023 (Tuesday) till February 20, 2023. During this training period, the company incorporated a lots of effort in providing a skilled trainer for each defined skillset mentioned below:

- SQL Basic Training
- ReactJs
- Programming Basics
- OOPs
- C# Language
- .Net Framework Basics
- NodeJS
- Testing
- Material UI
- Walkthrough of Financial Terms & Hedge Fund Ecosystem Overview
- Version control system - Git
- .Net core
- Business Etiquette and Email Writing Skills
- Interpersonal Communication
- Time Management
- Working in Teams
- Introduction to Capital Markets
- Asset Classes Overview
- ASP.NET
- Equities and Bonds
- Derivatives & SWAPs
- WebAPI
- Trade Lifecycle Overview
- ADO.NET
- Entity Framework
- Data Sets - positions, trades, NAV
- Design Patterns
- Final Case Study Noida

Following is the training schedule followed during the internship period-

Mon, Jan 9, 23	6 Hours	11:00 - 17:00	Joining Formalities
Tue, Jan 10, 23	1 hour	13:30 - 14:30	HR Induction
	30 Mins	14:30 - 15:00	IT Induction
	30 Mins	15:30 - 16:15	Firm Compliance Induction
	30 Mins	16:30 - 17:00	POSH Induction
	30 mins	17:00 - 17:30	Finance Induction
	30 Mins	17:30 - 18:00	Admin Induction
	1 hour	18:30 - 19:30	IVP Overview & IVP Client Overview_ Domain 101
	1 hour	19:30 - 20:30	Introduction to IVP Solutions
Wed, Jan 11, 23	1 / 5 days	10:00 - 18:00	SQL Basics Training
Thu, Jan 12, 23	2 / 5 days	10:00 - 18:00	SQL Basics Training
Fri, Jan 13, 23	3 / 5 days	10:00 - 18:00	SQL Basics Training
Mon, Jan 16, 23	4 / 5 days	10:00 - 18:00	SQL Basics Training
Tue, Jan 17, 23	5 / 5 days	10:00 - 18:00	SQL Basics Training
Wed, Jan 18, 23	1/3 day	10:00 - 18:00	ReactJs
Thu, Jan 19, 23	2/3 day	10:00 - 18:00	ReactJs
Fri, Jan 20, 23	3/3 day	10:00 - 18:00	ReactJs
Tue, Jan 31, 23	3 hours	13:30 - 17:30	Programming basics
			Object Oriented Programming -I
			Object Oriented Programming -II
	4 hours	17:30 - 21:30	C#
Thu, Feb 2, 23	3 hours	13:30 - 17:30	.Net framework basic
	3 hours	18:00 - 21:00	NodeJS
Tue, Feb 7, 23	3 hours	13:30 - 17:30	.Net framework basic
	1.5 hours	18:00 - 19:30	Testing
Thu, Feb 9, 23	4.5 hours	13:30 - 18:30	MUI
	1.5 hours	19:00 - 20:30	Walkthrough of Financial Terms & Hedge Fund Ecosystem Overview_Domain 101
Fri, Feb 10, 23	2 hours	14:00 - 16:00	Version control system - Git
	4 hours	16:30 - 21:00	.Net core
Mon, Feb 13, 23	3.5 hours	14:00 - 18:00	Business Etiquette and Email Writing Skills
	2 hours	18:30 - 20:30	.Net core
	30 mins	20:30 - 21:00	Case Study Overview NOIDA
Tue, Feb 14, 23	2 hours	14:00 - 16:00	Interpersonal Communication
	1.5 hours	16:30 - 18:00	Time Management
	30 mins	18:00 - 18:30	Working in Teams

	1 hour	18:30 - 19:30	Introduction to Capital Markets_Domain 101
	1 hour	20:00 - 21:00	Asset Classes Overview_Domain 101
Wed, Feb 15, 23	3 hours	13:30 - 17:30	ASP.NET
	1.5 hours	18:00 - 19:30	Equities and Bonds
	30 mins	19:30 - 20:00	Case Study Doubt Clearing
Thu, Feb 16, 23	2 hours	14:00 - 16:00	Derivatives & SWAPs
	1 hour	16:30 - 17:30	WebAPI
	1 hour	17:30 - 18:30	Trade Lifecycle Overview_Domain 101
	2 hours	18:30 - 20:30	ADO.NET
Fri, Feb 17, 23	4.5 hours	13:30 - 19:00	Entity Framework
	1 hour	19:00 - 20:00	Data Sets - positions, trades, NAV etc_Domain 101
Mon, Feb 20, 23	2 hours	14:00 - 16:00	Design Patterns
	30 mins	16:30 - 17:00	BA D/AD Catch Up
	30 mins	17:00 - 17:30	Domain 101 Evaluation
	2 hours	18:00 - 20:00	MCQ
Tue, Feb 21, 23	5 hours	15:00 - 20:00	Final Case Study Noida

About the project training

After the industrial training was completed, the company assigned me to the Raptor. Raptor development team has assigned two weeks training essential for their project and working domain. The training schedule is as follows-

ID	Actions	Date
1	Azure ,Tortoise Git and Time sheet(Actual)	February 23, 2023
2	Code overview and Architecture	February 23, 2023
3	Database structure and Important Tables	February 24, 2023
4	Dashboard	February 24, 2023
5	Form Creation	February 27, 2023 - March 01, 2023
6	Form UI	March 01, 2023 - March 03, 2023
7	File Download	March 03, 2023 - March 04, 2023
8	Configs and Access	March 04, 2023
9	App and DB servers of Raptor	March 04, 2023
10	Important Procs Information	March 09, 2023
11	React	March 09, 2023 - March 10, 2023
12	Raptor Functional Training	February 28, 2023

Related Works

- **SQLServer + .Net + React(TypeScript) based grid based project**

  Problem Statement: Security Master Solution For A Fund
  Create A Web-Based System To Display And Save The Security Information For A Hedge Fund. Security Master is a Repository of all securities that a fund is interested in buying. It maintains data of security fields necessary for various purposes which can be Identifiers, Terms and Conditions, Schedules, Reference data, etc.

  Security data is needed for multiple purposes:
  - Trading
  - Risk Management
  - Accounting
  - Corporate Actions

  A solution to this from programming perspective would ask for the following:
  - A database system to store all the relevant data.
  - A system (business layer) that can interact with the database to commit and retrieve data from the database as and when required.
  - An analysis of different Security types and various security fields that are required to service
  - the different requirements for the funds.
  - A Web User Interface to assist users to create and update security information

Above case study given by the company during the industrial training final phase. This case study demands the full stack solution using SQL, WebApi(.Net) and React based UI.

.Net WebApi Code is as follows:

EquitiesController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using AutoMapper;
using casestudy.Model;
using casestudy.DBLayer;
using casestudy.DTOs;

namespace casestudy.Controllers;

[ApiController]
[Route("api/[controller]")]
public class EquitiesController : ControllerBase
{
    private readonly ILogger<EquitiesController> _logger;
    public readonly DatabaseContext _DbContext;
    private readonly IMapper _mapper;
    public EquitiesController(ILogger<EquitiesController> logger, DatabaseContext
dbContext, IMapper mapper)
    {
        this._logger = logger;
        this._DbContext = dbContext;
        this._mapper = mapper;
    }

    [HttpGet]
    public async Task<IActionResult> getEquities()
    {
```

```
        return Ok(await _DbContext.Equities.ToListAsync());
    }

    [HttpGet]
    [Route("{EquityId}")]
    public async Task<IActionResult> getEquity([FromRoute] string EquityId)
    {
        var equity = await _DbContext.Equities.FindAsync(EquityId); //
.FirstOrDefault<Bond>()??

        if(equity == null)
        {
            return NotFound();
        }
        return Ok(equity);
    }

    [HttpPost]
    public async Task<IActionResult> addEquity(EquityDtoCreate newEquity)
    {
        var equity = new Equity();
        _mapper.Map(newEquity, equity);
        await _DbContext.Equities.AddAsync(equity);
        await _DbContext.SaveChangesAsync();
        return Ok(equity);
    }

    [HttpPut("{EquityId}")]
    public async Task<IActionResult> UpdateEquity([FromRoute] string EquityId,
EquityDtoUpdate updateEquity)
    {
        var equity = await _DbContext.Equities.FindAsync(EquityId);
        if(equity != null)
        {
            _mapper.Map(updateEquity, equity);
            await _DbContext.SaveChangesAsync();
            return Ok(equity);
        }
        return NotFound();
    }

    [HttpDelete("{EquityId}")]
    public async Task<IActionResult> deleteEquity([FromRoute] string EquityId)
    {
        var equity = await _DbContext.Equities.FindAsync(EquityId);
        if(equity != null)
        {
            _DbContext.Remove(equity);
            await _DbContext.SaveChangesAsync();
            return Ok(equity);
        }
        return NotFound();
    }
}
```

BondsController.cs

```
using Microsoft.AspNetCore.Mvc;
using casestudy.Model;
using casestudy.DBLayer;
using Microsoft.EntityFrameworkCore;
using casestudy.DTOs;
using AutoMapper;

namespace casestudy.Controllers;
```

```
[ApiController]
[Route("api/[controller]")]
public class BondsController : ControllerBase
{
    private readonly ILogger<BondsController> _logger;
    public readonly DatabaseContext _DbContext;
    private readonly IMapper _mapper;
    public BondsController(ILogger<BondsController> logger, DatabaseContext
dbContext, IMapper mapper)
    {
        this._logger = logger;
        this._DbContext = dbContext;
        this._mapper = mapper;
    }

    [HttpGet]
    public async Task<IActionResult> getBonds()
    {
        return Ok(await _DbContext.Bonds.ToListAsync());
    }

    [HttpGet]
    [Route("{BondId}")]
    public async Task<IActionResult> getBond([FromRoute] string BondId)
    {
        var bond = await _DbContext.Bonds.FindAsync(BondId);

        if(bond == null)
        {
            return NotFound();
        }
        return Ok(bond);
    }


    [HttpPost]
    public async Task<IActionResult> addBond(BondDtoCreate newBond)
    {
        var bond = new Bond();
         _mapper.Map(newBond, bond);
        await _DbContext.Bonds.AddAsync(bond);
        await _DbContext.SaveChangesAsync();
        return Ok(bond);
    }

    [HttpPut("{BondId}")]
    // [Route("{BondId}")]
    public async Task<IActionResult> UpdateBond([FromRoute] string BondId,
BondDtoUpdate updateBond)
    {
        var bond = await _DbContext.Bonds.FindAsync(BondId);
        if(bond != null)
        {
            _mapper.Map(updateBond, bond);
            await _DbContext.SaveChangesAsync();
            return Ok(bond);
        }
        return NotFound();
    }

    [HttpDelete("{BondId}")]
    public async Task<IActionResult> deleteBond([FromRoute] string BondId)
    {
```

```
        var bond = await _DbContext.Bonds.FindAsync(BondId);

        if(bond != null)
        {
            _DbContext.Remove(bond);
            await _DbContext.SaveChangesAsync();
            return Ok(bond);
        }
        return NotFound();
    }
}
```

Program.cs

```
using casestudy.DBLayer;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

const string corsPolicyName = "ApiCORS";

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddCors(options =>
{
    options.AddPolicy(corsPolicyName, policy =>
    {
        policy.WithOrigins("http://localhost:5039");
        policy.WithOrigins("http://localhost:3001");
        policy.WithOrigins("http://localhost:3000");
    });
});

builder.Services.AddAutoMapper(typeof(Program));

// s - injecting dbcontext to services
builder.Services.AddDbContext<DatabaseContext>(options=>{
    options.UseSqlServer(builder.Configuration.GetConnectionString("connstring"));
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseCors(corsPolicyName); // 👈 This should be above the UseStaticFiles();

app.UseStaticFiles(); // 👈 Below the UseCors();

app.UseAuthorization();

app.MapControllers();
```

```
app.Run();
```

There are supporting files for the above like Modal class, .csproj file, appsettings.json, Mapper.cs, DTOs.

Plan for the frontend development shown below:

```
-- header
      -- Security Master Demo
      -- Reset
-- Paper
      -- Grid
            -- Dropdown (onChange)
                  -- item: Equity
                  -- item: Corporate Bond
            -- Dropdown (onChange)
                  -- item: Equity.Security Summary
                  -- item: Equity.Security Identifier
                  -- item: Equity.Security Details
                  -- item: Equity.Risk
                  -- item: Equity.Regulatory Details
                  -- item: Equity.Reference Data
                  -- item: Equity.Pricing Details
                  -- item: Equity.Dividend History

                  -- item: Corporate Bond.Security Summary
                  -- item: Corporate Bond.Security Identifier
                  -- item: Corporate Bond.Security Details
                  -- item: Corporate Bond.Risk
                  -- item: Corporate Bond.Regulatory Details
                  -- item: Corporate Bond.Reference Data
                  -- item: Corporate Bond.Put Schedule
                  -- item: Corporate Bond.Pricing and Analytics
                  -- item: Corporate Bond.Call Schedule
            -- Button: Create
                  -- Navigation: Swipeable Drawer
                        -- Grid
                              -- Dropdown (onChange)
                                    -- item: Equity
                                    -- item: Corporate Bond
                        -- Grid
                              -- Form
      -- Grid row: (ith) Equity/Bond
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  TextField(Property Attribute's Data)
            .
            .
            .
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  (Update Icon):(onClick)
                              -- Enable the corresponding row as editable text field
```

```
                              -- Alert: Event Success? Alert severity = Success:
Alert severity = Error

      -- Grid row: (ith + 1)  Equity/Bond
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  TextField(Property Attribute's Data)
            .
            .
            .
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  (Update Icon):(onClick)
                              -- Enable the corresponding row as editable text field
                              -- Alert: Event Success? Alert severity = Success:
Alert severity = Error
      .
      .
      .
      -- Grid row: (ith + n) Equity/Bond
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  TextField(Property Attribute's Data)
            .
            .
            .
            -- Grid column
                  TextField(Property Attribute's Data)
            -- Grid column
                  (Update Icon):(onClick)
                              -- Enable the corresponding row as editable text field
                              -- Alert: Event Success? Alert severity = Success:
Alert severity = Error


-- footer
      -- Security Master Demo @ CS14_SS
```

- **IVP's React Components Wrapper Optimization for Raptor**

  Problem Statement: Optimize the logic for the IVP React Raptor Question Component.
  QuestionComponent uses the IVPTextFieldWrapper for creating a form based component. For question number 26, section 2, form tab the rendering time for the question to be visible on the browser's screen is around 50 seconds in the development environment. For an ideal application, this turnaround time should be < 1 second.

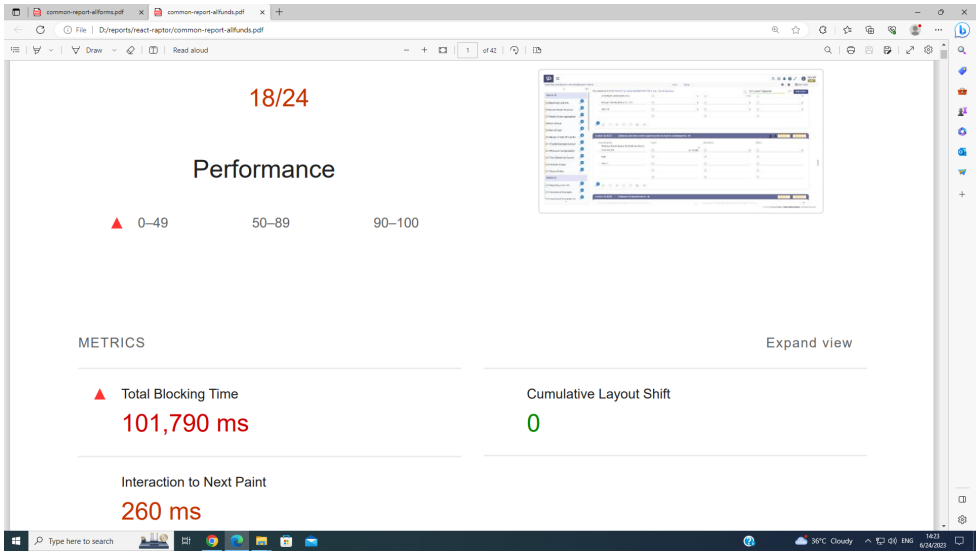
  Solution presented to the above problem statement are follows -

I had presented the complete POC (Proof of Context) compared to the HTML <input> tag, MUI <TextField> and IVP's <IVPTextField> component. The result of the POC came out to be two main conclusions -

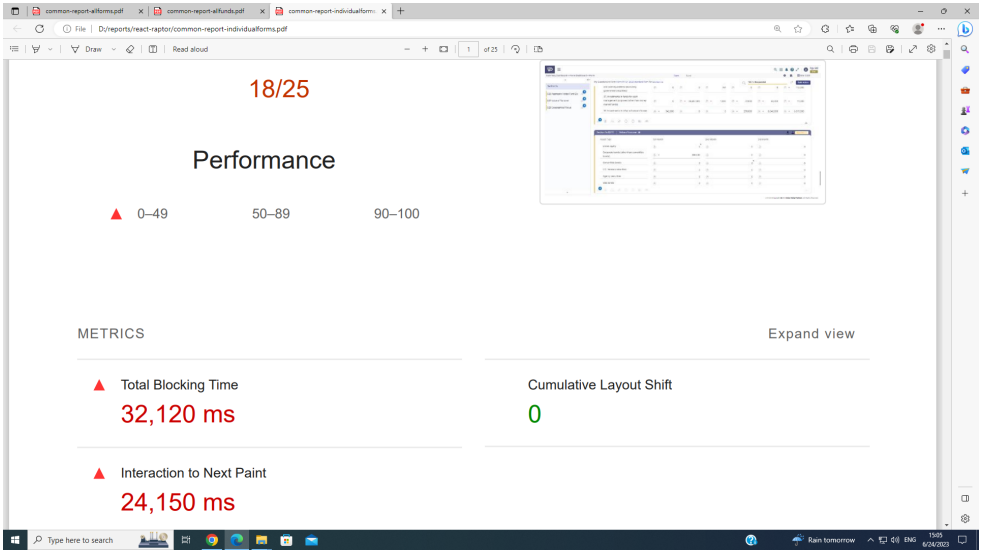
1. Using of Arrays.map in the legacy code resulting in the very high space complexity that ends up heavy DOM
2. IVP's <Textfield> tag created as the generalized component that has set the lot of attributes to some common fixed value to it, which results in the returning a component 10 times heavier than HTML <input>.
3. Most of the code consists of the nested map, i.e., time complexity is $O(n^3)$, where n is the number of text fields in a particular question. This results in heavy rendering of the component and locking the screen for more than a minute.

Some snips of the above reports are shown below:

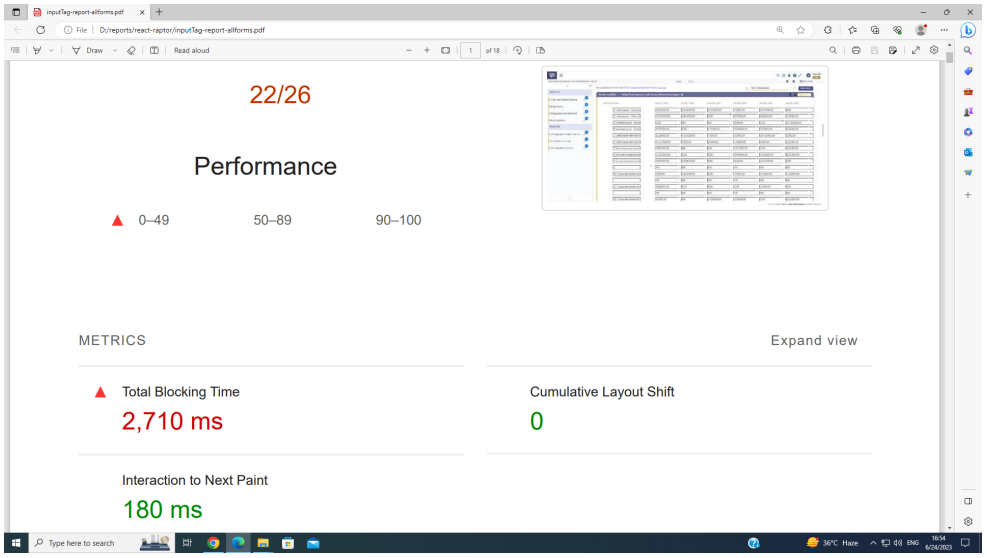
Initially, the total blocking time for the component was around 101000 ms.



Using the same data using MUI <Textfield>, the total blocking time reduces to 24150 ms.



Again, using the html <input> tag, the total blocking time was 2710 ms (~50 times less as compared to the IVP's <TextField>)



This POC takes about 40 hours to return with the above conclusions. This helps the developers team to identify the cause for the performance bottleneck in the application.

- **Debug based tasks for the requirement collection for the project work**
  Problem Statement: *Debug the deployed application to collect all the requirements for the Question Preview, Question Review component, and Bulk Actions.*

  Conclusion for the above task is the successful collection all the UI/UX requirements, API integrations, Web Services, Store Procedures and get them approved by one of the senior software engineer and director as well.It takes about 30 hours to successfully completion of the task

- **Consolidated SQL Script Generator for the Annex4 FAQ**
  Problem Statement: *Generate a complete SQL side solution i.e, creating a new table for Annex4 FAQ, insert the details in the table mentioned in the documentation, generate the complete script [create table, insert table, delete from table, stored procedure to handle the query fetched from question ui footer component].*

  Conclusion for the above task is the successful completion of the task by getting the script reviewed by the senior software engineer and director. The above tasks take around 16 hours of development efforts.

Project Work

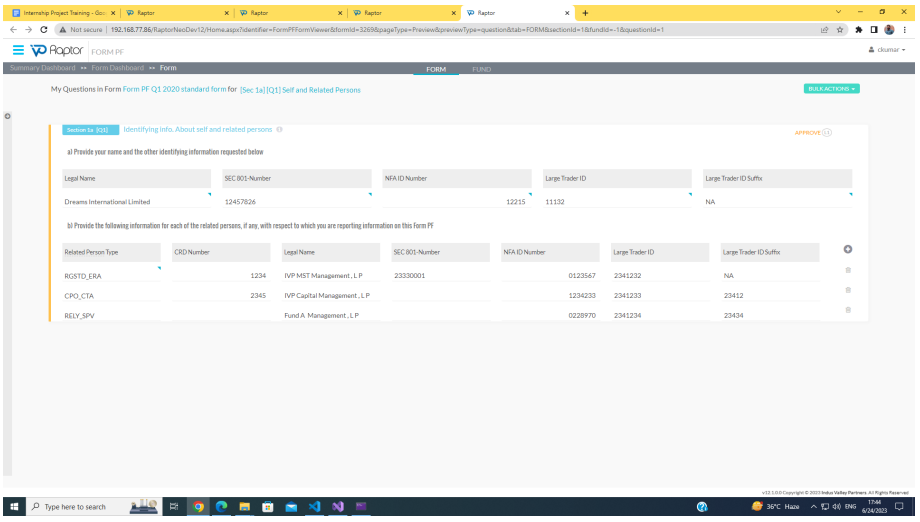
The main project work consists of most of the development in the Frontend tech - React(Typescript), Redux, CSS Styling and Javascript. Backend web services are developed if required for these task using the C# and LINQ.

Task 1: Question Review Component developed for the 'Raptor' project.
Problem Statement: Develop a question review component for the Raptor as per the existing application. Use the requirements collected during the task "**Debug based tasks for the requirement collection for the project work**" to complete the task within estimated effort of 10 days(2 weeks).

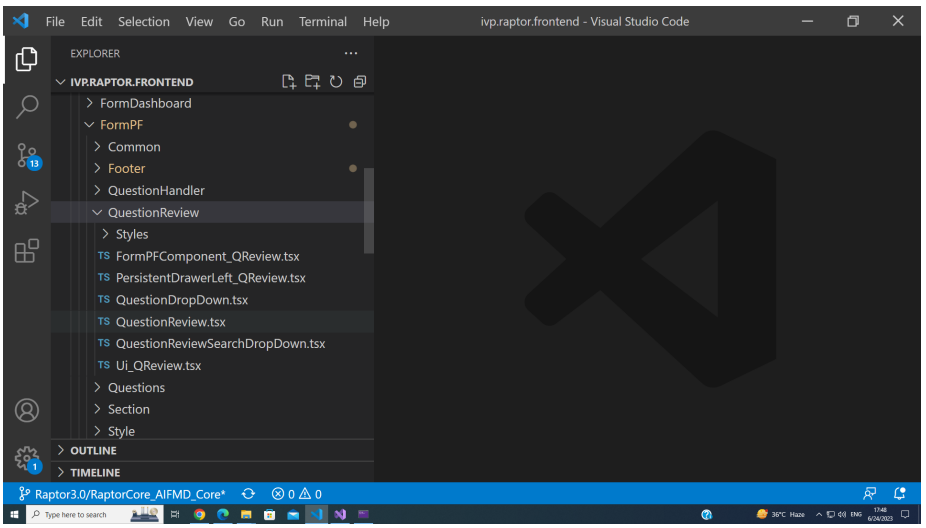
Question Review component enable its user to due multiple actions in their form application, some of the most important actions among them are-

- Multiple Question Filling
- Multiple Filling Approve/Reject
- Complete Filling Approve/Reject at once
- Save multiple edits
- Revert multiple edits
- Filling the details for the questions presented in multiple sections in the form without jumping to each section.
- Print their filling to the pdf/print to page.

The design and requirements were taken from the existing Raptor Application. Screen captures are as follows:



Development Hierarchy of the code is:



In the above directory structure, the main folder for the component is the “Question Review”. Inside the main folder, main component or the starting point of the development process begins from the FormPFComponent_QReview.tsx, where the return statement code is as follows:

```
return (
        <div id="pageContainer" className={cssClasses.FormPFComponent_pageTabPageContainer}>
          <Box className={cssClasses.FormPFComponent_OuterBox}>
            <Box className={cssClasses.FormPFComponent_InnerBox}>
              <div className={cssClasses.FormPFComponent_OuterDiv}>
                <div className={cssClasses.FormPFComponent_InnerDiv}>
                  <Link className={cssClasses.FormPFComponent_pageTabBreadCrumpDisabled} to="/">
                    <div className={cssClasses.FormPFComponent_pageTabBreadCrumpDisabled}>Summary Dashboard</div>
                  </Link>
                  <div className={cssClasses.FormPFComponent_pageTabBreadCrumpArrow}>&gt;&gt;</div>
                  <Link className={cssClasses.FormPFComponent_pageTabBreadCrumpDisabled} to={formSummaryUrl}>
                    <div className={cssClasses.FormPFComponent_pageTabBreadCrumpDisabled}>
                      Form Dashboard
                    </div>
                  </Link>
                  <div className={cssClasses.FormPFComponent_pageTabBreadCrumpArrow}>&gt;&gt;</div>
                  <div className={cssClasses.FormPFComponent_pageTabBreadCrump}>Form</div></div>
              </div>
              <div className={cssClasses.FormPFComponent_TabDiv}>
                <IvpTabControl
                  onChange={tabChangeHandler}
                  tabsObj={tabsObj}
                  variant='IvpTabControlSecondaryStandard'
                  controlId="1"
                  tabStyling={{fontSize: "0.813rem"}}
                />
              </div>
            </Box>

            {tab == "Form" && <Ui_QReview refresh={refresh} setRefresh={setRefresh} tab={tab} formId={formId} fundId={fundId} questionId={questionId}
            initialView={initialView} previewType="question" />}
            {tab == "Fund" && <Ui_QReview refresh={refresh} setRefresh={setRefresh} tab={tab} formId={formId} fundId={fundId} questionId={questionId}
            initialView={initialView} previewType="question" />}

          </Box>
        </div>
  )
```

This return method will call the Ui_QReview.tsx file with the defined properties. Ui_QReview.tsx file is responsible for developing the complete Question UI for the question review page. The return statement for the file is as follows:

```
return (FormPFreviewJSONdataLoading ?<ThreeDotSpinner/>:
        <div id="FormPFUi_QuestionReview">
          <Grid container spacing={2}>
            <Grid item xs={open ? 2.2 : 0.2} sm={open ? 2.2 : 0.2} md={open ? 2.2 : 0.2}>
              <div>
                <PersistentDrawerLeft_QReview
                  initialView={props.initialView}
                  previewType={props.previewType}
                  fundId={props.fundId}
                  setFundSelected = {setfundSelected}
                  questionSelected={questionSelected}
                  FormProperties={FormProperties}
                  data={props.tab == "Fund" ? LeftMenuQuestionData: undefined}
                  open={open}
                  setOpen={setOpen}
                />
              </div>
            </Grid>
            <Grid item xs={open ? 12 - 2.2 : 12 - 0.2} sm={open ? 12 - 2.2 : 12 - 0.2} md={open ? 12 - 2.2 : 12 - 0.2}>
              <div >
                <QuestionReviewDropDown
                  initialView = {props.initialView}
                  printState={ printState}
                  setPrintState={setPrintState}
                  tab={props.tab}
                  refresh={props.refresh}
                  setRefresh={props.setRefresh}
                  FormProperties={FormProperties}
                  previewType={props.previewType}
                  formDetailsQuestionReview={FormPFformDetailsQuestionReview}
                  questionSelected={questionSelected}
                  setSelectedQuestion={getQuestion}
```

```
                    setQuestion={setQuestion}
                    questionData={questionData}
                  />
                </div>
                <div id="questionContainer">
                        <div data-testid="FormPFUiquestionContainer" id="questionContainerInner" className={open == false ? cssClasses.FormPFUI_navBodyClose :
                        cssClasses.FormPFUI_navBodyOpen}>
                                < QuestionFilter
                                    formId = {FormProperties.formId}
                                    ref={componentRef}
                                    questionData={questionData}
                                    setQuestion={setQuestion}
                                    FormProperties={FormProperties}
                                    sectionSelected={questionSelected}
                                    tab={props.tab}
                                    previewType={props.previewType}
                                  />
                         </div>
                </div>
              </Grid>
            </Grid>
          </div>

  )
```

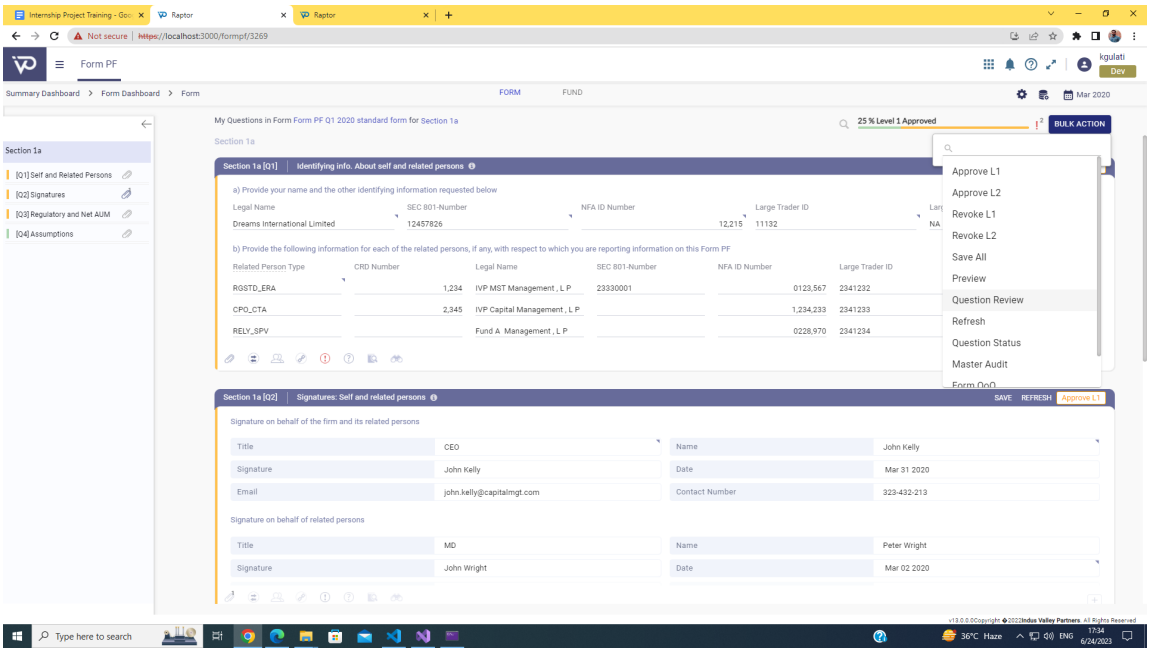
There are four main child components in the file, namely **ThreeDotSpinner, PersistentDrawerLeft_QReview, QuestionReviewDropDown, QuestionFilter.**

- **ThreeDotSpinner:** Responsible for showing the loading until the data is completely fetched from the API.
- **PersistentDrawerLeft_QReview:** Responsible for displaying a responsive vertical menu drawer if there are more than one funds for the same question.
- **QuestionReviewDropDown:** Responsible for displaying a bulk actions button with the dropdown of actions of Questions Review page.
- **QuestionFilter:** Responsible for creating the question UI for the Question Review page with the header utilities enabled and footer utilities disabled.

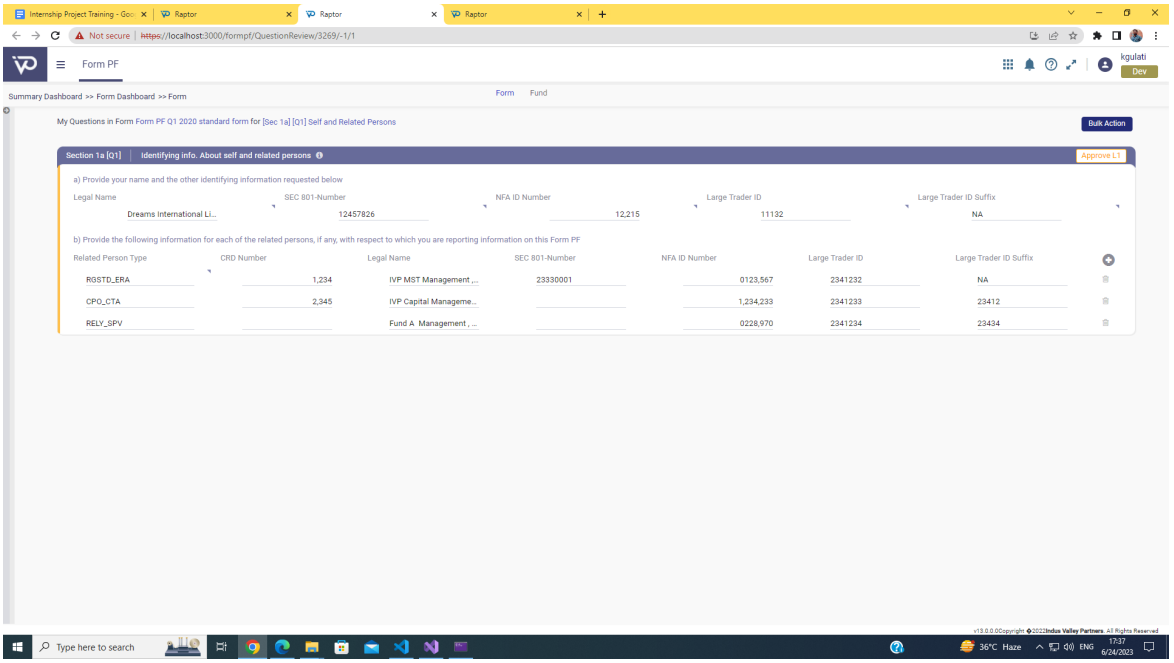
The following function will render the Question Review Page to the new tab on clicking on “Question Review” in the Question UI page.

```
const ViewOnlyFormQuestionView = () => {
      var formId = props.FormProperties.formId;
      var freezedQuestionId = "1";
      var questions = $('#questionContainer').children().children();
      $.each(questions, function (index, ele) {
         if ($(ele).position().top > 0) {
            freezedQuestionId = $(ele)?.attr('id');
            if (freezedQuestionId != undefined)
               return false;
         }

      });
      var tempData = freezedQuestionId.split('_');
      var fundId = tempData[2];
      var qId = tempData[3];
      var url = "/formpf" + "/QuestionReview/" + formId + "/" + fundId +"/" + qId;
      var win = window.open(url, '_blank');
      win.focus();
   }
```

Preview of the Question UI Component is as follows:



Preview of the Question Review component is as follows:



Task 2: Question Preview Component

Problem Statement: Using the existing code, create the logic and develop the Question Preview page for the Raptor. Use the requirements collected during the task "**Debug based tasks for the requirement collection for the project work**" to complete the task within estimated effort of 7 days(1.5 weeks).

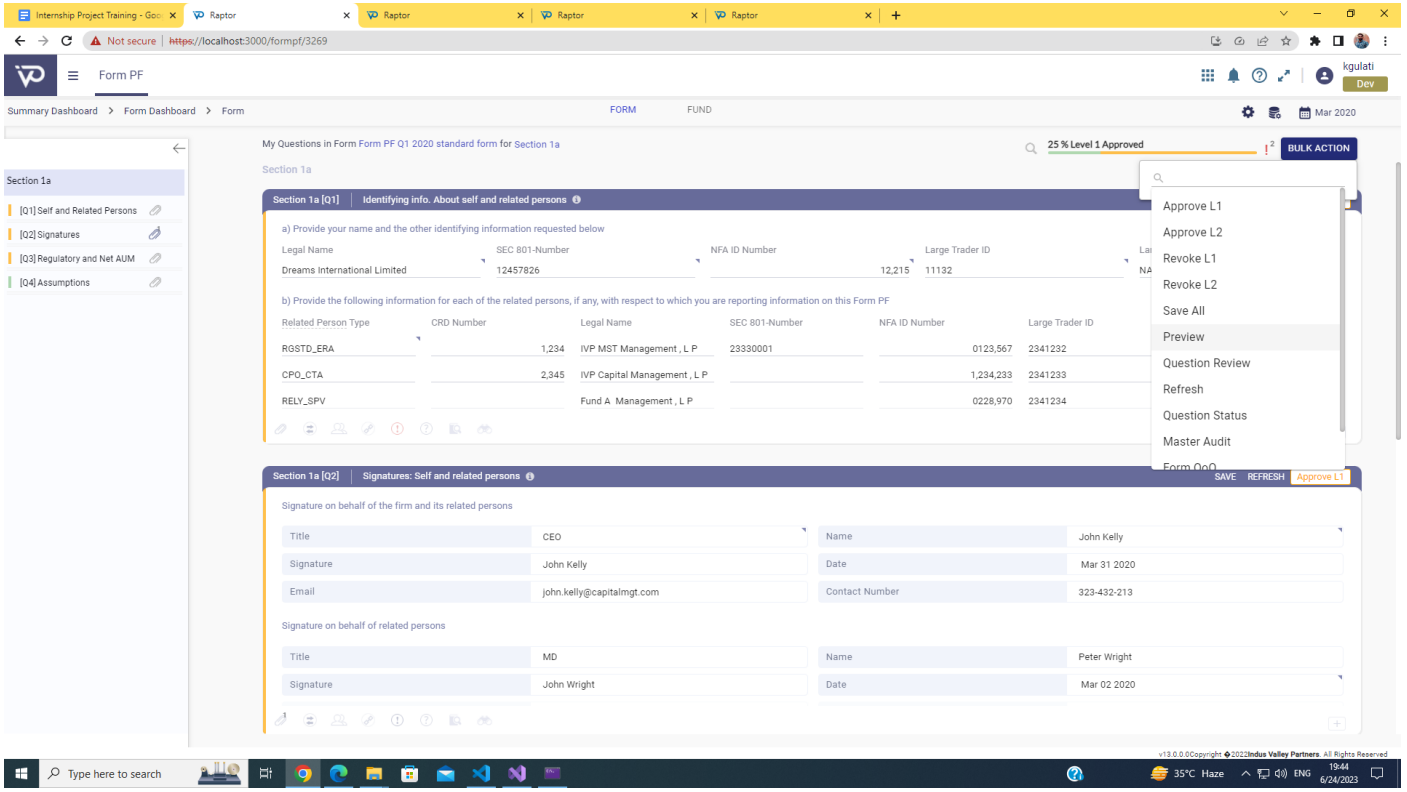
Question Preview Component enable its user to walkthrough the form application without disturbing its value filling informations or changing any form configuration throughout the walkthrough period, as the Question Preview disables all the editable fields throughout the application and also hides all the question ui header/footer and bulk actions utilities from the user.

Since the component is reusing the same code, multiple checks (if…else blocks and ternary operations) throughout the react application code have been set according to the requirements specified.
Following is code of the root component with the props responsible handling the inner checks for the preview page.

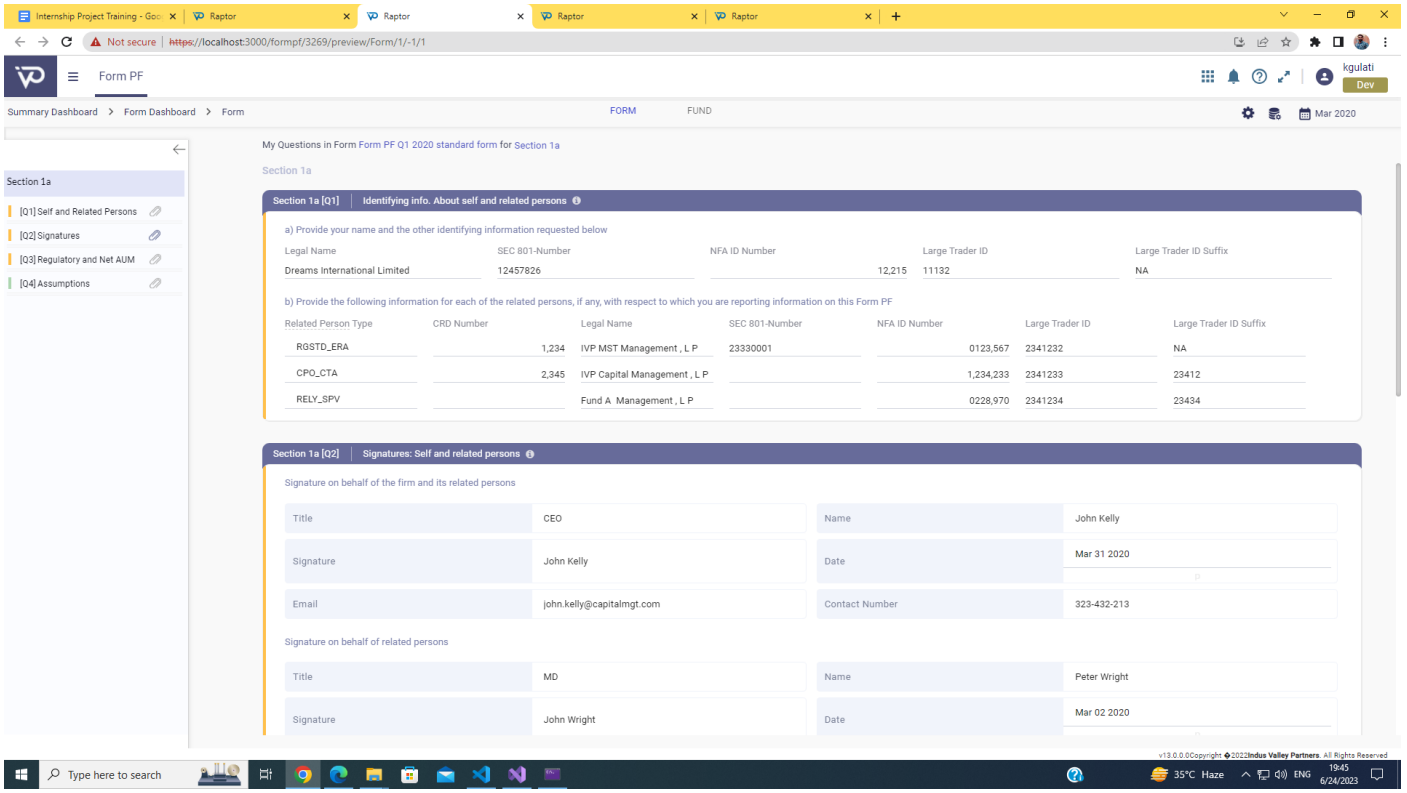
```
return (<>
    {isRender() && <ThemeProvider theme={theme}>
      <div id="pageContainer" className={cssClasses.FormPFComponent_pageTabPageContainer}>
        <Box className={cssClasses.FormPFComponent_OuterBox}>
          <Box className={cssClasses.FormPFComponent_InnerBox}>
            <div className={cssClasses.FormPFComponent_OuterDiv}>
              <div className={cssClasses.FormPFComponent_InnerDiv}>
                <CustomBreadcrumbs breadcrumbs={breadcrumbs} />
              </div>
            </div>
            <div className={cssClasses.FormPFComponent_TabDiv}>
                  <IvpTabControl onChange={tabChangeHandler} tabsObj={tabsObj} variant='IvpTabControlSecondaryStandard' controlId="1"
                  tabStyling={{fontSize:"0.813rem"}}
              />
            </div>

            <div className={cssClasses.FormPFComponent_OuterFormConfigDiv}>
              <div className={cssClasses.FormPFComponent_InnerFormConfigDiv}>
                    {formconfigpermission?.[0]?.IsConfig && <Link className={cssClasses.FormPFComponent_pageTabBreadCrumpDisabled} to={formConfigUrl}
                    > <i className={`fa fa-cog fa-2 ${cssClasses.FormPFComponent_pageTabFormConfigImage}`}>
                  </i></Link>}
                     
                    <Link className={cssClasses.FormPFComponent_pageTabBreadCrumpDisabled} to={formTemplateUrl} ><img alt="Template Upload Menu"
                    className={cssClasses.FormPFComponent_pageTabTemplateUploadIcon} src={String(TemplateUploadButton)} /></Link>
                     
                  <CalendarMonthIcon className={cssClasses.FormPFComponent_pageTabCalenderIcon} />
                  <div className={cssClasses.FormPFComponent_pageTabFormdate}>{formpfdate}</div>
              </div>
            </div>

          </Box>

          {tab == "Form" && <FormPFUI tab={tab} formId={formId} sectionId={sectionId} fundId={fundId} questionId={questionId} pageType={props?.pageType}
          previewInitialState={previewInitialState} />}
          {tab == "Fund" && <FormPFUI tab={tab} formId={formId} sectionId={sectionId} fundId={fundId} questionId={questionId} pageType={props?.pageType}
          previewInitialState={previewInitialState} />}

        </Box>
      </div>
      </ThemeProvider>
    }
  </>
  );
```

pageType prop will make sure that the component rendering is the Question Preview Page or Question UI Page, whereas, If the *pageType* sets to "*preview*" than, the *previewInitialState* will be *true* and once the complete Question Preview page is loaded, it blocks all the APIs and React states uses inside the application code and if the pageType sets to "*question*", it will be false, thus all the APIs and React states are able to get modified and performs their operations.

Preview of the Question UI Component is as follows:



Preview of the Question Preview component is as follows:



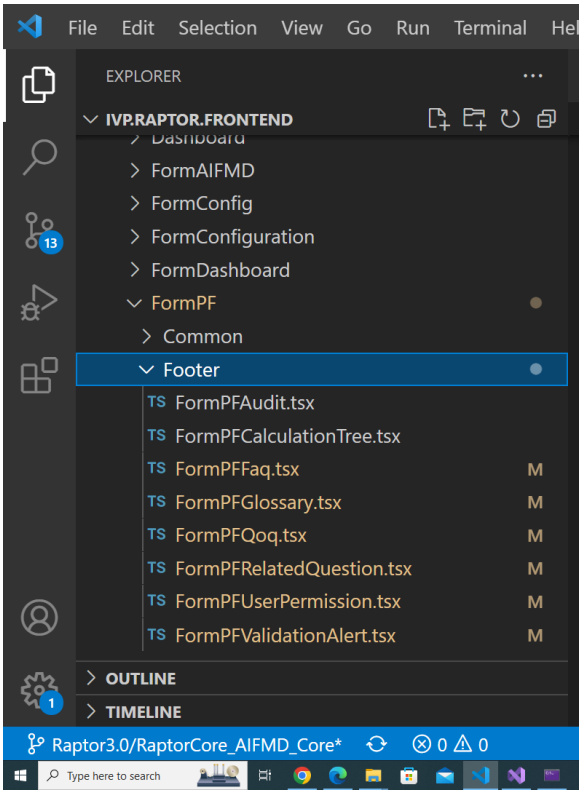
Task 3: Question Footer Component

Problem Statement: Develop all the Question Footer Utilities for the Raptor. Use the requirements collected during the task "**Debug based tasks for the requirement collection for the project work**" to complete the task within estimated effort of 20 days(~ 4 weeks).

Question Footer is a set of utilities that enable its user to perform certain tasks on a particular question in the form filling. These operations are -

- **Comment and Attachments**: A user can attach a comment for other users, share an attachment or can reply over the comment on other users. They can also delete their comment in this utility.
- **QoQ(Quarter over Quarter):** This utility enables its user to look back in their previous form filling for the entries they had filled earlier.
- **User Permissions:** A user can see the list of users accessing their fillings and level of access they have over their fillings.
- **Related Questions:** A user can see the number of related questions to a specific question, and can jump to the related question directly whether the question is present in the same section or other section.
- **Validation Alerts:** A smart utility that detects whether the user enters the correct information or not.
- **FAQs:** List of all the frequently asked questions for the question provided by the regulators.
- **Glossary:** All the available keywords in the question with their explanations are listed in a tabular form.
- **Audit:** All the changes made by different users can be easily seen in the Audit utility.
- **Calculation Tree:** This utility enables the user to look into the calculation for some of the fields that 'Raptor' automatically fills using the existing knowledge of the data provided by users for the question.

Directory for the Question Footer Utilities as follows:



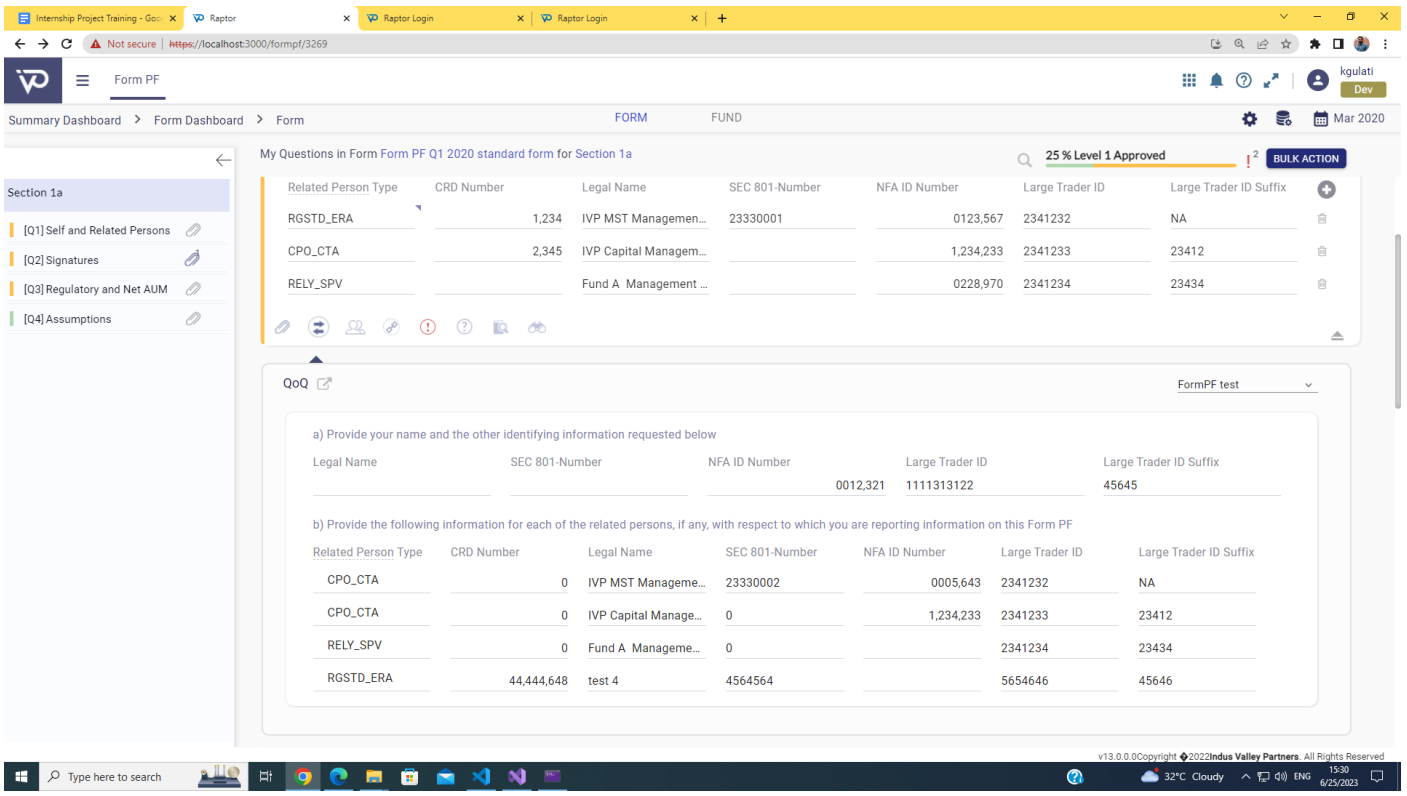
A sample return code for the component of Audit.tsx is shown below-

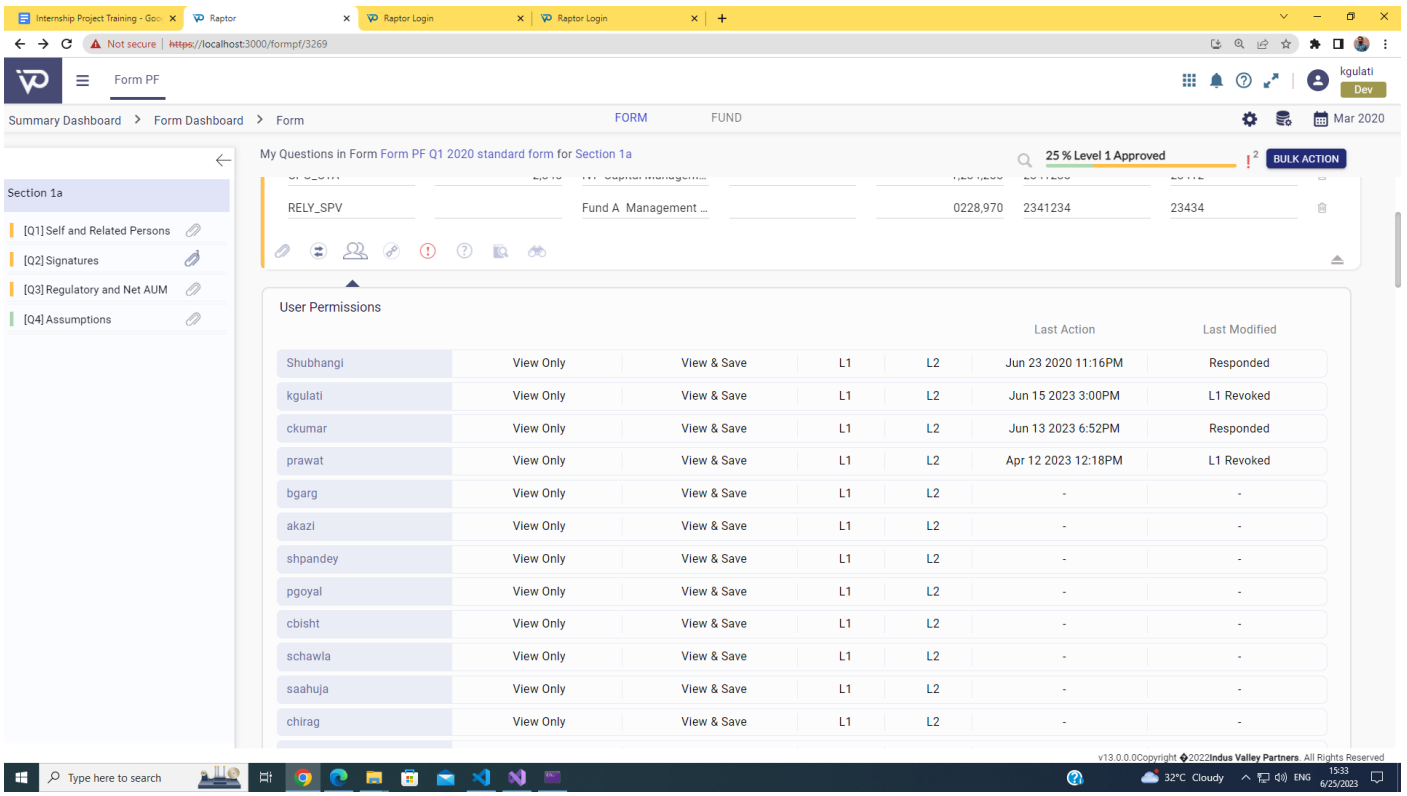
```
return (<div id={props.FormProperties.formId + "_" + props.FormProperties.currentFundId + "_" + props.questionId + "_" + "support"}
className={cssClasses.Audit_questionSupport}>
      <div id={props.FormProperties.formId + "_" + props.FormProperties.currentFundId + "_" + props.questionId + "_support_iframe_Audit"
      } className={cssClasses.Audit_ActiveArrow} style={{ 'left': props.leftStyle }}></div>
      <div className={cssClasses.Audit_outerdiv}>
         <div className={cssClasses.Audit_header}>
         <div className={cssClasses.Audit_commentheading}>
            AUDIT
            </div>
            <div className={cssClasses.Audit_tab}>
               <IvpTabControl onChange={tabChangeHandler} tabsObj={tabsObj} variant='IvpTabControlSecondaryStandard' controlId="1"
               tabStyling={{fontSize : "0.813rem"}}
                />
            </div>
         </div>
         {GetAuditWrapper(AuditData, ApprovalAuditData,  ManualAuditData,ApprovalAuditDataSnapshot,IsAuditSnapshot)}

      </div>
   </div>);
```

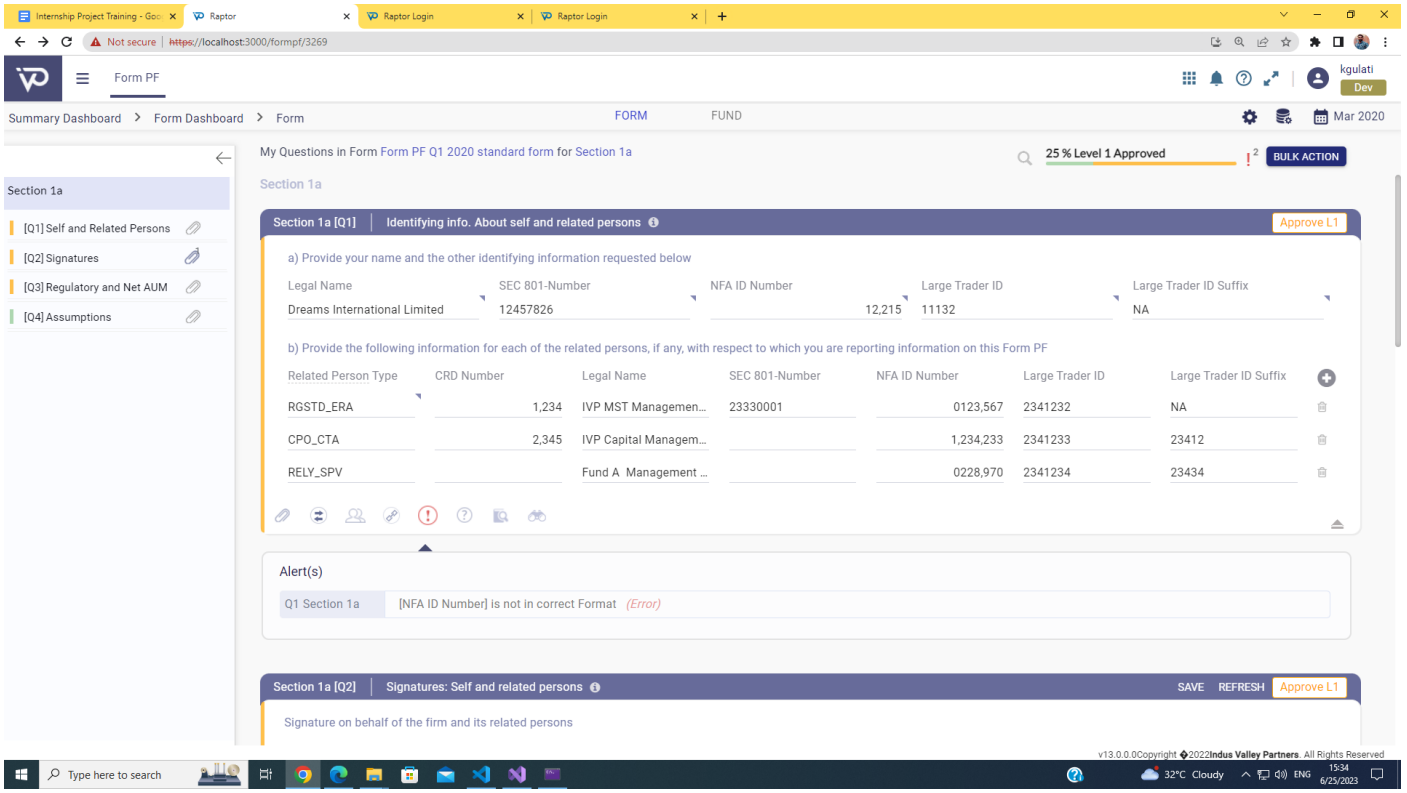
A similar structure is used to develop all the utilities for the Question Footer. In the above code, **GetAuditWrapper** is the main method which takes input as data fetched from the redux saga using dispatch method of saga, all the parameters are further displayed using the Grid Layout.

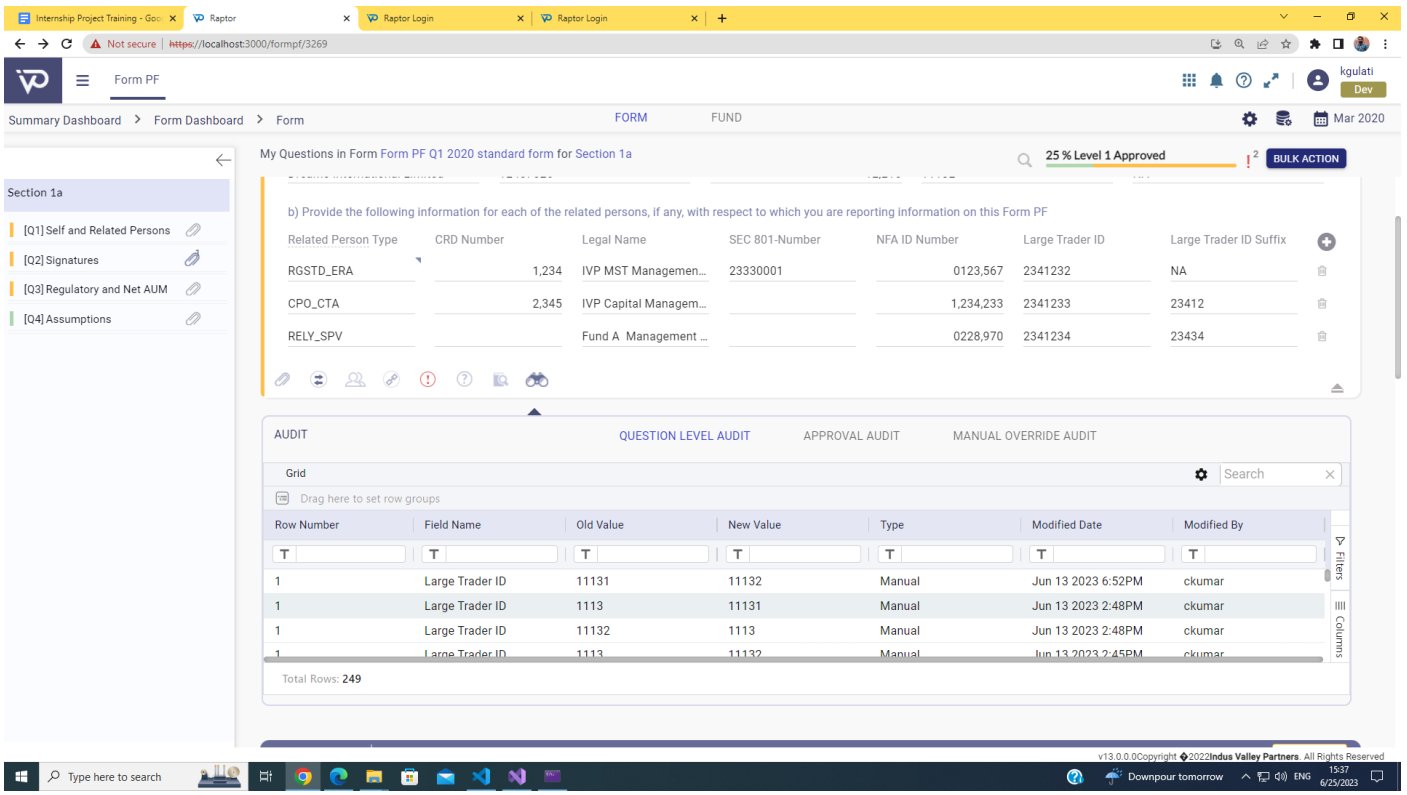
```
const GetAuditWrapper = (AuditData: any, ApprovalAuditData: any, ManualAuditData: any,ApprovalAuditDataSnapshot:any,IsAuditSnapshot:boolean) => {
      return (
         <>
            {
               tab == "Question Level Audit" &&
               <div className={cssClasses.Audit_grid}>
              <AdaptableGrid ref={GridRef} adaptableOptions={AuditData["adaptableOptions"]} GridProp={AuditData["gridProp"]} gridOptions={AuditData["gridOptions"]}
              />
               </div>
            }
            {
               tab == "Approval Audit" &&
               <div className={cssClasses.Audit_grid}>
                {
                IsAuditSnapshot ?
                <>
                <i className={`fa fa-times ${cssClasses.Audit_gridClose}`} aria-hidden="true" onClick={()=>setIsAuditSnapshot(false)}></i>
                           <AdaptableGrid ref={GridRef} adaptableOptions={ApprovalAuditDataSnapshot["adaptableOptions"]}
                           GridProp={ApprovalAuditDataSnapshot["gridProp"]} gridOptions={ApprovalAuditDataSnapshot["gridOptions"]}  />
                </>
                :
                           <AdaptableGrid ref={GridRef} adaptableOptions={ApprovalAuditData["adaptableOptions"]} GridProp={ApprovalAuditData["gridProp"]}
                           gridOptions={ApprovalAuditData["gridOptions"]}  />
                }
               </div>
            }
            {
               tab == "Manual Override Audit" &&
               <div className={cssClasses.Audit_grid}>
              <AdaptableGrid ref={GridRef} adaptableOptions={ManualAuditData["adaptableOptions"]} GridProp={ManualAuditData["gridProp"]}
              gridOptions={ManualAuditData["gridOptions"]}  />
               </div>
            }

         </>

      );
   }
```

Screen Snips of the UI for the utilities are as follows-






Internship Project Training - Goo
Raptor
Raptor Login
Raptor Login
Not secure | https://localhost:3000/formpf/3269
Form PF
kgulati
Dev
Summary Dashboard > Form Dashboard > Form
FORM
FUND
Mar 2020
My Questions in Form Form PF Q1 2020 standard form for Section 1a
25 % Level 1 Approved
BULK ACTION
Section 1a
[Q1] Self and Related Persons
[Q2] Signatures
[Q3] Regulatory and Net AUM
[Q4] Assumptions
Section 1a [Q1] | Identifying info. About self and related persons
Approve L1
a) Provide your name and the other identifying information requested below
Legal Name
SEC 801-Number
NFA ID Number
Large Trader ID
Large Trader ID Suffix
Dreams International Limited
12457826
12,215
11132
NA
b) Provide the following information for each of the related persons, if any, with respect to which you are reporting information on this Form PF
Related Person Type
CRD Number
Legal Name
SEC 801-Number
NFA ID Number
Large Trader ID
Large Trader ID Suffix
RGSTD_ERA
1,234
IVP MST Managemen...
23330001
0123,567
2341232
NA
CPO_CTA
2,345
IVP Capital Managem...
1,234,233
2341233
23412
RELY_SPV
Fund A Management ...
0228,970
2341234
23434
Alert(s)
Q1 Section 1a
[NFA ID Number] is not in correct Format (Error)
Section 1a [Q2] | Signatures: Self and related persons
SAVE
REFRESH
Approve L1
Signature on behalf of the firm and its related persons
v13.0.0.0Copyright ©2022Indus Valley Partners. All Rights Reserved
Type here to search
32°C Cloudy
ENG
15:34
6/25/2023


Internship Project Training - Goo
Raptor
Raptor Login
Raptor Login
Not secure | https://localhost:3000/formpf/3269
Form PF
kgulati
Dev
Summary Dashboard > Form Dashboard > Form
FORM
FUND
Mar 2020
My Questions in Form Form PF Q1 2020 standard form for Section 1a
25 % Level 1 Approved
BULK ACTION
Section 1a
[Q1] Self and Related Persons
[Q2] Signatures
[Q3] Regulatory and Net AUM
[Q4] Assumptions
b) Provide the following information for each of the related persons, if any, with respect to which you are reporting information on this Form PF
Related Person Type
CRD Number
Legal Name
SEC 801-Number
NFA ID Number
Large Trader ID
Large Trader ID Suffix
RGSTD_ERA
1,234
IVP MST Managemen...
23330001
0123,567
2341232
NA
CPO_CTA
2,345
IVP Capital Managem...
1,234,233
2341233
23412
RELY_SPV
Fund A Management ...
0228,970
2341234
23434
AUDIT
QUESTION LEVEL AUDIT
APPROVAL AUDIT
MANUAL OVERRIDE AUDIT
Grid
Search
Drag here to set row groups
Row Number
Field Name
Old Value
New Value
Type
Modified Date
Modified By
1
Large Trader ID
11131
11132
Manual
Jun 13 2023 6:52PM
ckumar
1
Large Trader ID
1113
11131
Manual
Jun 13 2023 2:48PM
ckumar
1
Large Trader ID
11132
1113
Manual
Jun 13 2023 2:48PM
ckumar
Total Rows: 249
Filters
Columns
v13.0.0.0Copyright ©2022Indus Valley Partners. All Rights Reserved
Type here to search
Downpour tomorrow
ENG
15:37
6/25/2023

Results

Efforts taken for each component

Task Name	Estimated Efforts	Actual Efforts
Question Review Component developed for the ‘Raptor’ project	10 days	12 days
Question Preview Component	7 days	6 days
Question Footer Component	20 days	18 days

Completion of component

Task Name	Development Completed
Question Review Component developed for the ‘Raptor’ project	Yes
Question Preview Component	Yes
Question Footer Component	Yes

Submit for the production

Task Name	Sent to deployment
Question Review Component developed for the ‘Raptor’ project	Under testing
Question Preview Component	Yes
Question Footer Component	Yes

Conclusion

Added skills

During the complete internship period, code management using github, requirement gathering for development, legacy code debugging and development following SDLC are some of the core skills added to my skill set.

Advantage of working in the team

Throughout the internship period, I have got the opportunity to work with the team of 8 SDE. I have learnt all the skills during this journey with the help of their experience from them. After these six months, I am thankful to the efforts of my seniors who have assigned the tasks to me in the manner that helps me set a progressive step to complete my internship successfully and continue my career as Associate SDE for the Raptor.

Experience the rich culture of healthy office suite

I had experienced the rich culture of the healthy office suite. My seniors guide me when I get stuck in the development phase. Other than work, we had 36 small parties as a part of team building activities, 2 quarterly parties and a memorable trip to Rishikesh during the month of May. All these activities and trips help me get connected with everyone, understand their roles and responsibilities, and step up to become a valuable team player in these months.

Future Scope

After completion of the internship period, the team has planned several tasks over the next 6 sprints. Some of the following are-

- **Work on different forms like Form AIFMD, CPO, H16 and 50 others**
  Similar to form ‘FormPF’, there are around 55 forms that are in the pipeline to be developed during the tenure of next 8 months. Together the team of 8 members and 2 more new joiners who will join in the month of July will be assigned different forms modules as per the plan set by the team lead. Out of these 55 forms, three forms AIFMD, CPO and H16 are in the development phase. From the remaining 51 forms, 12 forms will be assigned to me in the coming months. For them, I will be expected to use my experience throughout the internship period to successfully complete these modules and will help the juniors if they get stuck in between.

- **Work on the client side, resolving their issues within promised time period**
  Other than the form modules development, the team has assigned each engineer to the client side for a certain period of time. The team lead has planned to set me to the client side from October’23 to November’23. During this tenure, I will get the opportunity to understand Raptor from a business point of view, which will guide me in my upcoming journey with the raptor.

- **Train other team members who will switch from the ASP.NET to React**
  There are senior engineers who have not switched to React from the ASP.Net. From the month of September’23 , everyone will start working in React Development to give the boost to the development of the new application. For the same, the team lead assigned an additional responsibility of developing a curriculum for the team so that everyone who will be switching to the React, will take up the guided course and successfully start contributing to the development phase with less effort in learning the React development on their own.

- **Build the project training schedule for the upcoming freshers**
  For the July joiners, the lead assigned the role to create the Project Training plan for the juniors by the end of June’23.