DATA MIGRATION FRAMEWORK & INVOICE AUDIT MANAGEMENT SYSTEM

An Internship Report for

CSA 625: Industry Internship / Software Project

Credits: 16

Submitted in partial fulfillment of Master's Degree

Master of Computer Applications

By

SANKET GOVIND NARVEKAR

Seat Number: 2210

ABC ID: 225647951276

PRN: 201308756

Under the Mentorship of

MR. ROHIT KOTHAWADE

Goa Business School

Discipline of Computer Science and Technology

Goa University

Date: 6th June 2024

Examined by:

Seal of the School/Dept

DECLARATION BY STUDENT

I hereby declare that the data presented in this Internship report entitled, “DATA MIGRATION FRAMEWORK & INVOICE AUDIT MANAGEMENT SYSTEM” is based on the results of investigations carried out by me at Neighborly, under the mentorship of Mr. Rohit Kothawade and the same has not been submitted elsewhere for the award of a degree or diploma by me. Further, I understand that Goa University or its authorities will not be responsible for the correctness of observations / experimental or other findings given in the internship report/work. I hereby authorize the University/college authorities to upload this dissertation on the dissertation repository or anywhere else as the UGC regulations demand and make it available to any one as needed.

Sanket Narvekar

Signature and Name of Student

Seat no: 2210

Date: 6th June 2024

Place: Goa University

COMPLETION CERTIFICATE

This is to certify that the internship report "DATA MIGRATION FRAMEWORK & INVOICE AUDIT MANAGEMENT SYSTEM" is a bonafide work carried out by Mr. Sanket Govind Narvekar under my mentorship in partial fulfilment of the requirements for the award of the degree of Master of Computer Applications in the Discipline of Computer Science and Technology at the Goa Business School, Goa University

[illegible] [Rohit kothawade]

Date: 6TH June 2024

Signature and Name of Mentor

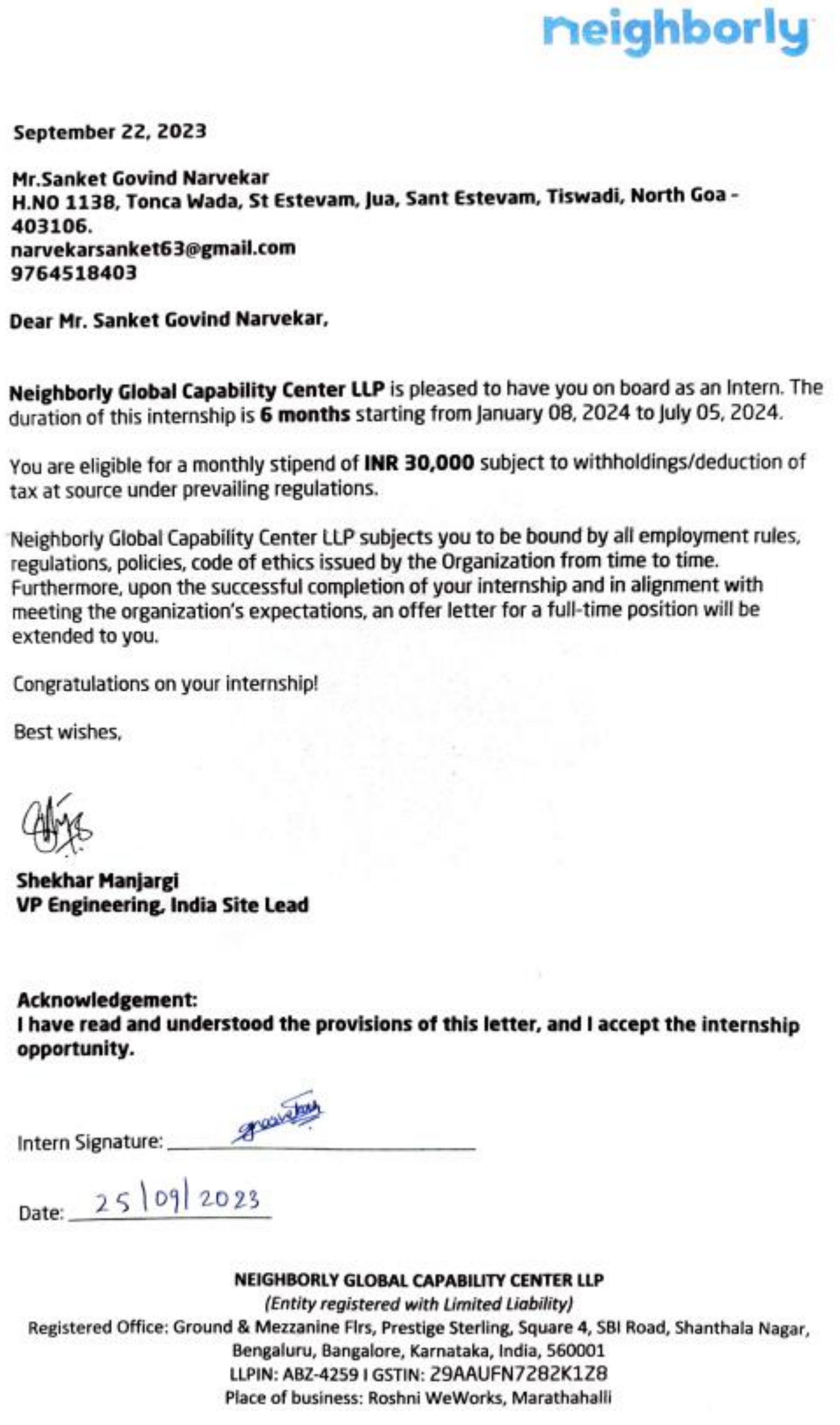
Signature of Dean of School

Date:

Department Stamp

Place: Bangalore

INTERNSHIP OFFER LETTER



neighborly

September 22, 2023

Mr.Sanket Govind Narvekar
H.NO 1138, Tonca Wada, St Estevam, Jua, Sant Estevam, Tiswadi, North Goa - 403106.
narvekarsanket63@gmail.com
9764518403

Dear Mr. Sanket Govind Narvekar,

Neighborly Global Capability Center LLP is pleased to have you on board as an Intern. The duration of this internship is **6 months** starting from January 08, 2024 to July 05, 2024.

You are eligible for a monthly stipend of **INR 30,000** subject to withholdings/deduction of tax at source under prevailing regulations.

Neighborly Global Capability Center LLP subjects you to be bound by all employment rules, regulations, policies, code of ethics issued by the Organization from time to time. Furthermore, upon the successful completion of your internship and in alignment with meeting the organization's expectations, an offer letter for a full-time position will be extended to you.

Congratulations on your internship!

Best wishes,

[illegible]

Shekhar Manjargi
VP Engineering, India Site Lead

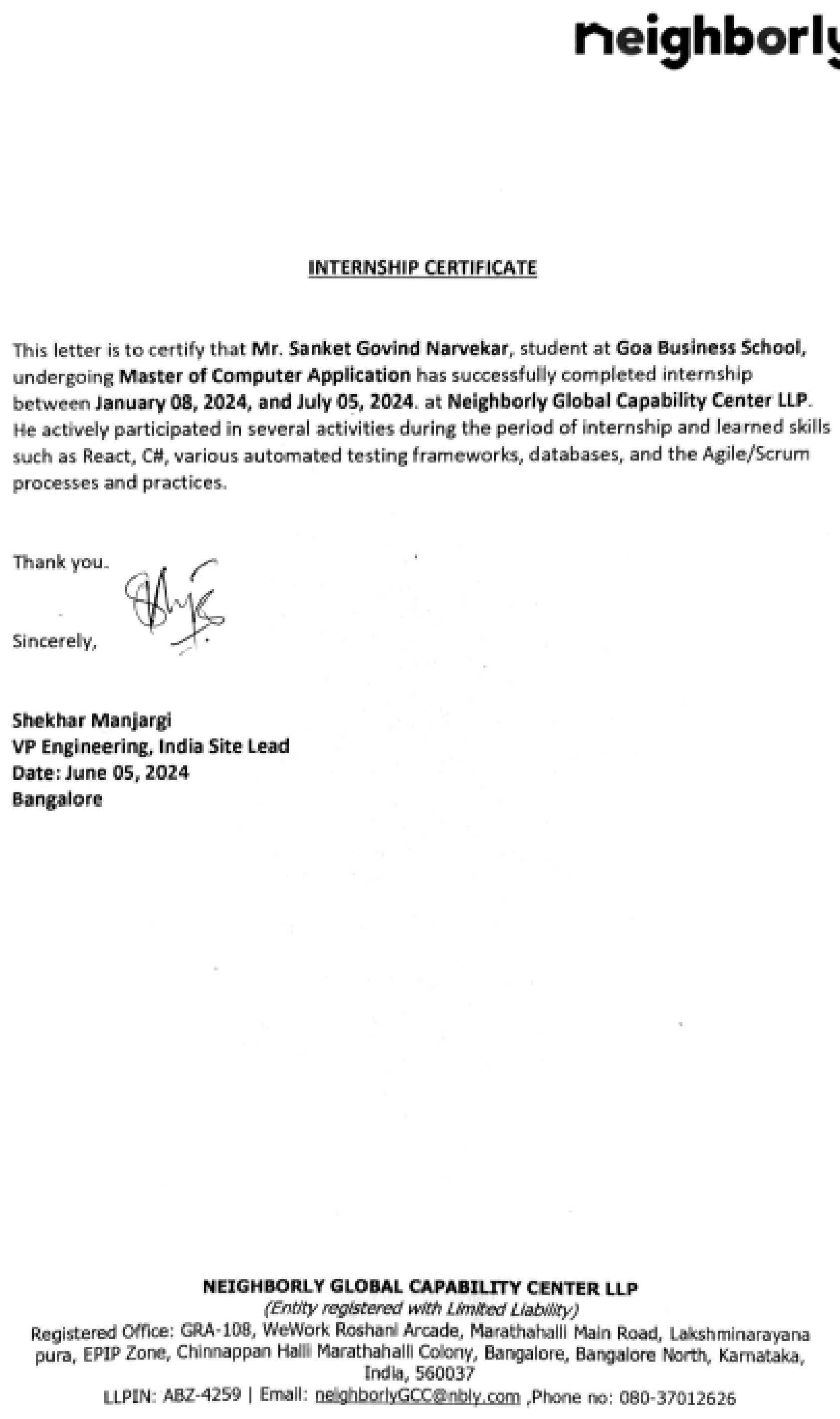
Acknowledgement:
I have read and understood the provisions of this letter, and I accept the internship opportunity.

Intern Signature: [illegible]

Date: 25|09|2023

NEIGHBORLY GLOBAL CAPABILITY CENTER LLP
(Entity registered with Limited Liability)
Registered Office: Ground & Mezzanine Flrs, Prestige Sterling, Square 4, SBI Road, Shanthala Nagar, Bengaluru, Bangalore, Karnataka, India, 560001
LLPIN: ABZ-4259 I GSTIN: 29AAUFN7282K1Z8
Place of business: Roshni WeWorks, Marathahalli

INTERNSHIP CERTIFICATE



neighborly

INTERNSHIP CERTIFICATE

This letter is to certify that **Mr. Sanket Govind Narvekar**, student at **Goa Business School**, undergoing **Master of Computer Application** has successfully completed internship between **January 08, 2024, and July 05, 2024**. at **Neighborly Global Capability Center LLP**. He actively participated in several activities during the period of internship and learned skills such as React, C#, various automated testing frameworks, databases, and the Agile/Scrum processes and practices.

Thank you.

[illegible]

Sincerely,

Shekhar Manjargi
VP Engineering, India Site Lead
Date: June 05, 2024
Bangalore

NEIGHBORLY GLOBAL CAPABILITY CENTER LLP
(Entity registered with Limited Liability)
Registered Office: GRA-108, WeWork Roshani Arcade, Marathahalli Main Road, Lakshminarayana pura, EPIP Zone, Chinnappan Halli Marathahalli Colony, Bangalore, Bangalore North, Karnataka, India, 560037
LLPIN: ABZ-4259 | Email: neighborlyGCC@nbly.com ,Phone no: 080-37012626

Acknowledgements

I extend sincere gratitude to Neighborly for providing me with the opportunity to collaborate with experienced professionals and gain invaluable insights into their fields.

I am thankful to Mr. Shekhar Manjargi, VP and Site Leader India, for fostering a supportive environment for interns and facilitating our transition from academia to industry.

Special thanks to Mr. Pankaj Jain for organizing informative sessions with company leaders and offering consistent guidance throughout the internship. I also appreciate the assistance of Ms. Shruthi Nagraj and Ms. Shruthi P.S. during the onboarding process. I also thank the HR and Finance departments for all the timely support they provided during the internship.

I am grateful to Mr. Rohit Kothawade for his support and leadership, as well as to Ms. Alisha Lotlekar, Mr. Ritesh Singh, and Ms. Apoorva Jadi for their assistance in integrating me into the team.

Thanks are due to Goa University, Goa Business School, and the Discipline of Computer Science and Technology for facilitating this internship opportunity.

I extend my appreciation to Mr. Hanumant Redkar and Mr. Jarret Fernandes for their guidance and support.

Special thanks to Dr. Jyoti Pawar, Mr. Ramrao Wagh, Mr. Ramdas Karmali, Mr. Baskar S., Dr. P. Payaswini, Mrs. Tina Vaz, and Ms. Gaurpriya Chodankar for their mentorship. I also thank the non-teaching staff for their assistance.

Lastly, I am grateful to my friends and family for their unwavering support throughout this journey.

Executive summary

This report documents the internship experience at Neighborly, undertaken as part of the Master of Computer Applications program at Goa University. The internship involved working on various projects and gaining hands-on experience with technologies and processes used in the company.

The first part talks about the company overview. It highlights the different sections in the company. These sections include Digital, Integration Apps, Data & Analytics, Quality Assurance, Systems, and Project. Each section has specific roles:

Digital: Manages customer interaction platforms and aims to unify brand websites.

Integration Apps: Handles data integration from acquired brands into Neighborly's systems.

Data & Analytics: Analyzes data to support business decisions.

Quality Assurance: Ensures product and service quality.

Systems: Focuses on automating software deployment and maintenance.

Project Management: Manages project execution and ensures alignment with goals.

The next section talks about the various tasks handled during the internships.

These tasks included designing and implementing an Invoice Audit Management System to automate the verification of vendor invoices. The system design encompassed system architecture, database design, and user interface development. Key tasks involved were Database Design, API Endpoints, Business Logic, Unit Testing the application.

This section also talks about a New Data Migration Framework to move data efficiently from the data lake to the Onverity Database using a command-line application with multithreading capabilities. This project is the intellectual property of the company and thus the codebase can not be put in the document being published.

The next section highlights the key learnings in the duration of the internship. The internship provided both technical and non-technical learning opportunities. The technical skills gained include knowledge in C#, .NET, Entity Framework Core, PostgreSQL, GIT, POSTMAN, and testing frameworks like xUnit, Karate, Gatling, and

Cypress. The non-technical skills gained include insights into project management, teamwork, and professional communication.

The next section highlights the challenges faced and how they were overcome, such as adapting to the company's tech stack, designing systems to meet specific requirements, and ensuring data integrity during migrations.

In conclusion, the internship at Neighborly was a valuable experience that bridged the gap between academic learning and industry practices. The projects undertaken provided a comprehensive understanding of software development, data management, and quality assurance in a professional setting.

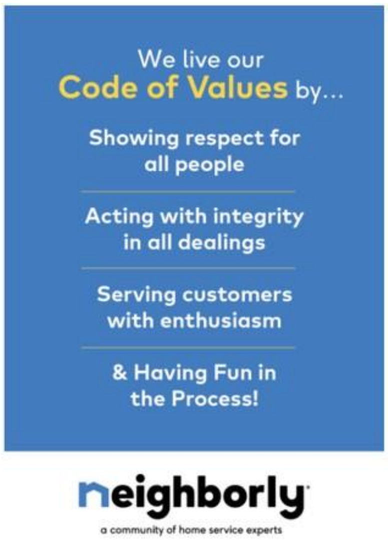
TABLE OF CONTENTS

1. Company Overview

1.1 Bird's Eye View

Neighborly is the world's largest home services company, delivering premium service experiences across homes and businesses, primarily through franchising, headquartered in Waco, Texas. They started over 40 years ago with one brand and have now grown to own 30+ brands and 5500+ franchises across the globe. They have operations in the US, Canada, Mexico, UK, Germany, France, Austria, Portugal, Ireland, Australia, and New Zealand. The company has reported a revenue of $4.1Billion in the year 2023.

Like most companies, Neighborly has an official mission statement and vision; however, unlike other companies, we also have a Code of Values that each employee is urged to follow and know by heart and with heart. The Code of Values serves as a set of universal guidelines by which we strive to adhere, from the bottom of the organization all the way to the top. In fact, any meeting of three or more employees of Neighborly begins with a reciting the Code of Values. (Neighborlybrands website, n.d.)



We live our
Code of Values by...

Showing respect for all people

Acting with integrity in all dealings

Serving customers with enthusiasm

& Having Fun in the Process!

neighborly
a community of home service experts

Figure 1: RICH - Code of Values

The brands under Neighborly are broadly classified into three categories, Repair, Maintain, Enhance.

Repair: Undertakes work that involves repairing. Generally, tends to be nonrepeating work.

Maintain: Undertakes work that involves maintenance and usually are jobs that repeat over time.

Enhance: Services that provide enhancements and like repair, tend to be nonrepeating.

The company has set up a Global Capabilities Center in India to support the company's technology transformation by helping build capabilities that benefit its brands, franchise owners, and customers. Neighborly has already embarked on a digital transformation journey with a refreshed Neighborly.com website and the launch of the Neighborly app in the U.S. The GCC will serve as a strategic facility to access India's digital talent pool to help drive future growth. The GCC will be the driver for modernizing of business capabilities and strengthening of technology foundations of the company for operation excellence and customer satisfaction. The GCC will look to catalyze the company's digital transformation journey by developing innovative enterprise grade E-Commerce, Data Analytics solutions.

The GCC is part of the CTO Organization which looks as shown below:

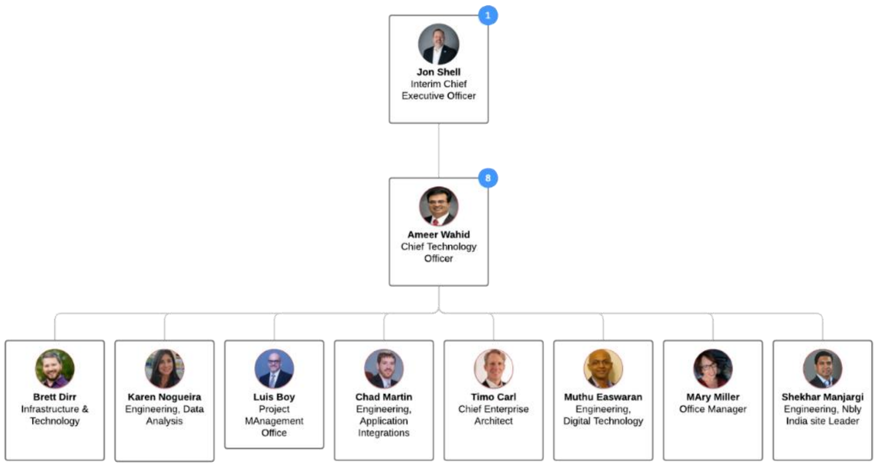

1
Jon Shell
Interim Chief Executive Officer
8
Ameer Wahid
Chief Technology Officer
Brett Dirr
Infrastructure & Technology
Karen Nogueira
Engineering, Data Analysis
Luis Boy
Project MAnagement Office
Chad Martin
Engineering, Application Integrations
Timo Carl
Chief Enterprise Architect
Muthu Easwaran
Engineering, Digital Technology
MAry Miller
Office Manager
Shekhar Manjargi
Engineering, Nbly India site Leader

Figure 2: Neighborly Leadership

Further, the team at the Neighborly India GCC looks like follows:

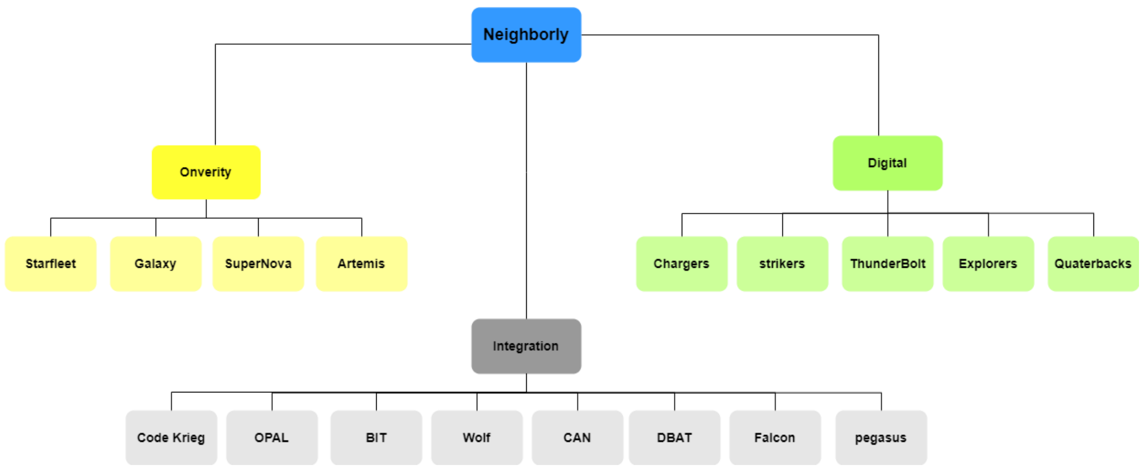


Figure 3: Neighborly GCC Teams

1.2 Products

1.2.1 Onverity

Neighborly has acquired 30+ brands. Some of these brands use some third-party Field Service Management (FSM) apps and Point of Sale (POS) apps. The brands need to pay to avail these services, and secondly, the brand does not own user data in the strictest sense of the word. Neighborly, therefore came up with a Field Service Management / Point of sale system of their own.

Onverity is a field service management system that manages companies' field operations: scheduling, dispatching, workorder management including leads, estimates, and invoices. This application is used by various users like neighborly admin, concept admin, business unit admins, business unit operator. These users are responsible for the creation of leads, estimates, workorder, managing various setting for the brands, adding invoicing details and other brand specific needs. Onverity also acts as a point of sale, allowing the brands to generate estimates and invoices, and execute the payment for the generated invoices and where sales taxes may become payable.

When the brands decide to use Onverity as their FSM/POS a few positive changes happen for Neighborly from a business perspective. The fees that were being paid for third party software now come to neighborly, so another revenue stream is available apart from royalties. Next, the brand since is a child entity of neighborly, owns all the user data, be it customer data, workorders, estimates, leads etc. This large amount of data allows the company to run effective data analytics to improve on business metrics and gain insights into key factors that can help improve the functional efficiency of the business.

Currently, Onverity have onboarded three brands: Dryer Vent Wizards, House Master Services, and Five Star Painting (in User Acceptance Testing). Next brand in line to be onboarded is Mosquito Joe.

There are multiple scrum teams for Onverity which take care of various facets of the application.

- Starfleet – Leads, estimates, workorder, dispatch
- Galaxy – Customer, configuration, notification, synchronization, product catalogs, analytics
- Artemis – invoice, billing
- Supernova – field application

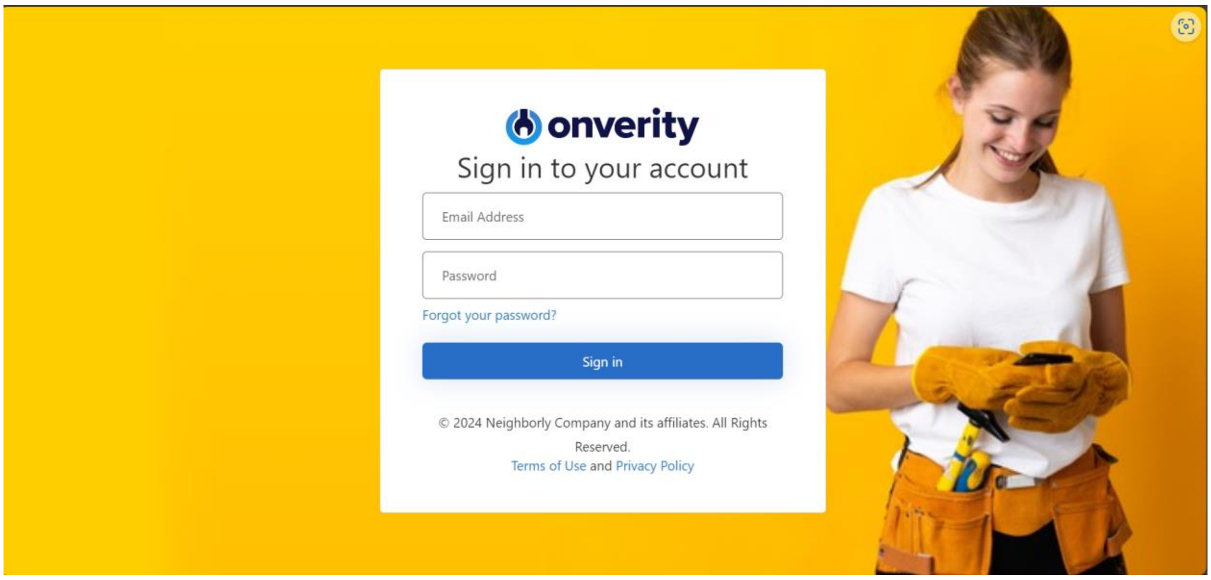

onverity
Sign in to your account
Email Address
Password
Forgot your password?
Sign in
© 2024 Neighborly Company and its affiliates. All Rights Reserved.
Terms of Use and Privacy Policy

Figure 4: Onverity Web App Login Page

1.3 Sections in the Organization

At a larger level, the operations of the GCC are classified into four main verticals: Onverity, Digital, and Integration Apps, and Data & Analytics. Apart from these, they also have a Systems, Quality Assurance, and Project Management teams, which work in collaboration with all other teams to provide support and enhance the functions of other teams.

1.3.1 Digital

Digital team manages all the platforms that interact with our customers, homeowners that request for home services provided by franchise.

Neighborly has several brands that have been acquired and operate under its umbrella. Each brand has different website which has its own distinctive look and feel. Digital aims at migrating all brand websites onto a unified platform while providing a consistent, personalized experience for Neighborly's web and mobile needs.

Migrating all brands to single platform will help better management of brand customers. while ensuring consistent look and feel across all the websites will help customers identify brands belonging to neighborly and transition them to multi brand customers.

Summary of the Websites Involved:

Neighborly Corporate Website (www.neighborly.com): Primary website for brand information, customer engagement, and service offerings.

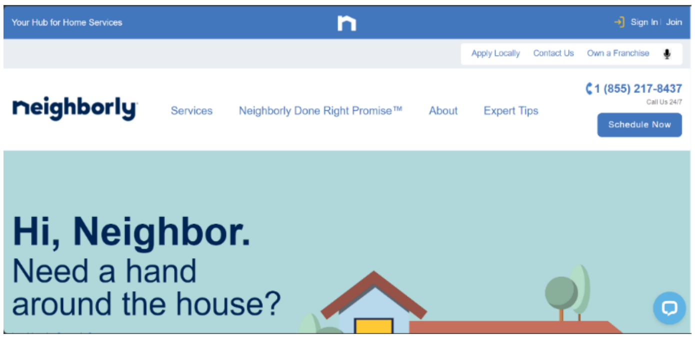


Figure 5: Neighborly Corporate Website

ProTradeNet (www.protradenet.com): B2B platform connecting businesses and vendors with a focus on secure transactions and vendor management.

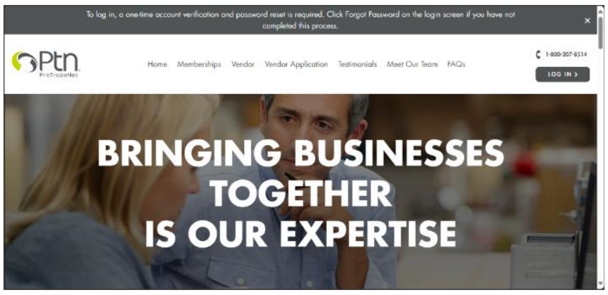

To log in, a one-time account verification and password reset is required. Click Forgot Password on the login screen if you have not completed this process.
Ptn
Home
Memberships
Vendor
Vendor Application
Testimonials
Meet Our Team
FAQs
1-800-207-8514
LOG IN
BRINGING BUSINESSES TOGETHER IS OUR EXPERTISE

Figure 6: ProTradeNet Website

Franchisee Onboarding (https://franchise.neighborly.com): Portal for new franchisee onboarding, including applications, resources, and training.

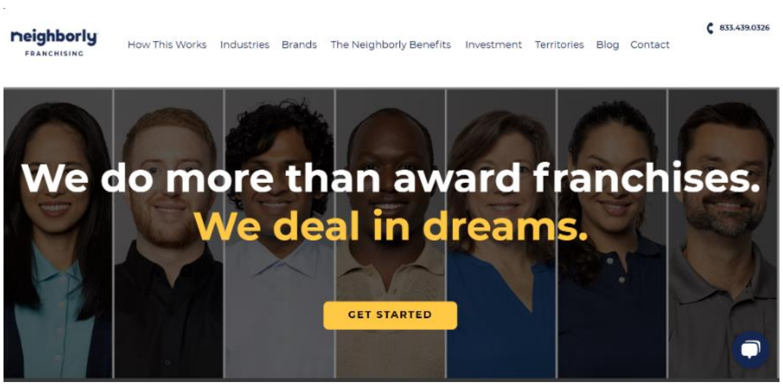


Figure 7: Neighborly Franchise Website

Neighborly Mobile App: Mobile application providing similar functionalities to the corporate website with additional mobile-specific features like push notifications and location services.

1.3.2 <u>Integration Apps</u>

The Integration Apps is the core function that deals with integrating brands that have been acquired by Neighborly by moving all the necessary data into the data lake and all other systems wherever required. They work closely with the Brand Owners and other stakeholders to learn the how the brand operates, what their needs are and then translate them into technical requirements, to be fulfilled by the respective team.

BIT interact with representatives of the brands to understand their processes, the kind of data they have and the profile of their customers, the kind of loads they handle generally etc. The next step is to get the brand to fill a contract management template. The filled template is verified by BIT and if any discrepancies are found, the brand is then made to rectify them. This template is then fed into a database called FranConnect. The data is then copied from the FranConnect into FranForce database which is the primary database for Brand Data such as Customers, Workorder, invoices, payments and such.

From FranForce, this data must be migrated to the Onverity Database using APIs by transforming it as required by the API payload structure. This data is then verified for correctness and integrity and if there are any lapses, they are corrected before the data goes into the production database.

The teams associated with this function are critical to business of the company as they deal with many of the data operations and make it readily available for other functions to use without which, their functioning would be much harder.

1.3.3 <u>Data & Analytics</u>

The Data & Analytics team works with the rest of the organization to help identify key metrics based on all the data available in the Data Lake and suggest improvements. Their analysis also helps the sales and marketing teams to make business decisions backed by data.

1.3.4 <u>Quality Assurance</u>

The quality assurance team is of absolute importance as they make sure that all products and services that are being delivered are fully functional and have no bugs, as bugs in production lead to loss in reputation and revenue. They also look out for continuous improvements in the processes and set up benchmarks for quality across the organization.

1.3.5 Systems

The DevOps engineers (systems team) focus on automating and streamlining the processes involved in deploying, managing, and maintaining software applications. They use tools and practices such as continuous integration, continuous delivery, and infrastructure as code to improve efficiency and reliability and make sure SRE practices are in place for all the applications.

1.3.6 Project Management

The project management team is responsible for planning, organizing, executing, and overseeing all aspects of a project from initiation to completion. Their primary role involves defining project objectives, creating schedules, allocating resources, managing budgets, and ensuring that project deliverables are met within scope, time, and budget constraints. They coordinate communication between stakeholders, facilitate collaboration among team members, identify and mitigate risks, and resolve issues that may arise during the project lifecycle. Additionally, the project management team monitors progress, tracks project metrics, and provides regular updates to stakeholders to ensure transparency and alignment with project goals. Ultimately, their goal is to successfully deliver the project on time, within budget, and to the satisfaction of stakeholders.

2. Task(s) Handled

As a part of the internship, I first had to learn some basics of the tech stack being used in the company. This mainly consisted of C# .NET, Entity Framework Core, Unit Testing, Design Principles and Design Patterns. After I completed the initial study, I was assigned to design and implement an Invoice Audit Management System.

2.1 Invoice Audit Management System (Warm up project)

Invoice Audit Management System is an internal tool that was developed to help the scrum masters keep track of various things. This was done to automate the task of verifying the number of hours invoiced by a vendor and finding any discrepancies in the invoice. This was also meant as a warmup project for the interns to get used to the tech stack and processes being followed in the company. This meant, we had to adhere to standards and practices that are followed in the company so that we get a hang of things.

The main Aim behind this project was to ease the work of the scrum master. The organization consists of several internal teams along with that it also has some vendor teams that work on company's internal projects. Some members work duration is based on project based, some on month based. At the end of the month the vendor teams send their invoice to neighborly. Now the scrum master's job is to check the invoice and billable hours and make sure that it tallies with the internal data. Manually doing this is a very tedious and time-consuming task and cuts into productive time for the person undertaking the task. It also increases the chance of errors due to manual work being done by people.

To get started, my team and I started off the system design. This included the system architecture, database design and the user interface mock.

The frontend and the backend were developed separately as microservices. I specifically accepted the tasks related to the backend so that I could learn more about backend functionality and how it works. The backend dealt with database connection, creating tables in the database, implementing design patterns.

2.1.1 Database design

To create the database models, I used the code first approach. In this approach, the models for the database tables are created in the code. Then, using an ORM (Entity Framework in our case), the tables are created in the database based on the models created in the code. The tables and fields we captured in the database looked something like this:

i. Vendors
ii. Location
iii. Project that they are working.
iv. Vendor holidays
v. Daily billable hours per team member.
vi. Team members of a vendor team.
vii. Whether the team member is supposed to be billed or not
viii. The leaves they have availed.

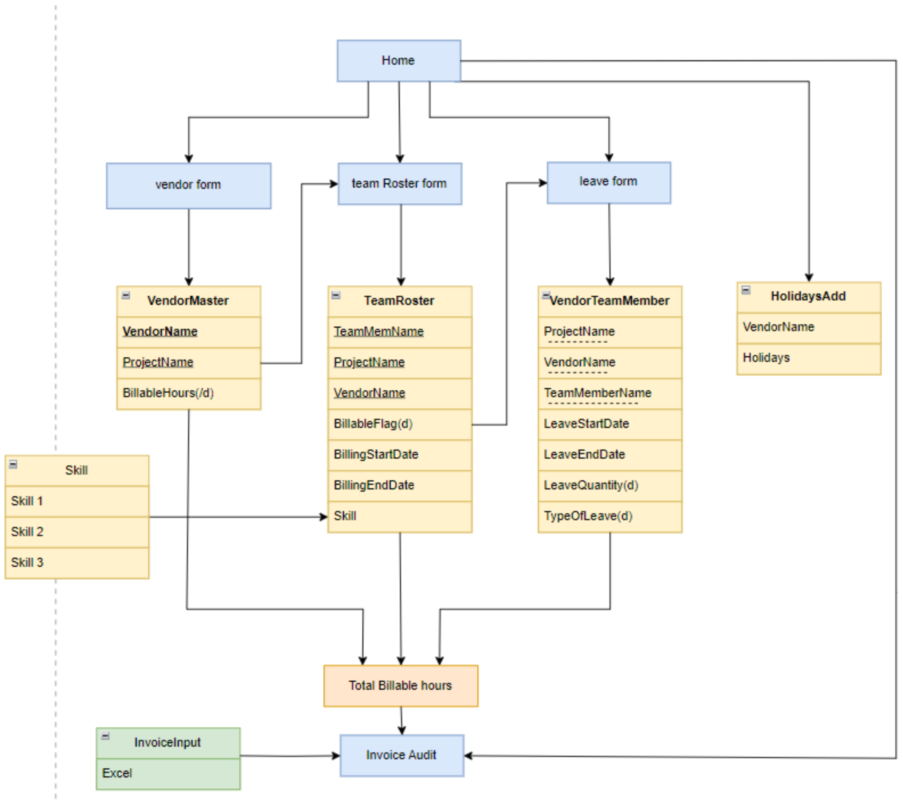


Figure 8: Database Design and Application Flow Diagram

Lastly, we also created a table to capture the discrepancies thrown by the system and used the data from there to be displayed in the frontend.

2.1.2 API Endpoints

Next thing I had to do was create API endpoints for the frontend to use. To do this, I used the Model View Controller Architecture to handle incoming requests. Here, I only needed to use the models and controllers since the view was being handled by the frontend microservice. In this task, I also had to implement repository service layer pattern. In this, the code for database access is written in the repository classes and the business logic is written in the service layer classes. This was a good learning experience as it allowed me to venture into specific implementation of design patterns. This also allowed me to assign single responsibility to classes and keep the codebase clean.

2.1.3 Business Logic

Lastly, the most important part of this was the business logic required for the application to run and perform its intended task. In this, I had to parse necessary inputs from an incoming excel file and then map them to specific models that mimicked the database table schema. This data could then be saved to the database. Next, I had fetch data from the database for specific vendor and project for a specific month, format it, and calculate the expected billable hours for each team member for that vendor, project and month combination. After calculating this, I had to parse the vendor invoice file and compare the calculated data to the invoice file data and check and report if there were any discrepancies and return them as an API response.

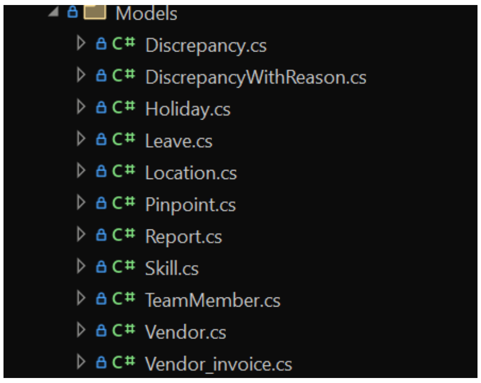
2.1.4 Authorization & Authentication

Upon the direction of the product owner, I had tried to implement authentication and authorization for this module, but this requirement was later scraped. In this, I implemented JWT based authorization and authentication for login.

2.1.5 Unit Testing

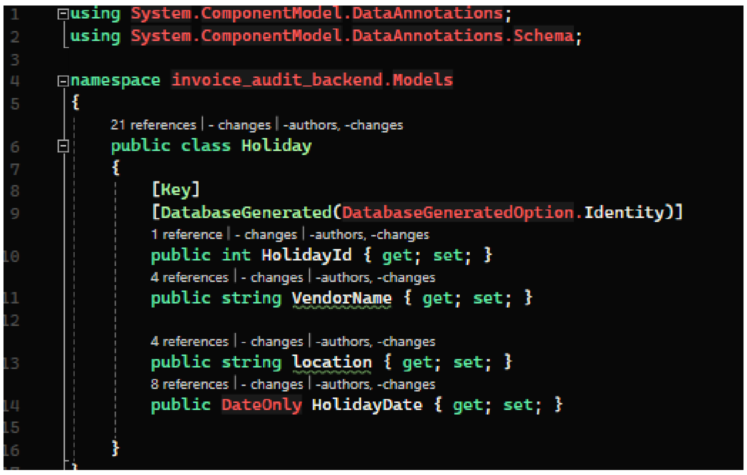
I wrote unit tests to cover for various scenarios and to ensure that I detect any major errors in the code. I used the xUnit testing framework for doing so.

Here are some screenshots from the frontend and the backend code that we created.



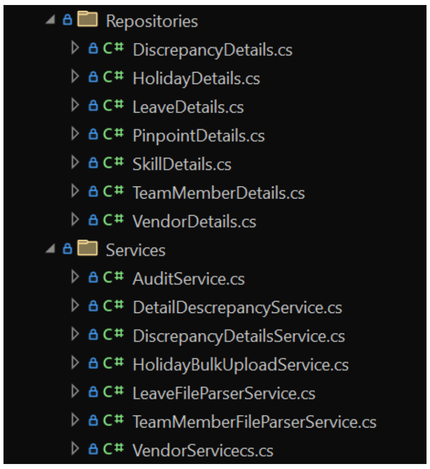
```
Models
  Discrepancy.cs
  DiscrepancyWithReason.cs
  Holiday.cs
  Leave.cs
  Location.cs
  Pinpoint.cs
  Report.cs
  Skill.cs
  TeamMember.cs
  Vendor.cs
  Vendor_invoice.cs
```

Figure 9: Snapshot of Models



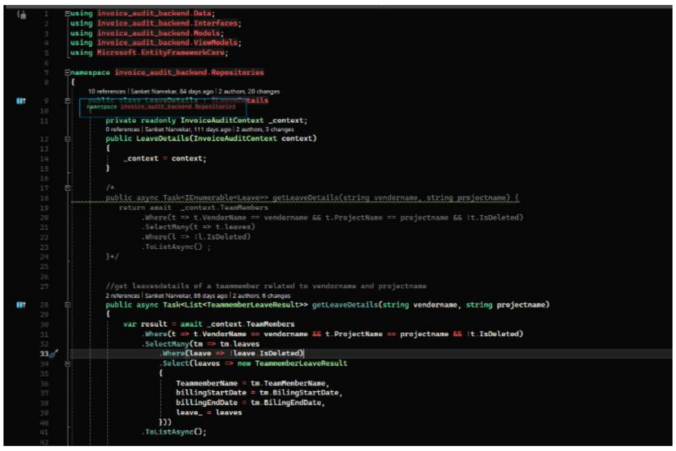
```csharp
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace invoice_audit_backend.Models
{
    21 references | - changes | -authors, -changes
    public class Holiday
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        1 reference | - changes | -authors, -changes
        public int HolidayId { get; set; }
        4 references | - changes | -authors, -changes
        public string VendorName { get; set; }

        4 references | - changes | -authors, -changes
        public string location { get; set; }
        8 references | - changes | -authors, -changes
        public DateOnly HolidayDate { get; set; }

    }
```

Figure 10: Snapshot of a model class



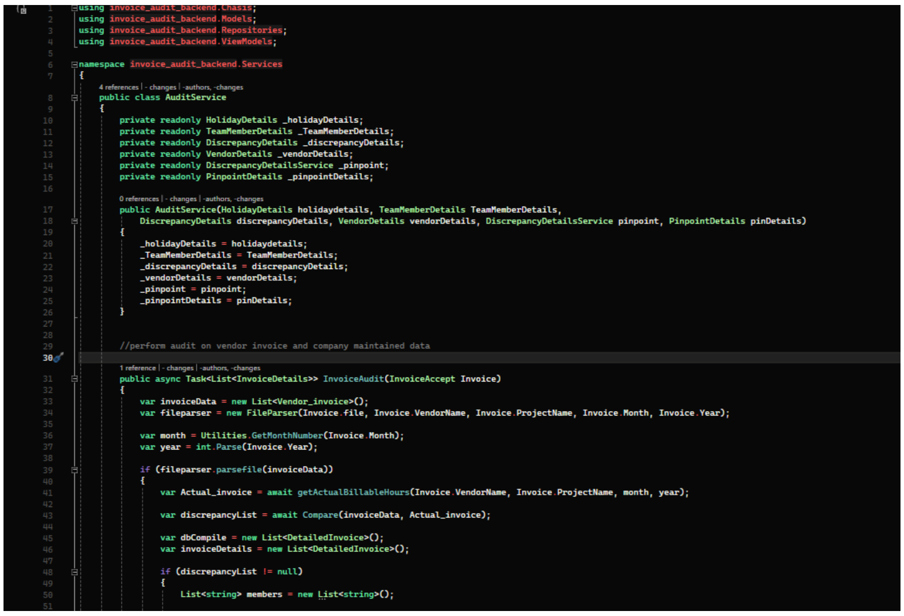
```
Repositories
  DiscrepancyDetails.cs
  HolidayDetails.cs
  LeaveDetails.cs
  PinpointDetails.cs
  SkillDetails.cs
  TeamMemberDetails.cs
  VendorDetails.cs
Services
  AuditService.cs
  DetailDescrepancyService.cs
  DiscrepancyDetailsService.cs
  HolidayBulkUploadService.cs
  LeaveFileParserService.cs
  TeamMemberFileParserService.cs
  VendorServicecs.cs
```

Figure 11: Snapshot showing the Repository Service Layer Pattern



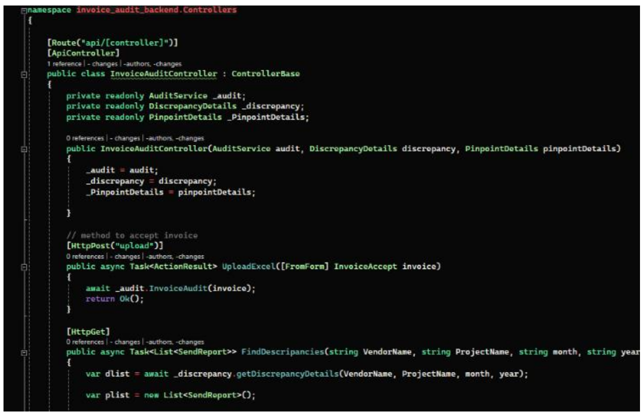
```
using invoice_audit_backend.Data;
using invoice_audit_backend.Interfaces;
using invoice_audit_backend.Models;
using invoice_audit_backend.ViewModels;
using Microsoft.EntityFrameworkCore;

namespace invoice_audit_backend.Repositories
{
    10 references | Sanket Narvekar, 84 days ago | 2 authors, 20 changes
    public class LeaveDetails : ILeaveDetails
    namespace invoice_audit_backend.Repositories
        private readonly InvoiceAuditContext _context;
        0 references | Sanket Narvekar, 111 days ago | 2 authors, 3 changes
        public LeaveDetails(InvoiceAuditContext context)
        {
            _context = context;
        }

        /*
        public async Task<IEnumerable<Leave>> getLeaveDetails(string vendorname, string projectname) {
           return await  _context.TeamMembers
                 .Where(t => t.VendorName == vendorname && t.ProjectName == projectname && !t.IsDeleted)
                 .SelectMany(t => t.leaves)
                 .Where(l => !l.IsDeleted)
                 .ToListAsync() ;
        }*/


        //get leavesdetails of a teammember related to vendorname and projectname
        2 references | Sanket Narvekar, 86 days ago | 2 authors, 6 changes
        public async Task<List<TeammemberLeaveResult>> getLeaveDetails(string vendorname, string projectname)
        {
            var result = await _context.TeamMembers
                .Where(t => t.VendorName == vendorname && t.ProjectName == projectname && !t.IsDeleted)
                .SelectMany(tm => tm.leaves
                    .Where(leave => !leave.IsDeleted)
                    .Select(leaves => new TeammemberLeaveResult
                    {
                        TeammemberName = tm.TeamMemberName,
                        billingStartDate = tm.BilingStartDate,
                        billingEndDate = tm.BilingEndDate,
                        leave_ = leaves
                    }))
                .ToListAsync();
```

Figure 12: Snapshot of the repository class with database access



```
using invoice_audit_backend.Chasis;
using invoice_audit_backend.Models;
using invoice_audit_backend.Repositories;
using invoice_audit_backend.ViewModels;

namespace invoice_audit_backend.Services
{
    4 references | - changes | -authors, -changes
    public class AuditService
    {
        private readonly HolidayDetails _holidayDetails;
        private readonly TeamMemberDetails _TeamMemberDetails;
        private readonly DiscrepancyDetails _discrepancyDetails;
        private readonly VendorDetails _vendorDetails;
        private readonly DiscrepancyDetailsService _pinpoint;
        private readonly PinpointDetails _pinpointDetails;

        0 references | - changes | -authors, -changes
        public AuditService(HolidayDetails holidaydetails, TeamMemberDetails TeamMemberDetails,
            DiscrepancyDetails discrepancyDetails, VendorDetails vendorDetails, DiscrepancyDetailsService pinpoint, PinpointDetails pinDetails)
        {
            _holidayDetails = holidaydetails;
            _TeamMemberDetails = TeamMemberDetails;
            _discrepancyDetails = discrepancyDetails;
            _vendorDetails = vendorDetails;
            _pinpoint = pinpoint;
            _pinpointDetails = pinDetails;
        }


        //perform audit on vendor invoice and company maintained data

        1 reference | - changes | -authors, -changes
        public async Task<List<InvoiceDetails>> InvoiceAudit(InvoiceAccept Invoice)
        {
            var invoiceData = new List<Vendor_invoice>();
            var fileparser = new FileParser(Invoice.file, Invoice.VendorName, Invoice.ProjectName, Invoice.Month, Invoice.Year);

            var month = Utilities.GetMonthNumber(Invoice.Month);
            var year = int.Parse(Invoice.Year);

            if (fileparser.parsefile(invoiceData))
            {
                var Actual_invoice = await getActualBillableHours(Invoice.VendorName, Invoice.ProjectName, month, year);

                var discrepancyList = await Compare(invoiceData, Actual_invoice);

                var dbCompile = new List<DetailedInvoice>();
                var invoiceDetails = new List<DetailedInvoice>();

                if (discrepancyList != null)
                {
                    List<string> members = new List<string>();
```

Figure 13: Snapshot of the service class with business logic



```
namespace invoice_audit_backend.Controllers
{

    [Route("api/[controller]")]
    [ApiController]
    1 reference | - changes | -authors, -changes
    public class InvoiceAuditController : ControllerBase
    {
        private readonly AuditService _audit;
        private readonly DiscrepancyDetails _discrepancy;
        private readonly PinpointDetails _PinpointDetails;

        0 references | - changes | -authors, -changes
        public InvoiceAuditController(AuditService audit, DiscrepancyDetails discrepancy, PinpointDetails pinpointDetails)
        {
            _audit = audit;
            _discrepancy = discrepancy;
            _PinpointDetails = pinpointDetails;

        }

        // method to accept invoice
        [HttpPost("upload")]
        0 references | - changes | -authors, -changes
        public async Task<ActionResult> UploadExcel([FromForm] InvoiceAccept invoice)
        {
            await _audit.InvoiceAudit(invoice);
            return Ok();
        }

        [HttpGet]
        0 references | - changes | -authors, -changes
        public async Task<List<SendReport>> FindDescripancies(string VendorName, string ProjectName, string month, string year
        {
            var dlist = await _discrepancy.getDiscrepancyDetails(VendorName, ProjectName, month, year);

            var plist = new List<SendReport>();
```

Figure 14: Snapshot of the controller class

This warmup project started on the 17th of January 2024, and it took three sprints of two weeks each to complete. It ended on the 13th of March 2024 and has hence been tested. The deployment is pending due to the CE and systems team being busy with other important deployments.

At the end of this project, I had a working knowledge of C# .NET, Entity Framework Core, PostgreSQL, GIT, POSTMAN, xUnit, Karate Gatling and Cypress. This gave me the confidence to take on work for live projects after being assigned to a team.

2.2 New Data Migration Framework

After the warmup project was completed, we were divided and assigned to different teams. I was assigned to the Brand Integration team under the Integration Apps function. This team is involved in migration of data from the data lake to the Onverity Database by transforming it in the necessary format.

The status quo is that the team has developed a command line application that takes certain parameters to migrate the necessary data for a given franchise and the specific business unit. The code uses semaphores to implement multithreading so that large amounts of data can be moved relatively fast. The application is run manually on demand whenever there is any migration needs to be done. It takes parameters like the license number, sub license number, source type id etc. It then fetches the franchise based on these data points. Then from these franchises, we further select the specific business units. After the business unit is found, we migrate the customers and related tables for this business unit and then migrate the workorders, leads, estimates, invoices etc., using the Onverity APIs.

The need for a new data migration framework arises because of a few hiccups that we face in this approach.

- Firstly, using semaphores is slow since it is does not account for the size of the batch being processed by each thread. For example, if there are 2 threads, T1 and T2. If T1 is processing 10 data items and T2 is processing 1000 data items. Then T1 will finish first, but it will have to wait for T2 to finish. This leads to a loss in efficiency making the application slower.
- It was noticed that the initial migration for the pilot business unit takes 30-45 days of time due to various factors like unmigrated data and errors in migrations.
- The developers must run this application on a local machine manually to migrate data and that becomes tedious. There is no mechanism for graceful recovery in case of any issues like power or internet failure.

- The application also takes a lot of manual intervention by the developers.

So, the solution proposed was that all this data from Franforce needs to be transformed beforehand and stored in a staging database from where it can be directly fed into Onverity APIs. It was also proposed to make use of Reactive Extensions for .NET which allows us to use observables for multithreading and spares us the efficiency problem we face when using semaphores.

The new solution also allows us to read the status of migration in real time with the help of a request id which is associated with an ongoing migration and shows the status of migration.

2.2.1 Data Transformation Objects

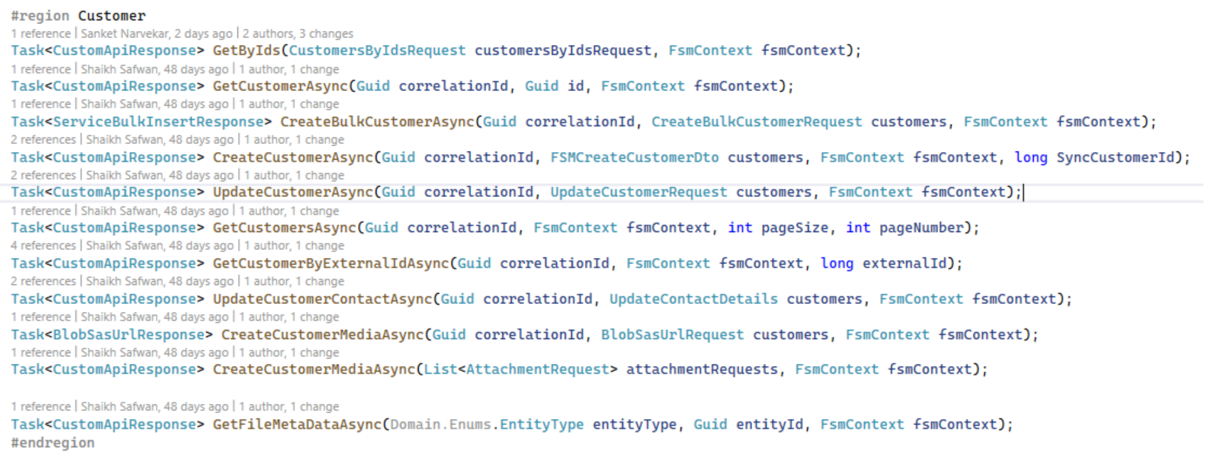
This application deals with transfer of data from one database to another. This means that the schema of the databases may not necessarily be the same and thus to make sure that the data moves from one database to another, I had to create data transformation objects that could map the data to both the databases.

2.2.2 Reactive Extensions for .NET

To allow multithreading and fast transfer of data from the database, it was decided to implement observables and observers so that the data could be streamed from the database over different threads and that way, the job would be done faster. Using observables, it is expected that the time for migration will be reduced.

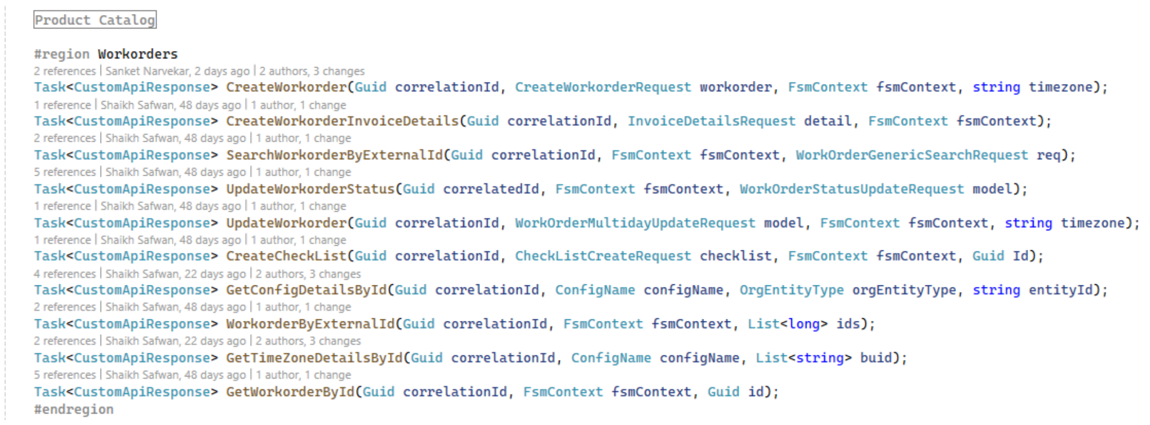
2.2.3 Command Query Responsibility Segregation

The CQRS design patter was implemented to make sure that the method calls to get and save data to the database are implemented separately allowing us to write cleaner code and keep it maintainable.



```
#region Customer
1 reference | Sanket Narvekar, 2 days ago | 2 authors, 3 changes
Task<CustomApiResponse> GetByIds(CustomersByIdsRequest customersByIdsRequest, FsmContext fsmContext);
1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> GetCustomerAsync(Guid correlationId, Guid id, FsmContext fsmContext);
1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<ServiceBulkInsertResponse> CreateBulkCustomerAsync(Guid correlationId, CreateBulkCustomerRequest customers, FsmContext fsmContext);
2 references | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> CreateCustomerAsync(Guid correlationId, FSMCreateCustomerDto customers, FsmContext fsmContext, long SyncCustomerId);
2 references | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> UpdateCustomerAsync(Guid correlationId, UpdateCustomerRequest customers, FsmContext fsmContext);
1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> GetCustomersAsync(Guid correlationId, FsmContext fsmContext, int pageSize, int pageNumber);
4 references | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> GetCustomerByExternalIdAsync(Guid correlationId, FsmContext fsmContext, long externalId);
2 references | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> UpdateCustomerContactAsync(Guid correlationId, UpdateContactDetails customers, FsmContext fsmContext);
1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<BlobSasUrlResponse> CreateCustomerMediaAsync(Guid correlationId, BlobSasUrlRequest customers, FsmContext fsmContext);
1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> CreateCustomerMediaAsync(List<AttachmentRequest> attachmentRequests, FsmContext fsmContext);

1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> GetFileMetaDataAsync(Domain.Enums.EntityType entityType, Guid entityId, FsmContext fsmContext);
#endregion
```

Figure 15: Snapshot of Customer Migration Methods in the Sync Interface



```
Product Catalog

#region Workorders
2 references | Sanket Narvekar, 2 days ago | 2 authors, 3 changes
Task<CustomApiResponse> CreateWorkorder(Guid correlationId, CreateWorkorderRequest workorder, FsmContext fsmContext, string timezone);
1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> CreateWorkorderInvoiceDetails(Guid correlationId, InvoiceDetailsRequest detail, FsmContext fsmContext);
2 references | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> SearchWorkorderByExternalId(Guid correlationId, FsmContext fsmContext, WorkOrderGenericSearchRequest req);
5 references | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> UpdateWorkorderStatus(Guid correlatedId, FsmContext fsmContext, WorkOrderStatusUpdateRequest model);
1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> UpdateWorkorder(Guid correlationId, WorkOrderMultidayUpdateRequest model, FsmContext fsmContext, string timezone);
1 reference | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> CreateCheckList(Guid correlationId, CheckListCreateRequest checklist, FsmContext fsmContext, Guid Id);
4 references | Shaikh Safwan, 22 days ago | 2 authors, 3 changes
Task<CustomApiResponse> GetConfigDetailsById(Guid correlationId, ConfigName configName, OrgEntityType orgEntityType, string entityId);
2 references | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> WorkorderByExternalId(Guid correlationId, FsmContext fsmContext, List<long> ids);
2 references | Shaikh Safwan, 22 days ago | 2 authors, 3 changes
Task<CustomApiResponse> GetTimeZoneDetailsById(Guid correlationId, ConfigName configName, List<string> buid);
5 references | Shaikh Safwan, 48 days ago | 1 author, 1 change
Task<CustomApiResponse> GetWorkorderById(Guid correlationId, FsmContext fsmContext, Guid id);
#endregion
```

Figure 16: Snapshot of Workorder Migration Methods in the Sync Interface

3. Learnings

3.1 Technical Learning

3.1.1 Programming Languages:

C#: C# is a programming language developed by Microsoft. It is widely used for developing various types of applications, including web, desktop, mobile, and gaming applications. C# is known for its simplicity, type-safety, and object-oriented programming features.

3.1.2 Frameworks

Dotnet: .NET is a software framework developed by Microsoft. It provides a comprehensive set of libraries and tools for building and running applications on various platforms, including Windows, macOS, and Linux. .NET supports multiple programming languages, including C#, F#, and Visual Basic.

React.js: React.js, or React, is a JavaScript library for building user interfaces, primarily for single-page applications. Developed by Facebook, React simplifies UI development with its component-based architecture, where UI elements are reusable and encapsulated. It uses a virtual DOM for efficient rendering and offers a declarative syntax for clearer code. React is widely adopted for its performance, simplicity, and scalability.

xUnit Framework: xUnit is an open-source unit testing framework that provides a set of tools for writing and executing automated tests in various programming languages, including C#, Java, and Python. xUnit follows a simple and extensible architecture, with the core principles of simplicity, consistency, and extensibility. It offers features such as test fixtures, assertions, parameterized tests, and test runners, enabling developers to write concise and readable tests for their code. By promoting automated testing practices, xUnit frameworks help developers improve code quality, identify defects early, and ensure the correctness and reliability of their software applications.

Cypress: Cypress is an open-source end-to-end testing framework designed for modern web applications. It allows developers to write, run, and debug tests directly within the browser, providing a seamless testing experience. Cypress offers features such as automatic waiting, real-time reloads, and built-in assertions, making it easy to write and maintain tests. Its robust API enables comprehensive testing of web applications, including interacting with DOM elements, making HTTP requests, and mocking server responses. Cypress also provides powerful debugging tools and detailed test reports, helping developers quickly identify and fix issues.

3.1.3 IDE, Database & API tools

Visual Studio Code: Visual Studio Code is a lightweight, open-source code editor developed by Microsoft. It supports various programming languages and offers features like syntax highlighting, code completion, debugging, and Git integration.

Visual Studio: Visual Studio is an integrated development environment (IDE) developed by Microsoft. It provides comprehensive tools for building various types of applications, including web, desktop, mobile, and cloud-based applications. It offers features like code editing, debugging, testing, and collaboration tools.

PostgreSQL: PostgreSQL is an advanced open-source relational database management system known for its robust features, extensibility, and standards compliance. Offering a wide range of features including SQL compliance, JSON support, full-text search, and advanced indexing, PostgreSQL is highly regarded for its reliability, performance, and scalability. With a strong focus on data integrity and extensibility, PostgreSQL supports various data types, procedural languages, and custom extensions. It provides comprehensive administration tools and security features, making it suitable for a wide range of applications from small projects to large enterprise systems.

PG Admin: pgAdmin is an open-source administration and development platform for PostgreSQL databases. It provides a graphical user interface (GUI) that allows users to perform various database administration tasks, such as managing databases, tables, views, indexes, and users, without needing to use the command-line interface. pgAdmin offers features like SQL query tool, graphical query builder, database object browser, and server status monitoring. It supports multiple server connections and allows users to manage PostgreSQL instances from a centralized interface.

MS SQL Server: Microsoft SQL Server (MSSQL) is a robust relational database management system developed by Microsoft, offered in various versions and editions tailored to diverse organizational needs. Featuring a rich set of capabilities including Transact-SQL (T-SQL) for querying and programming, stored procedures, triggers, views, and advanced features like Reporting Services, Integration Services, and Analysis Services, MSSQL enables efficient data management and analysis. It boasts comprehensive administration tools such as SQL Server

Management Studio (SSMS) and robust security features encompassing authentication, authorization, and encryption. It seamlessly integrates with other Microsoft products and services.

SQL Server Management Studio (SSMS): SSMS is a graphical integrated development environment (IDE) provided by Microsoft for managing SQL Server databases. It offers a wide range of tools and features to help database administrators, developers, and analysts interact with SQL Server instances and databases. SSMS allows users to perform tasks such as creating, modifying, and deleting databases, tables, views, stored procedures, and other database objects through a user-friendly graphical interface. It also provides tools for writing and executing SQL queries, debugging T-SQL code, and analyzing query performance. Additionally, SSMS includes features for database backup and restore, security management, and server configuration.

3.1.4 Other Tools

Entity Framework Core: Entity Framework Core (EF Core) is an open-source object-relational mapping (ORM) framework developed by Microsoft. It enables developers to work with data in a database using .NET objects and provides a high-level abstraction for database interactions, allowing developers to focus more on application logic rather than dealing with low-level database operations. EF Core supports various database providers including SQL Server, MySQL, PostgreSQL, SQLite, and others, offering flexibility in choosing the underlying database technology.

LINQ: Language Integrated Query (LINQ) is a feature in .NET programming languages, such as C# and VB.NET, that provides a unified way to query and manipulate data from various sources, including collections, databases, XML, and more, directly within the language syntax. LINQ allows developers to write queries using familiar language constructs, such as lambda expressions and query expressions, making it easier to express complex data operations in a readable and concise manner. It supports both querying (retrieving data) and transformation (modifying data) operations, providing a flexible and powerful toolset for data manipulation.

Swagger: Swagger, now known as OpenAPI Specification (OAS), is a widely adopted standard for defining and documenting APIs. It provides a machine-readable format for describing RESTful APIs, including details such as available endpoints, request/response formats, parameters, authentication methods, and more. Swagger allows developers to generate interactive documentation, client SDKs, and server stubs automatically from the API specification, improving collaboration between development teams and facilitating API consumption by third-party developers.

Reactive Extensions for .NET: Reactive Extensions (Rx) for .NET is a library that provides a set of powerful tools for composing asynchronous and event-based programs using observable sequences. Inspired by functional reactive programming (FRP), Rx allows developers to work with asynchronous data streams using a unified API across various platforms and programming languages. With Rx, developers can easily create, manipulate, and subscribe to observable sequences, which represent streams of data or events over time. Rx provides operators for transforming, filtering, combining, and aggregating these sequences, enabling concise and expressive code for handling asynchronous operations. Additionally, Rx supports concurrency control and error handling, making it suitable for building responsive and scalable applications.

Azure DevOps: Azure DevOps is a Microsoft cloud service, integrates tools for streamlined software development. It includes Azure Repos for Git hosting, Azure Pipelines for CI/CD automation, Azure Boards for agile project management, Azure Test Plans for testing, and Azure Artifacts for package management. The service facilitates collaboration, automation, and agility, enabling teams to deliver high-quality software efficiently.

3.2 Non-Technical Learning

Apart from the technical learning, what I learnt was the practical application of knowledge. It was fascinating to see how things work in the industry. Here are some of the things that I learned.

Project Management: Especially how agile practices are implemented in daily work. I learnt all that by participating in the agile ceremonies like sprint planning, daily standups, story grooming, retrospectives and so on.

Clean code: Writing clean code involves crafting software that is easy to understand, maintain, and extend. It encompasses practices such as meaningful naming, consistent formatting, clear comments, and modular design. Clean code follows **SOLID**[1] principles, encourages readability, and minimizes complexity by breaking tasks into smaller, manageable units. It prioritizes simplicity over cleverness and emphasizes the importance of writing

[1] SOLID principles in software design advocate for single responsibility, open/closed, Liskov substitution, interface segregation, and dependency inversion to promote maintainable and scalable code.

code for humans rather than machines. By writing clean code, developers contribute to a more collaborative and efficient development process, enabling smoother maintenance and enhancing the overall quality and longevity of the software.

Code smells: Code smells are indicators of potential design issues or shortcomings in software code. Identifying and addressing code smells is crucial for maintaining code quality and ensuring maintainability and scalability. Common code smells include duplicated code, long methods or classes, excessive comments, and complex conditional logic. These smells can lead to difficulties in understanding and modifying code, increased bug susceptibility, and decreased readability and maintainability. Detecting and refactoring code to eliminate smells improves code quality, enhances developer productivity, and reduces technical debt. By addressing code smells promptly, developers contribute to the creation of cleaner, more efficient, and more maintainable software systems.

I learnt how to collaborate with a team in much greater detail than I had ever known during my academics. I learnt why sharing responsibility and trusting your colleagues is of utmost importance when working on complex systems on tight deadlines. I understood the importance of good time management and how companies drive that with the help of agile framework and practices.

Working through the warmup project led me to the discovery of my liking for working on the backend. I also realized that although I like learning new technologies on the fly, it is much more sustainable to understand the basic concepts and learning to apply them across different languages. The project also highlighted the importance of unit testing during development and general testing of an application.

4. Challenges Faced

I came across a few challenges while undergoing the internship. Some of them really pushed me to grow and overcome them. First and foremost, working in a corporate, I had to grasp the top to bottom functionality of the company to ensure that I understood everything properly. Without doing so, one cannot add value to the operation. Thus, the first challenge was to understand every function of the company in as much detail as possible. This becomes difficult when there are multiple complex functions running parallelly.

The bootcamp organized for all the interns giving an overview of the whole organization helped, no doubt. But I had to know more. Hence, I spoke to many people across different teams in their free time and asked them questions about what they did and how their team functioned. This also helped me to establish good rapport with people across the organization.

The next big challenge was to learn the tech stack while simultaneously beginning the work on the warmup project. I tackled this by dividing the workload with my teammates so that all of us could focus on the work as well as on the learning. Managing the time well also helped here.

After being assigned to the team, the first big challenge was to understand the work the team undertakes to a much deeper level. I interacted extensively with the team members to understand the nature of work they do. Next was to understand their code structure and what the code did. To overcome this, I had to sit with the developers and the QA engineer and ask them to explain all the facets of the code. This helped me get a clear idea of functionality of the code.

Once I started work on the new data migration framework, I had to implement it in a new template that was further designed using clean architecture wherein there were different projects within the same solution for executing different functionality. I had trouble setting up the project due to the confusing nature of the template initially. For this, I interacted with the solution architect working in the US who had created this template. I spoke to him a few times to clear all my doubts about the template and got a move on.

As things stand, I am right now continuing my work on the new framework, and I come across new challenges every now and then which I deal with by consulting with my seniors and sorting out the issues at hand.

When facing technical challenges, AI tools like ChatGPT have been of immense help. But to use them effectively, I had to learn to give proper prompts to the model to get appropriate responses. This helped me enhance my skill in using prompt engineering as well.

Appendix I

4.1 Figure of table

Appendix II