

FSM CONFIG MODULE, INVOICE AUDIT SYSTEM AND COPILOT POC

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

VIGNESH VINOD HALDANKAR

Seat Number: 22P0320055

ABC ID: 140611574737

PRN: 20190425

Under the Mentorship of

MR. KEERTHI BHATT

Goa Business School

Discipline of Computer Science and Technology



Goa University

Date: 3rd June 2024

Examined by:

Seal of the School/Dean

DECLARATION BY STUDENT

I hereby declare that the data presented in this Internship report entitled, “FSM Config Module, Invoice Audit System and Copilot POC” is based on the results of investigations carried out by me at Neighborly, under the mentorship of Dr. Jyoti Pawar 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.

Vignesh Haldankar

Date:

Signature and Name of Student

Place: Goa University

Seat no: 2232

COMPLETION CERTIFICATE

This is to certify that the internship report "FSM Config Module, Invoice Audit System and Copilot POC" is a bonafide work carried out by Mr. Vignesh Vinod Haldankar 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.



Keerthi Bhatt

Signature and Name of Mentor

Date : 06th June 2024

Signature of Dean of School

School/Department Stamp

Date:

Place: Goa University



INTERNSHIP CERTIFICATE

This letter is to certify that **Mr. Vignesh Vinod Haldankar**, 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.

Sincerely,

A handwritten signature in blue ink, appearing to be "Shekhar Manjargi".

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

OFFER LETTER

September 22, 2023

Mr. Vignesh Vinod Haldankar
St. Jaquim Road, A-3, 3rd Floor, Chaitra Apts Housing Society, Borda, Margao, South
Goa - 403602.
vigneshhaldankar.vvh@gmail.com
8796924987

Dear Mr. Vignesh Vinod Haldankar,

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,

A handwritten signature in black ink, appearing to read "Shekhar Manjargi".

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:

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 | GSTIN: 29AAUFN7282K1Z8
Place of business: Roshni WeWorks, Marathahalli

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 am grateful to Mr. Keerthi Bhatt and Mr. Manoj Panicker for their support and leadership, as well as to Mr. Hari Shankar, Ms. Sonia Phillip, and Mr. Basavaraju Kidiyappa 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. 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 internship report presents my work and experiences during the internship at Neighborly, completed as a requirement for the Master of Computer Applications at Goa Business School, Goa University. The internship, supervised by Mr. Keerthi Bhatt, involved significant contributions to Neighborly's digital transformation projects.

Throughout the internship, I was tasked with developing an Invoice Audit Management System, a pivotal project aimed at automating the verification process for vendor invoices. This system was developed using a combination of frontend technologies such as React with TypeScript, Material UI, and Bootstrap, and backend API integrations, ensuring a streamlined and accurate auditing process.

In addition to the primary project, I engaged in enhancing the Onverity Field Service Management system by implementing feature short codes for API requests, facilitating efficient feature management. The report elaborates on the methodologies adopted, the technologies used, and the problem-solving approaches applied to overcome various challenges encountered during the projects.

The internship provided a practical platform for applying academic knowledge to real-world scenarios, significantly improving my proficiency in software development and project management. The experience gained during this period is expected to contribute substantially to my professional growth and readiness for future challenges in the tech industry.

Table of Contents

DECLARATION BY STUDENT	i
INTERNSHIP CERTIFICATE.....	iii
OFFER LETTER	iv
ACKNOWLEDGEMENTS	v
EXECUTIVE SUMMARY	vi
1. COMPANY OVERVIEW	1
1.1 Bird's Eye View	1
1.2 Products	3
1.3 Sections in the Organization	4
1.3.1 Digital	4
1.3.2 Integration Apps	6
1.3.3 Data & Analytics	6
1.3.4 Quality Assurance	6
1.3.5 Systems	6
1.3.6 Project Management.....	7
2. TASK(S) HANDLED.....	8
2.1 Invoice Audit Management System.....	8
2.1.1 Section I have worked in.....	9
2.1.2 Type of tasks and Hands-on experience.....	9
2.1.3 Working Schedule.....	10
2.1.4 Software Used	10
2.1.5 Languages, Libraries and frameworks Used	10
2.1.6 Relationship of the task with the course you studied in the classroom	11
2.2 Feature ShortCode Introduction.....	12
2.2.1 Section I have worked in.....	13
2.2.2 Type of tasks and Hands-on experience.....	13
2.2.3 Working Schedule.....	14
2.2.4 Software Used	14
2.2.5 Languages, Libraries and frameworks Used	14
2.3 POC on Github Copilot	15
2.3.1 Type of tasks and Hands-on experience.....	15
2.3.2 Predefined Commands and shortcuts	17
2.3.3 Copilot Usage.....	17
2.3.4 Sub-Task 1	21
2.3.5 Sub-Task 2	21

2.3.6	Sub-Task 3	24
2.3.7	Working Schedule.....	29
2.3.8	Software Used	29
2.3.9	Languages, Libraries and frameworks Used	29
2.3.10	Relationship of the task with the course you studied in the classroom	29
3.	MY LEARNING	30
3.1	My learning from the practical exposure	30
3.2	Tools and Software	31
3.3	Languages, Database and Frameworks	32
4.	Challenges	35
Appendix I:		36
Appendix II :		39
Appendix III :		40
4.1.1	New File Completelt Generated by Copilot	40
4.1.2	Comparative Analysis 1	41
4.1.3	Comparative Analysis 2	43
Appendix IV :		45

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.)



Figure 1: Code OF Values

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

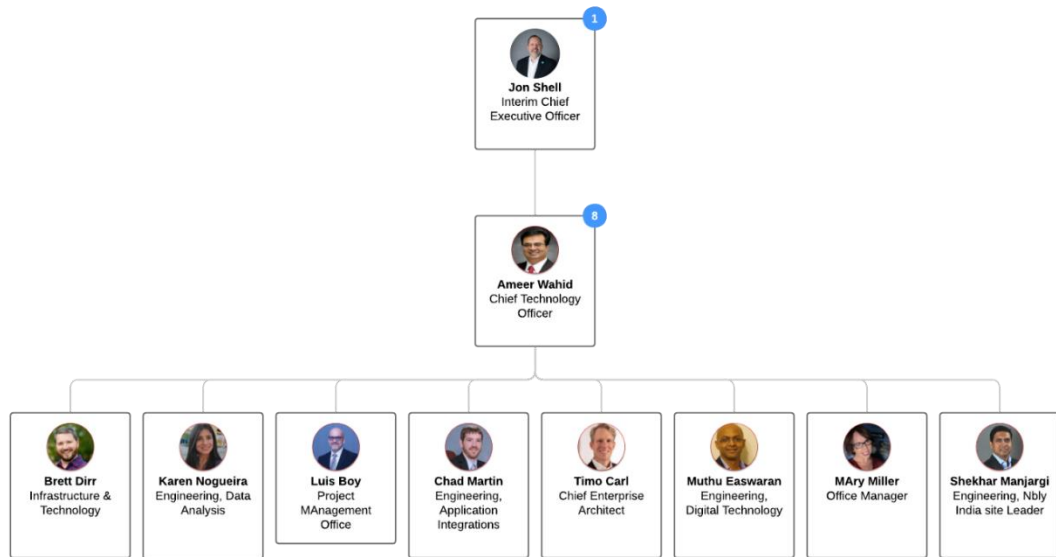


Figure 2: CTO Organization

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

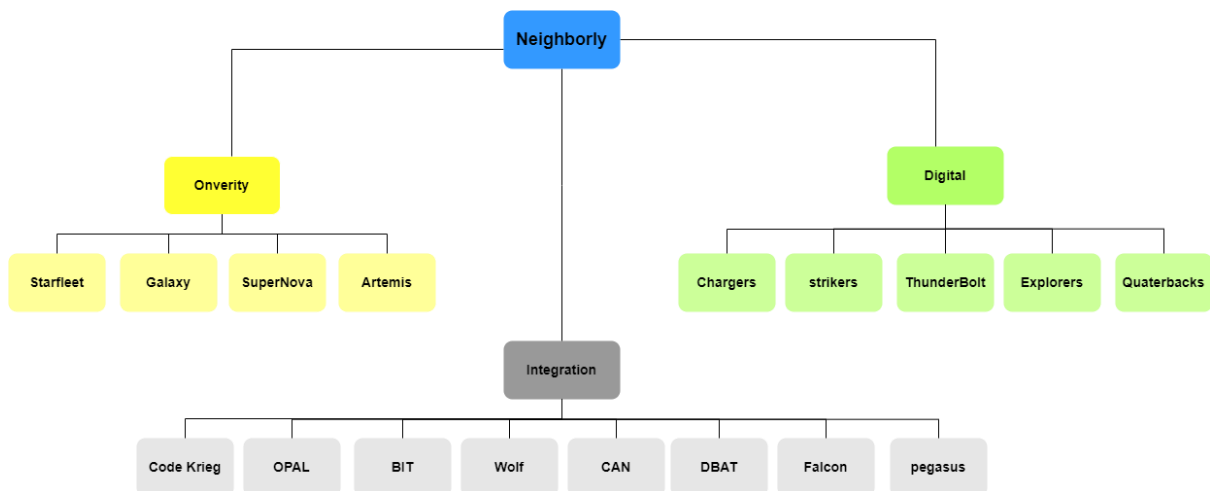


Figure 3 : Neighborly Team Structure

1.2 Products

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 decided to come 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

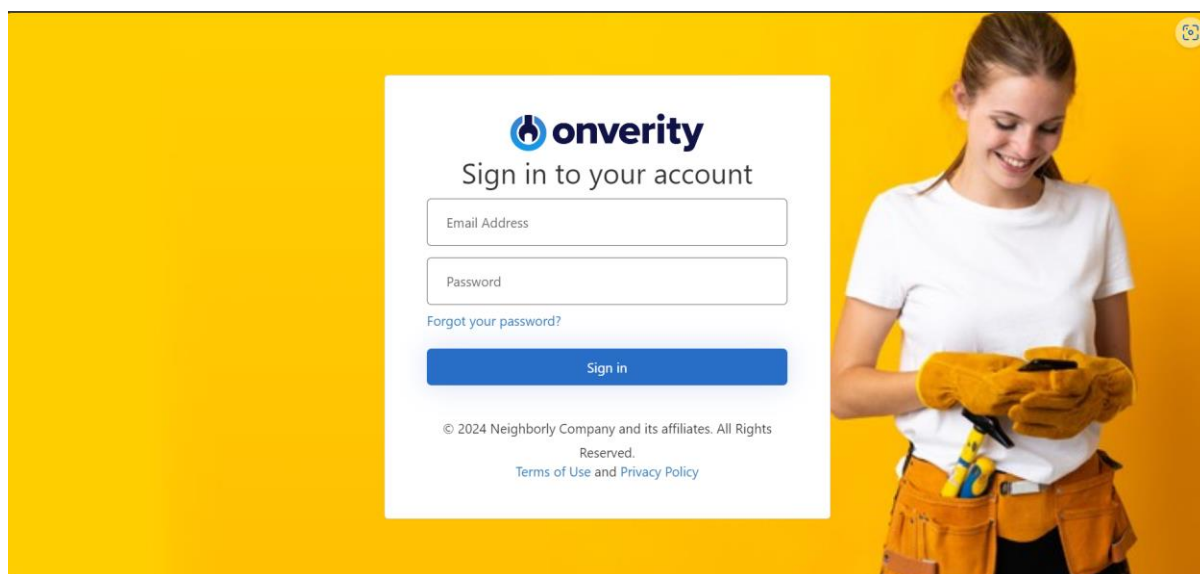


Figure 4: Landing Page of the Onverity Web application.

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 website 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.

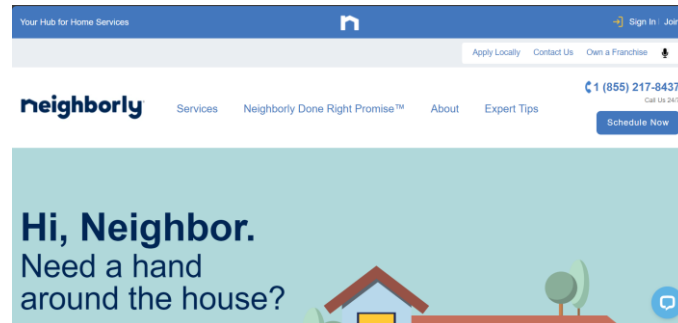


Figure 5: Neighborly Corporate Website

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

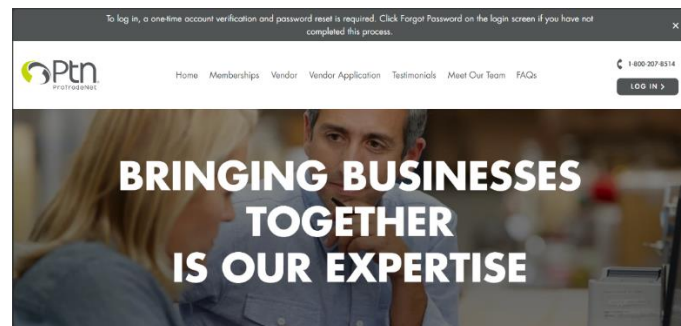


Figure 6 : ProTradeNet website

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

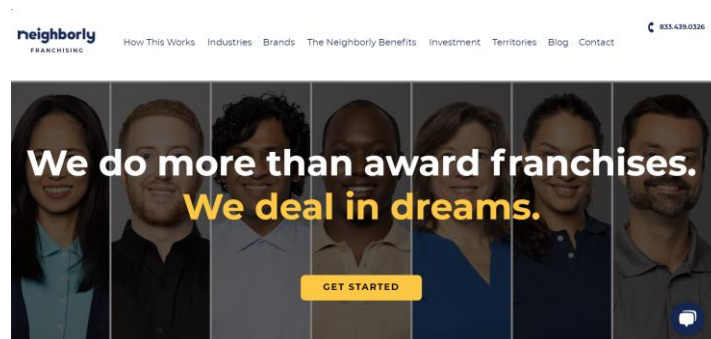


Figure 7 : Neighborly Franchisee Website

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 Integration Apps

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 detrimental 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 Data & Analytics

The Data & Analytics team works with the rest of the organization to help identify key improvement metrics based on all the data available in the Data lake

1.3.4 Quality Assurance

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.

1.3.5 Systems

The systems team focuses on automating and streamlining processes involved in deploying, managing, and maintaining software applications. We utilize tools and practices like continuous integration,

continuous delivery, and infrastructure as code to enhance efficiency and reliability. Additionally, we ensure that Site Reliability Engineering (SRE) practices are implemented across all 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

Invoice Audit Management System is an internal tool that was developed to help the scrum masters keep track of various things. This was done in order to automate the task of verifying the number of hours invoiced by a vendor and finding any discrepancies in the invoice.

The need for this tool was that the scrum masters spend a lot of time comparing invoices given by the vendor for each month to check if all the details in the invoice are correct. Manually doing this is a very tedious and time-consuming task and cuts into productive time for the person undertaking the task.

They included:

- 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.

And considering all these factors, calculate the expected billable hours for each team member and compare them against the hours from the invoice given by the vendor.

To address these challenges, we introduced the Invoice Auditing System. This system comprised four core modules:

- i. Vendor Management

- ii. Team Member Management
- iii. Team Member Leave Management
- iv. Vendor Invoice Auditing

In addition to these modules, the system offered functionalities such as bulk file upload and team member search across all modules. We also included an "Add Holidays" module to capture vendor-specific holidays, enhancing the system's comprehensiveness and adaptability.

This implementation aimed to streamline the invoice auditing process, reduce manual effort, and minimize the risk of errors, ultimately enhancing efficiency and accuracy in financial operations.

2.1.1 Section I have worked in

I have worked on Frontend development of this Project.

2.1.2 Type of tasks and Hands-on experience

I have worked on Frontend development of this Project, kicking off with designing some pages and crafting mockups to visualize the layout using Canva.

implemented an interactive popup skill module using React with TypeScript, Bootstrap, React libraries, Syncfusion, and Material UI. This module streamlined the process of saving employees' skills in the database while adding new team member details.

Additionally, I took charge of implementing a core module of the project, the Leave Management system. This involved dynamically fetching employee data from the database based on the selected project and vendor using API. I integrated autocomplete suggestions using Material UI to enhance user experience. This module played a important role in recording details of employees applying for leaves during a specific time frame, capturing information such as leave dates, types, and quantities. These details proved invaluable during auditing, providing comprehensive insights into employees' working hours and leave records.

Furthermore, I took on the responsibility of maintaining the website's overall CSS and consistency, ensuring adherence to company specific color palettes, logos, fonts, date formats and styling guidelines.

In order to enhance the user experience during lengthy invoice calculation processes, I implemented a Lottie loader, which helped manage user expectations and improve overall usability.

Implemented Search functionality in the Audit module, Team Member module, and Leaves Management module to streamline the efforts of the Scrum Master in locating specific employees among a large pool of personnel.

Conducted manual testing of the Leaves Management module, thoroughly ensuring its functionality and user experience met expectations. Additionally, I explored Cypress component testing to further validate the module's performance and reliability.

2.1.3 Working Schedule

The working schedule at neighborly - Work hours: 08:00 AM - 05:00 PM from Monday to Friday

2.1.4 Software Used

- i. Visual studio
- ii. Visual studio code
- iii. Chrome
- iv. Swagger
- v. Pg admin

2.1.5 Languages, Libraries and frameworks Used

- i. React javascript
- ii. Css styling
- iii. React with typescript
- iv. Material UI library
- v. React Component Library
- vi. Bootstrap library
- vii. Sync fusion library
- viii. Cypress
- ix. C#
- x. Dotnet
- xi. PostgreSQL
- xii. MSql Server

2.1.6 Relationship of the task with the course you studied in the classroom

The web development course I took during my academic studies proved helpful in growing my understanding of key technologies such as React, JavaScript, HTML, CSS, and the Bootstrap library. Through practical exercises and projects, I sharpened my skills in front-end development, gaining skill in building responsive and visually appealing web interfaces.

Additionally, my coursework in database management systems provided me with the knowledge and skills necessary to write fluent queries and effectively interact with databases. Understanding database structures, query languages, and database management principles has proven invaluable in various projects, enabling me to efficiently retrieve, manipulate, and manage data for web applications and other software projects.

2.2 Feature ShortCode Introduction

I was tasked with creating a new field in the Post API Endpoint and implementing mapping based on that field. As part of Team Onverity, I was specifically assigned the responsibility of introducing feature toggle short codes in the POST API request body of ManageFeatureToggle. This enhancement aimed to streamline the mapping process based on short codes, enabling efficient toggling of features.

Previously, feature toggle mapping relied on feature IDs, which were randomly generated in each environment. Accessing these IDs required running the seed script, causing delays during deployment as obtaining the feature ID was time-consuming. To address this challenge, we identified the need for a feature shortcode field in the API request for feature toggle mapping. This shortcode, an alphanumeric string, was proposed for its readability and simplicity. The objective was to establish mapping based on these shortcodes. If a shortcode was not provided, the mapping defaulted to the feature ID. However, if a shortcode was provided, it took precedence in the mapping process, allowing for more granular control over feature toggling.

Advantage of Introducing feature ShortCode

- i. **User-Friendly Mapping:** Using short codes makes it easier for developers and administrators to map features to toggle configurations. Short codes are typically human-readable and easier to remember than GUIDs, which are long and cryptic.
- ii. **Intuitive Configuration:** With feature short codes, configuring toggles becomes more intuitive. Developers can easily identify features by their short codes and toggle them on or off as needed without having to reference lengthy GUIDs.
- iii. **Reduced Error Potential:** Short codes reduce the potential for errors when configuring toggles. Developers are less likely to mistype or misinterpret short codes compared to GUIDs, which can lead to unintended toggling of features.
- iv. **Improved Collaboration:** Short codes can facilitate better collaboration among team members. Since they are more intuitive and easier to remember, team members can communicate about feature toggles more effectively, leading to smoother collaboration and fewer misunderstandings.
- v. **Flexibility in Configuration:** Short codes provide flexibility in configuration by allowing developers and administrators to use meaningful labels or abbreviations for features. This flexibility enables them to

tailor the toggle configuration to match the specific requirements and preferences of the system or project.

- vi. **Enhanced Documentation:** Feature short codes can also improve documentation by providing clear and concise references to features within documentation materials. This makes it easier for new team members to understand the system and how features are configured.
- vii. **Scalability:** As the system grows and more features are added, using short codes for feature toggles ensures scalability. Developers can continue to manage toggles efficiently without being burdened by the increasing complexity of GUID-based configurations.

2.2.1 Section I have worked in

I have worked on Backend development of this Task

2.2.2 Type of tasks and Hands-on experience

My mentor Hari Shankar handed me this task, giving me a good rundown of what needed to be done. We kicked things off with a meeting where I got all the details and made sure I understood why we were doing this task in the first place. Understanding the problem with the current setup was key.

Once I got the green light, I dove into the project repository. There, I explored the code structure and found some interesting patterns they'd been using, like the Repository Service pattern and CQRS. Setting up the local database to access Onverity's data was a bit of a challenge, but I got through it.

With a grasp of how everything was laid out, I started putting things into action. I created new models and method signatures to accommodate the new shortcode feature. It felt good to see the shortcode field pop up in the API endpoint after that first step.

Next up was figuring out how to map the shortcode and fetch the right data from the database. I had to modify the existing code that mapped based on feature ID and come up with a logic that prioritized the shortcode if it was provided in the API request. It took some trial and error, but I got it working.

Once that was sorted, I needed to make sure the shortcode also showed up in the API response. That was a bit challenging because the response was based on a different table that didn't have the shortcode info. So, I had to tweak the table model, add the shortcode column, ran the migration script and created execution and rollback script and made sure everything synced up properly.

After that, it was time to put everything to the test. I did some manual testing on my own machine. Then, I got down to creating unit test cases using the xUnit framework and MOQ library to cover all the bases. Once I was confident everything was working as it should, I pushed the code to the development environment and tested it there as well.

2.2.3 Working Schedule

The working schedule at neighborly - Work hours: 08:00 AM - 05:00 PM from Monday to Friday

2.2.4 Software Used

- i. Visual studio
- ii. Swagger
- iii. Pg admin
- iv. postman

2.2.5 Languages, Libraries and frameworks Used

- i. C#
- ii. Dot net
- iii. Xunit with mock
- iv. PostgreSQL

2.3 POC on Github Copilot

The primary objective was to explore Github Copilot's capabilities and its understanding of different prompts and code snippets. Conducting a thorough investigation of this AI tool was crucial for the benefit of my organization. My task involved working on the company's live project repository to generate unit test cases for code developed by our team of developers. The aim was to assess whether Copilot could produce superior test cases covering various scenarios and edge cases, thereby enhancing code coverage in terms of lines and branches, and identifying potential bugs. This, in turn, would add value to the project, as unit tests play a pivotal role in development.

Also, it would make life easier for our developers. If Copilot could write tests for us, our developers could spend more time writing code instead of worrying about tests. That would help us get things done faster and better.

2.3.1 Type of tasks and Hands-on experience

Methodology:

i. Familiarization with GitHub Copilot and the Project:

Before diving into the POC, a thorough familiarization with GitHub Copilot's functionalities is essential. This includes understanding its integration with the Integrated Development Environment (IDE) and its support for C# .NET projects. Additionally, establishing a comprehensive understanding of the existing project and its testing framework is crucial.

ii. Selection of Methods for Test Case Generation:

A subset of methods or functions within the existing project is carefully chosen to generate unit test cases. These selections prioritize methods with less test coverage, as well as those known for effectively capturing edge cases and maintaining overall quality.

iii. Prompt Creation for GitHub Copilot:

For each selected method, a tailored prompt is crafted, delineating the method signature and any specific requirements or constraints. These may include directives for utilizing the Moq library for mocking dependencies and ensuring coverage of various scenarios. These prompts serve as input to GitHub Copilot for generating unit test cases.

iv. Unit Test Case Generation with GitHub Copilot:

GitHub Copilot is invoked within the Integrated Development Environment (IDE), and the formulated prompts are provided as input. Leveraging its code generation capabilities, Copilot generates unit test cases for the selected methods based on the provided prompts.

v. Review of Existing Manual Test Cases:

Manual unit test cases for the same methods, developed by the team, are thoroughly gathered, and analysed. This review encompasses an analysis of test coverage, effectiveness in capturing edge cases, and overall quality.

vi. Comparative Analysis:

A comprehensive comparative analysis is conducted to evaluate the test cases generated by GitHub Copilot against the existing manual test cases. This analysis considers factors such as test coverage, quality, and efficiency in identifying potential issues.

vii. Documentation and Reporting:

Findings from the comparative analysis are meticulously documented in a comprehensive report. This report details observations, insights, strengths, weaknesses, understandings, and recommendations. Additionally, it provides insights into the effectiveness of GitHub Copilot in test case generation and its potential implications for the project's testing practices.

2.3.2 Predefined Commands and shortcuts

Alt + /	Invoke copilot In code
Tab	Accept inline suggestion
Esc	Dismiss inline suggestion
Alt + .	Next suggestion
Alt + ,	Previous suggestion
/ comment	Generates code based on comment
#filename	Reference a file/code snippet

2.3.3 Copilot Usage

Step 1: Invocation

Method 1: Invoke Copilot Suggestion

- In your code editor, press Alt + / to invoke Copilot's suggestion.

Method 2: Open Copilot Chat

- Navigate to View in the top menu.
- Select GitHub Copilot Chat to open the chat interface.

Step 2: Prompt Creation

Be Clear and Specific

- State the Purpose: Clearly state what the test is supposed to verify.
- Specify the Method: Indicate which method you are testing and what the expected outcome is.
- Input Parameters: Include details about the input parameters and their types.

Use Descriptive Method Names (Optional)

- Descriptive Names: Use method names that clearly describe what is being tested and the expected result. Example: Add_ReturnsCorrectSum

Be Specific About Frameworks and Dependencies

- Testing Framework: Specify the testing framework (e.g., xUnit, NUnit, MSTest).
- Libraries and Dependencies: Mention any specific libraries or dependencies required for the test.

Write Comments

- Detailed Comments: Write detailed comments that describe each step of the test: setup, action, and assertion.

Sample Prompts

- i. /tests generate <testingFramework> test case for <methodName> method.
- ii. /tests generate <testingFramework> test case using <testingLibrary> library for mocking and mock the <methodName> method.
- iii. /tests generate <testingFramework> test case using <testingLibrary> library for mocking and mock the <methodName> method #<selectedCode>.
- iv. /tests generate <testingFramework> test case using <testingLibrary> library for mocking and mock the <methodName> method. Refer the file #<fileName>
- v. /tests generate <testingFramework> test case using <testingLibrary> library for mocking and mock the <methodName> method #<selectedCode>. Refer the file #<fileName>

/fix Command

- The /fix command in GitHub Copilot can help resolve minor issues and errors in your code.
- This command provides quick fixes and suggestions to improve code quality and rectify common mistakes.
- Copilot generates new functions, constructors and asks us to implement them in our src code
- This can be fixed by prompting

- `/fix <function>` doesnot exist in my project, do necessary changes in test case

Invoke Copilot Suggestions

- In your code editor, press `Alt + /` to invoke Copilot's suggestion interface.
- Alternatively, open Copilot chat: View -> GitHub Copilot Chat.

Identify the Issue

- Navigate to the part of your code with an issue or error.
- Place your cursor on or near the problematic code.

Use the `/fix` Command

- Context Matters: Ensure the context around the `/fix` comment is clear so Copilot can provide accurate suggestions.
- Review Suggestions: Always review the suggested fixes to ensure they align with your intended logic and project requirements.
- Iterative Improvement: Use `/fix` iteratively to address multiple issues in your code step by step.

Sample Prompts

- `/fix` errors in the selected code snippet to ensure correct functionality.
- `/fix` issues in line 24 to resolve any bugs or inconsistencies.
- `/fix` (mention error) in the selected code to prevent runtime errors.
- `/fix` (will check the whole file for errors

Handle Errors, Exceptions & Hallucinations:

When using GitHub Copilot to generate unit test cases in Visual Studio Professional, you may encounter various issues such as minor errors, exceptions, and AI hallucinations. This guide provides strategies for resolving these issues effectively.

ERRORS

Resolving Minor Errors Using the /fix Command:

- Identify the Error: Navigate to the part of the code with the error.
- Invoke Copilot: Press Alt + / to open Copilot suggestions or use View -> GitHub Copilot Chat
- Review Suggestions: Apply the appropriate fix suggested by Copilot.

Resolve Using Copilot chat:

- Identify the Error: Navigate to the part of the code with the error.
- Copy the error message and paste it in Copilot chat
- Review Suggestions: Apply the appropriate fix suggested by Copilot.

EXCEPTIONSHandling Common Exceptions: System.NullReferenceException

- Identify the Issue: This often occurs when Copilot fails to mock properties or parameters correctly.
- Use the /fix Command: To resolve the issue.
- Check Parameters: Ensure all parameters are mocked and passed properly.

Debugging Steps:

- Add Debugger: Add a debugger on the ACT and ASSERT lines.
- Debug Line by Line: Find where the exception occurs.
- Mock Missing Properties:
- Manually or with Copilot: Tell Copilot to mock the specific property.

OR

- Select the line where exception occurs
- Tell copilot that you are getting an SYSTEM NULL REFERENCE EXCEPTIONS on line #linenumber and mock the necessary properties for testing

HALLUCINATIONSUnderstanding Hallucinations:

- Lack of Context: AI might generate functions or constructors that do not exist in your project.
- Provide Proper Context: Clearly define the methods and properties to be mocked.

Resolving Hallucinations:

- Identify Non-existent Functions: When Copilot generates new functions or constructors.

2.3.4 Sub-Task 1

Generating unit test case for a newly developed file

Objective:

The aim of this task is to generate a comprehensive unit test file using GitHub Copilot for the Brand Specific that was newly developed by the Neighborly developers., leveraging the Xunit framework and Moq library.

The Aim:

- Evaluate the ability of GitHub Copilot to generate unit test cases that cover all scenarios and edge cases.
- Ensure the generated test cases achieve full line coverage and branch coverage.
- Produce test cases that are better than those written by developers manually.
- Reduce the burden on developers for writing test cases, allowing them to focus more on development tasks.

2.3.5 Sub-Task 2

Generating unit test cases for an Existing file and comparing it with the unit tests created by Developers.

Objective:

The goal of this task is to generate unit test case files for existing code using GitHub Copilot and then compare these generated test cases with those already developed by the Neighborly developers.

The Aim:

- Evaluate the coverage and quality of GitHub Copilot-generated test cases.
- Assess whether the generated test cases cover all scenarios and edge cases.
- Determine if the generated test cases fulfil line and branch coverage requirements.

- Identify areas where GitHub Copilot can reduce the burden on developers, allowing them to focus more on development.

Result:

Sonar Cloud Quality Gate Result of Test file Written By Developers

- Line Coverage – 26%
- Branch Coverage – 50%

Sonar Cloud Quality Gate Result of Test file Generated by Github Copilot

- Line Coverage – 97%
- Branch Coverage – 83%

Report

COMMON ELEMENTS

Shared Imports and Libraries:

- Both files use `Moq`, `Serilog`, `Xunit`, and other necessary libraries.
- They both mock the same interfaces: `IWorkOrderRepository`, `ILogger`, and `IAWSS3BucketHelper`.

Test Class Initialization:

- Both test classes initialize a `MockRepository` and create mocks for `IWorkOrderRepository`, `ILogger`, and `IAWSS3BucketHelper`.
- Both classes follow a similar structure in their constructors to set up these mocks.

DIFFERENCES

Test Cases and Assertions:

DEV FILE:

- Contains four main tests: ``HandleAsync_StateUnderTest_ExpectedBehavior``, ``HandleAsync_Exception``, ``HandleAsync_Exception_Valid_WorkId_AgreementId_SignedAgreement``, and ``HandleAsync_Exception_Valid_WorkId_AgreementId_UnSignedAgreement``.
- Uses a try-catch block in each test method to handle exceptions and asserts using ``Assert.False(false)`` and ``Assert.NotNull(result)``.
- The setup within each test includes mock setups and direct handling of exceptions.

COPILLOT FILE:

- Contains more diverse test methods, including tests for different statuses and a method named ``GenerateHtml`` for HTML generation.
- Includes more specific assertions, such as ``Assert.Equal``, to verify that specific properties of the returned object match expected values.
- Uses the ``[Theory]`` attribute and ``[InlineData]`` to handle parameterized tests, making it more extensible and maintainable for multiple cases.
- Contains a specific test to handle exceptions and verify error logging using ``mockLogger.Verify``.

ADDITIONAL FUNCTIONAL TESTS: COPILLOT FILE has additional functional tests:

- ``GenerateHtml_StateUnderTest_ExpectedBehavior``: Tests an internal method for generating HTML.
- ``GetAggrement_StateUnderTest_ExpectedBehavior``: Tests another internal method.
- ``HandleAsync_ThrowsException_ErrorLoggedAndApiExceptionThrown``: Specifically tests error logging and exception handling using ``ApiException``.

MOCK SETUP AND USAGE:

DEV FILE:

- Mock setups are repetitive across tests, e.g., setting up ``mockWorkOrderRepository`` and ``mockLogger`` in each test method.
- Tests primarily focus on handling exceptions and verifying results are not null.

COPILOT FILE :

- More extensive use of mocks, including setups for `GetPreSignedURL` and `GetFile`.
- Utilizes a `mockRepository.Verify` to ensure methods are called the expected number of times, providing better coverage and verification.
- More sophisticated setup and verification of mock behavior to ensure the mocks behave as expected in the tests.

SUMMARY

- DEV FILE is straightforward but lacks extensibility and detailed verification. It focuses on handling basic cases and ensuring that exceptions do not break the test flow.
- COPILOT FILE is more comprehensive, covering additional functionalities and providing more detailed assertions and verifications. It includes parameterized tests, method-specific tests, and better mock verification, making it more robust and maintainable.

VERDICT : File 2 demonstrates a more advanced and thorough testing approach

2.3.6 Sub-Task 3

Generating unit test cases for an Existing file and comparing it with the unit tests created by Developers.

Objective:

The goal of this task is to generate unit test case files for existing code using GitHub Copilot and then compare these generated test cases with those already developed by the Neighborly developers.

The Aim:

- Evaluate the coverage and quality of GitHub Copilot-generated test cases.
- Assess whether the generated test cases cover all scenarios and edge cases.
- Determine if the generated test cases fulfil line and branch coverage requirements.
- Identify areas where GitHub Copilot can reduce the burden on developers, allowing them to focus more on development.

Result:

Sonar Cloud Quality Gate Result of Test file Written By Developers

- Line Coverage – 47%
- Branch Coverage – 50%

Sonar Cloud Quality Gate Result of Test file Generated by Github Copilot

- Line Coverage – 100%
- Branch Coverage – 100%

Report :**Overview**

DEV FILE:

- Focuses on testing `Send` method.
- Uses `Moq` for mocking dependencies.
- Contains tests for both non-generic and generic `Send` methods.
- Tests for null arguments to ensure exception handling.
- Asserts that the mocked mediator calls are verified.

COPILLOT FILE:

- Tests a broader range of methods: `Send`, `SendTResult`, `Broadcast`, and `Fetch`.
- Uses `Moq` for mocking dependencies.
- Contains data-driven tests using `Theory` and `InlineData`.
- Includes tests for null arguments to ensure exception handling.
- Asserts that the mocked mediator calls are verified.

SCOPE & COVERAGE

DEV FILE: Focuses narrowly on the `Send` method, both generic and non-generic variants.

Tests include:

- Sending a command and verifying behavior.
- Handling null commands and checking for `ArgumentNullException`.

COPILLOT FILE : Broader in scope, covering `Send`, `SendTResult`, `Broadcast`, and `Fetch` methods.

Tests include:

- Sending commands and receiving results.
- Broadcasting events.
- Fetching queries and receiving results.
- Data-driven tests for multiple scenarios.
- Handling null inputs for all methods and checking for `ArgumentNullException`.

USE OF MOCKING

DEV FILE:

- Uses a strict `MockRepository` to ensure all expectations are met.
- Mocks the `IMediator` and sets up expected behaviors for `SendAsync`.
- Verifies all mocked interactions.

COPILLOT FILE:

- Similarly uses a strict `MockRepository`.
- Mocks the `IMediator` for `SendAsync`, `BroadcastAsync`, and `FetchAsync`.
- Sets up expected behaviors and verifies all mocked interactions.

EXCEPTION HANDLING

DEV FILE:

- Tests for null command arguments to ensure `ArgumentNullException` is thrown.
- These tests cover both generic and non-generic `Send` methods.

COPILLOT FILE:

- Tests for null inputs for `Send`, `SendTResult`, `Broadcast`, and `Fetch` methods.
- Ensures that `ArgumentNullException` is thrown in each case.

DATA DRIVEN TESTING

DEV FILE:

- Does not use data-driven testing; each test is independent and hardcoded.

COPILLOT FILE:

- Uses `Theory` and `InlineData` attributes to test multiple scenarios with different inputs for `SendTResult` and `Fetch`.

STRENGTHS AND WEAKNESSES

DEV FILE:

- Strengths: Focused, ensures expected behavior and exception handling for `Send` method.
- Weaknesses: Limited in scope, does not cover other methods like `Broadcast` and `Fetch`.

COPILLOT FILE:

- Strengths: Comprehensive coverage, includes a wider range of methods, uses data-driven testing, thorough exception handling.
- Weaknesses: Generated code might lack some context-specific nuances that developers might consider important.

RECOMMENDATIONS FOR IMPROVEMENT

DEV FILE:

- Expand the test coverage to include other methods such as `Broadcast` and `Fetch`. Introduce data-driven tests to increase robustness.

COPILLOT FILE:

- Ensure that generated tests align with specific business logic and requirements. Review and refine tests to capture any context-specific edge cases that automated tools might miss.

CONCLUSION

- The GitHub Copilot-generated file offers a more comprehensive and varied set of tests. It includes thorough exception handling and data-driven tests, making it robust for ensuring reliability. However, it is essential to review and refine these generated tests to ensure they meet specific application needs.

VERDICT

- GitHub Copilot-generated file demonstrates a more advanced and comprehensive testing approach. It is more thorough in covering different methods and scenarios, making it the superior choice for robust testing. However, combining the focused and context-specific insights from Developers File with the broader coverage and robustness of GitHub Copilot-generated file would provide the most comprehensive testing strategy.

2.3.7 Working Schedule

The working schedule at neighborly - Work hours: 08:00 AM - 05:00 PM from Monday to Friday

2.3.8 Software Used

- i. Visual studio
- ii. Visual studio code
- iii. Github Copilot

2.3.9 Languages, Libraries and frameworks Used

- i. C#
- ii. Dotnet
- iii. Xunit Framework
- iv. MOQ library

2.3.10 Relationship of the task with the course you studied in the classroom

In my academic curriculum, I had a course on Machine Learning where we delved into exploring ChatGPT to generate various types of data. One notable experience was during an exam, where our teacher set a question paper, and the answers were generated by ChatGPT. Later, we manually answered the same paper, and the results were compared. This course not only facilitated interaction with AI but also improved my prompt engineering skills.

3. MY LEARNING

3.1 My learning from the practical exposure

Working in a corporate environment provided me with invaluable insights into real-world working culture and problem-solving processes. I had the opportunity to actively participate in Scrum ceremonies such as daily standups, Sprint planning, grooming, and backlog refinement. These experiences, which were previously theoretical concepts in my curriculum, allowed me to witness how teams collaborate and address challenges efficiently.

Moreover, working on live projects within the company exposed me to technologies like C# .NET, an object-oriented language extensively used in backend development. I gained practical experience in implementing and maintaining design patterns such as CQRS, RSP, and MVC, which are crucial in real-world application development.

During my first task, I acquired in-depth knowledge of frontend technologies including React, TypeScript, HTML, CSS, Bootstrap, and Material UI libraries. Additionally, I also came across UI testing using Cypress, which enhanced my understanding of ensuring application quality and user experience.

Neighborly had given us access to platforms like LinkedIn Learning that played a crucial role in my learning journey, allowing me to dive deeper into various topics such as C# .NET and its functionalities. Furthermore, I actively sought out articles and resources to expand my understanding of specific areas related to backend development.

At first, I liked working on the frontend of projects. But as I started doing backend tasks, I got more curious about it and ended up liking it too. This journey has broadened my skill set and deepened my understanding of the workings involved in developing robust and efficient software solutions.

Working with AI tools like ChatGPT, Codium AI, and GitHub Copilot has been instrumental in my understanding of large language models. I've learned effective practices for interacting with AI, including generating prompts that yield better results and accomplishing specific tasks efficiently. These experiences have provided insights into how AI models comprehend and respond to input, guiding me in crafting prompts to elicit desired outcomes.

Moreover, utilizing AI tools has enhanced my ability to generate quality test cases for various methods. By leveraging these tools, I can efficiently create comprehensive test scenarios, ensuring

thorough coverage of functionality and edge cases. This hands-on experience has deepened my understanding of software testing principles and methodologies, contributing to improved software quality assurance practices.

Furthermore, my internship journey has equipped me with essential soft skills. I've refined my time management, team collaboration, and problem-solving abilities. Learning to work within sprints and assisting others has fostered a culture of mutual support and growth within the organization. Additionally, developing communication skills through interactions with colleagues has been invaluable for personal and professional development.

3.2 Tools and Software

i. 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.

ii. 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.

iii. PG Admin :

PGAdmin is an open-source administration and development platform for PostgreSQL databases. It provides a graphical interface to perform tasks such as database management, querying, schema design, and data manipulation.

iv. Swagger :

Swagger is a tool for designing, documenting, and testing APIs. It allows developers to define API specifications using a YAML or JSON format, which can then be used to generate interactive API documentation and client SDKs.

v. Github Copilot :

GitHub Copilot is an AI-powered code completion tool developed by GitHub. It suggests code snippets and completions based on the context of the code being written, helping developers write code faster and more efficiently.

vi. Sonar cloud :

SonarCloud is a cloud-based code quality and security platform. It analyzes code for bugs, vulnerabilities, code smells, and other quality issues, providing actionable feedback to developers to improve code quality and maintainability.

vii. ADO :

ADO (Azure DevOps): Azure DevOps, formerly known as Visual Studio Team Services, is a set of cloud-based collaboration tools for software development. It provides features like version control, Agile project management, continuous integration/continuous deployment (CI/CD), and application monitoring.

3.3 Languages, Database and Frameworks

i. 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.

ii. Dot net :

.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.

iii. 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.

iv. Typescript :

TypeScript is a superset of JavaScript developed by Microsoft. It adds static typing and other features to JavaScript, making it more scalable and maintainable for large-scale applications. TypeScript code is transpiled to plain JavaScript for execution in web browsers or Node.js environments.

v. Css :

CSS (Cascading Style Sheets) is a stylesheet language used for describing the presentation of a web page written in HTML. It is used to control the layout, styling, and appearance of HTML elements on a web page.

vi. Material UI :

Material UI is a popular React component library based on Google's Material Design principles. It provides pre-designed UI components and styles that can be easily integrated into React applications to create modern and visually appealing user interfaces.

vii. Bootstrap Library :

Bootstrap is a front-end framework for developing responsive and mobile-first websites and web applications. It provides a collection of CSS and JavaScript components that simplify the process of building user interfaces.

viii. React Component Library :

React component library consists of pre-designed UI components that can be easily integrated into React applications. These libraries provide ready-to-use components like buttons, forms, and cards, streamlining UI development. They follow consistent design systems for cohesive interfaces and include popular libraries such as Material-UI and Ant Design. Utilizing a component library accelerates development and ensures a uniform look and feel across applications.

ix. Xunit Framework :

xUnit is a testing framework for the .NET framework. It provides a simple and extensible architecture for writing and running unit tests in C#, F#, and other .NET languages. xUnit supports features like parameterized tests, test fixtures, and test runners.

x. MOQ library :

Moq is a popular mocking library for .NET applications. It allows developers to create mock objects for testing purposes, enabling isolated unit testing of classes and components that have dependencies.

xi. PostgreSQL :

PostgreSQL is an open-source relational database management system (RDBMS). It is known for its robustness, reliability, and advanced features like support for JSON, JSONB, and geospatial data types. PostgreSQL is widely used in web development and enterprise applications.

xii. Cypress :

Cypress is a JavaScript-based end-to-end testing framework for web applications. It provides an elegant and developer-friendly API for writing and organizing tests, as well as features like automatic waiting, time travel, and real-time reloads for rapid test development.

xiii. MS SQL Server :

MS SQL Server, developed by Microsoft, is a powerful relational database management system (RDBMS) known for its scalability, security, and performance. It enables efficient storage, retrieval, and management of structured data, while also offering robust features for business intelligence, high availability, and integration with the Microsoft ecosystem.

4. Challenges

Initially, adjusting to different Scrum ceremonies like daily standups, Sprint planning, grooming, backlogs posed a challenge, but after a few days, I became accustomed to them. The use of software tools like ADO was entirely new to me; navigating user stories and locating the required data, logging daily hours proved to be a task. While I had prior experience with plain React.js during my curriculum, adapting to TypeScript was initially challenging. Defining different types and resolving errors required some time and effort.

Learning new languages and frameworks like C# and .NET was a gradual process. I relied on tutorials, articles, and courses, seeking guidance from colleagues and seniors along the way. Additionally, various design patterns and strategies were employed in the project repository, requiring an understanding of workflows and project structures.

Understanding the code written by different developers making sure the AI generated test cases thoroughly cover the methods, validate various scenarios, and address edge cases demands careful analysis and foresight. Ensuring that test cases effectively detect bugs, validate input boundaries, and handle exceptional conditions requires attention to detail.

Verifying the quality of test cases involves assessing their effectiveness in validating the intended behaviour of methods and detecting potential defects. It requires evaluating test coverage, identifying gaps, and ensuring that tests are robust, maintainable, and reusable. plus getting in touch with the developers that had written these files as some of them were from different time zones.

Meeting deadlines was paramount, and I prioritized completing tasks ahead of schedule. Despite the challenges, I successfully managed to complete all my assignments promptly.

Appendix I:

TASK 1 – Invoice Audit System

Team Roster

Vendor Name: Vendor X | Project Name: Project X

Choose File (No file chosen) | DOWNLOAD TEMPLATE

Search Employee

Team Member Name	IsBillable	Location	Start Date	End Date	Skills	Action
ENTER NAME	YES	India	Start Date	End Date	Select Skills	ADD
Jhon doe	YES	India	17/8/2024	27/8/2024	.net,react,C#	[Edit] [Delete]

Team Roster Module that helps in saving the team member details based on the Vendor and project
Data can be added manually or by uploading a file

Team Roster

Vendor Name: Vendor X | Project Name: Project X

Choose File (No file chosen) | DOWNLOAD TEMPLATE

Search Employee

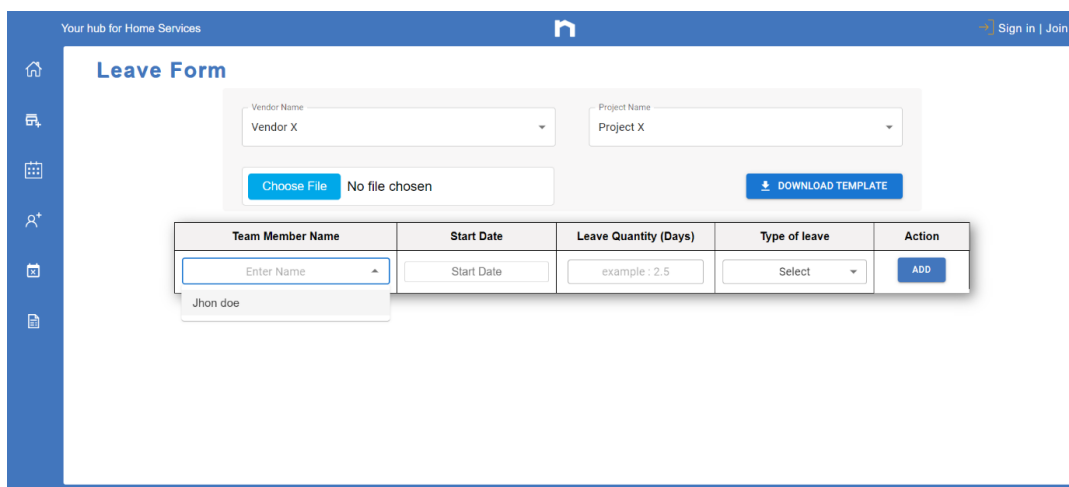
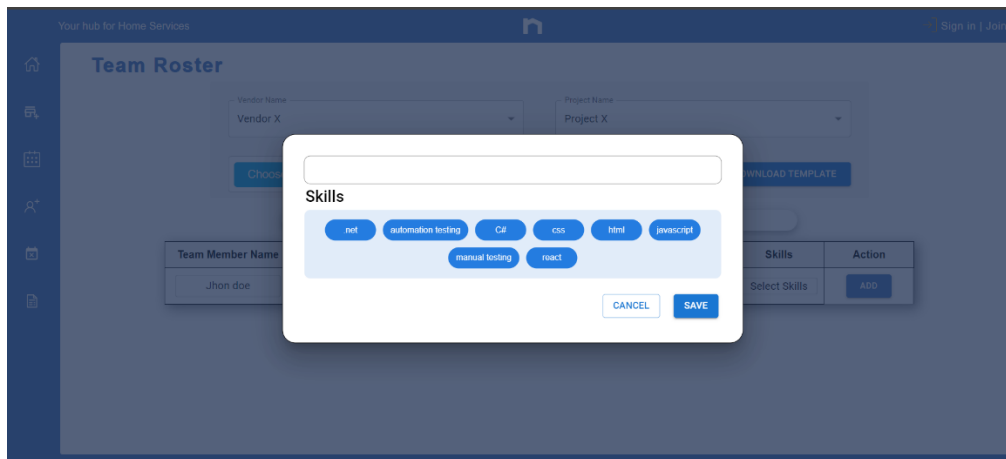
Skills

.net react C#

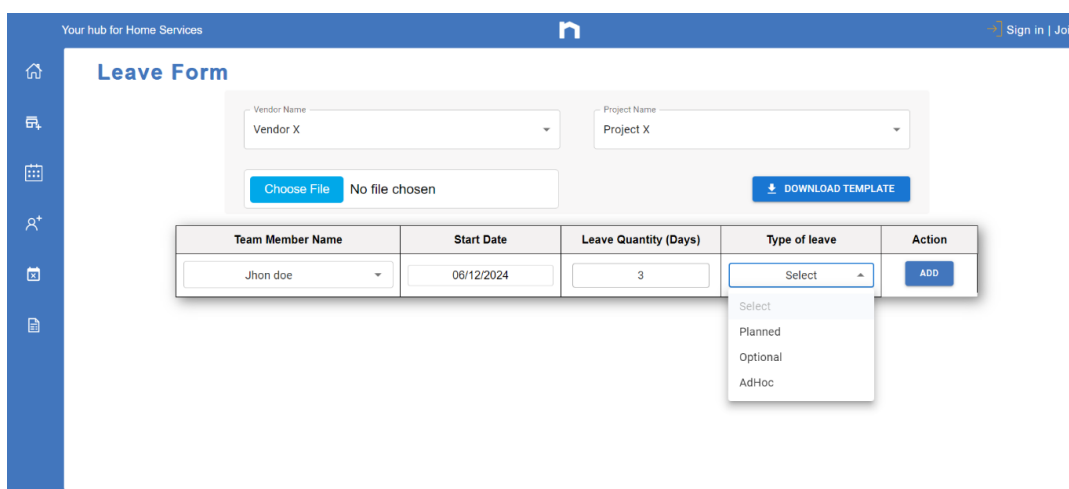
.net automation testing C# css html javascript manual testing react

CANCEL SAVE

Team member skill module The above model was developed by me using Material UI, Skills were fetched from database using Api.



Leave management module that recorded leaves of Team members. With autocomplete functionality that fetched team member names from database based on Vendor and Project name. data can be added manually/through a file



Your hub for Home Services Sign in | Join

Invoice Audit

Vendor Name
Vendor X


Project Name
Project X

Month
June

Year
2024

Choose File No file chosen

AUDIT



LOADING

Invoice Audit Module where the actual auditing was performed.

Added lottie loader to improve user experience

Your hub for Home Services Sign in | Join

Invoice Audit

Vendor Name
[REDACTED]

Project Name
[REDACTED]

Month
June

Year
2024

Choose File No file chosen

AUDIT

Discrepancies : 8.5 hr Number of Discrepancies : 2

Vendor Name : [REDACTED] Month : January
Project Name : [REDACTED] Year : 2024

Search Employee

Employee	Invoiced Hours	Expected Hours	Variance
[REDACTED]	136	128	8
[REDACTED]	144.5	144	0.5
[REDACTED]	0	184	-184

Scroll to Top

Final Output of this project. Number of discrepancies and hours along with the team member names

Your hub for Home Services Sign in | Join

Invoice Audit

Vendor Name
[REDACTED]

Project Name
[REDACTED]

Month
June

Year
2024

Choose File No file chosen

AUDIT

Discrepancies : 8.5 hr Number of Discrepancies : 2

Vendor Name : [REDACTED] Month : January
Project Name : [REDACTED] Year : 2024

Search Employee
Ba[REDACTED]

Employee	Invoiced Hours	Expected Hours	Variance
ba[REDACTED]	136	128	8

Scroll to Top

Search functionality in audit module

Appendix II :

TASK 2 – Introduction of feature toggle short code

API ENDPOINT



The image shows a web-based API client interface for a POST request to the endpoint `/api/Toggle/ManageToggleFeature`. The interface includes a 'Parameters' section with 'No parameters', a 'Request body' section with a content type of `application/json-patch+json`, and an 'Example Value' section. The example value is a JSON object with a `featureId` and a `code` field. The `code` field is highlighted with a red box and a callout indicating it is the 'Feature Short Code'.

POST /api/Toggle/ManageToggleFeature

Parameters Try it out

No parameters

Request body application/json-patch+json

Example Value | Schema

```
[
  {
    "featureId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "code": "string",
  }
]
```

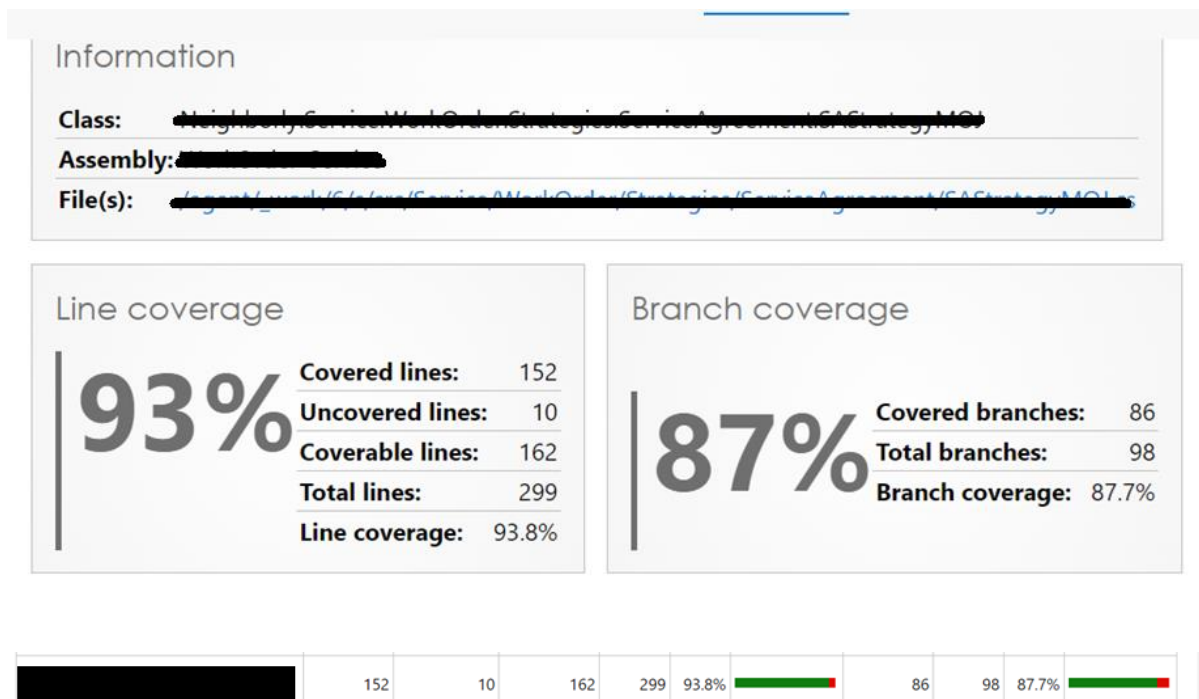
field labeled Code in API Request body ("Feature Short Code")

Feature Short Code (code) field added in Manage Toggle Feature Endpoint

Appendix III :

TASK 3 – POC on Github Copilot

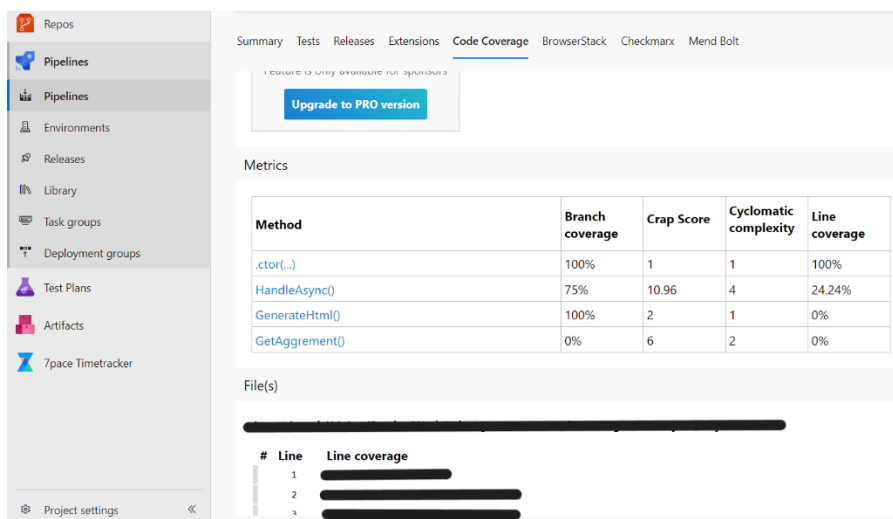
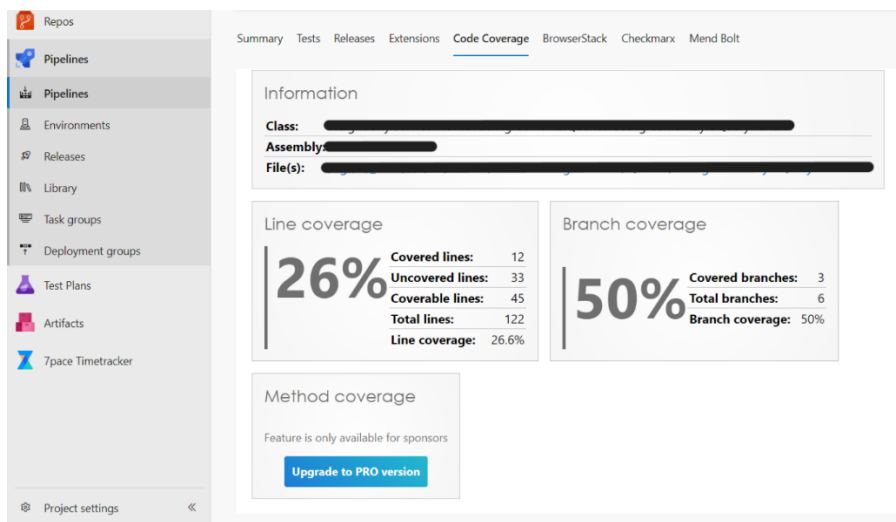
4.1.1 New File Completelt Generated by Copilot



File generated by copilot generated test cases for all the methods of FILE 1, covered different scenarios, generated test cases for handling exceptions, test cases using different inputs in payload, null and empty parameter, data driven test cases. As a result covering max lines, branches and actually generating better cases and adding value to the file

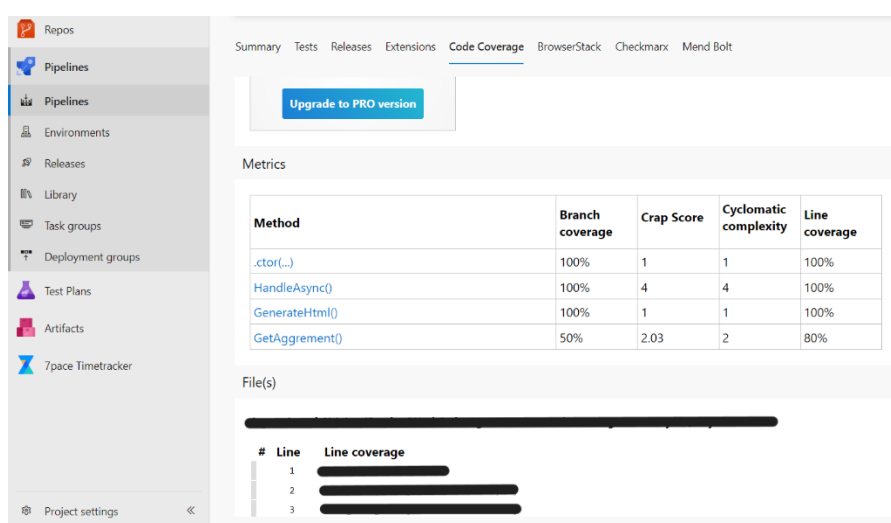
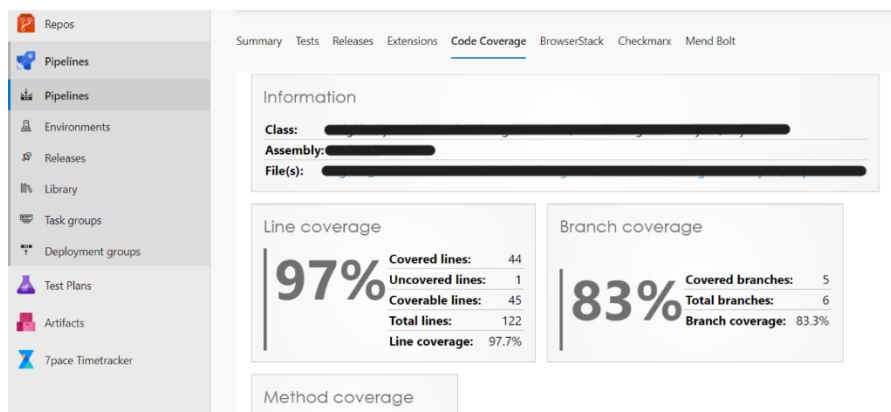
4.1.2 Comparative Analysis 1

Sonar Cloud Quality Gate Result of Test file Written By Developers



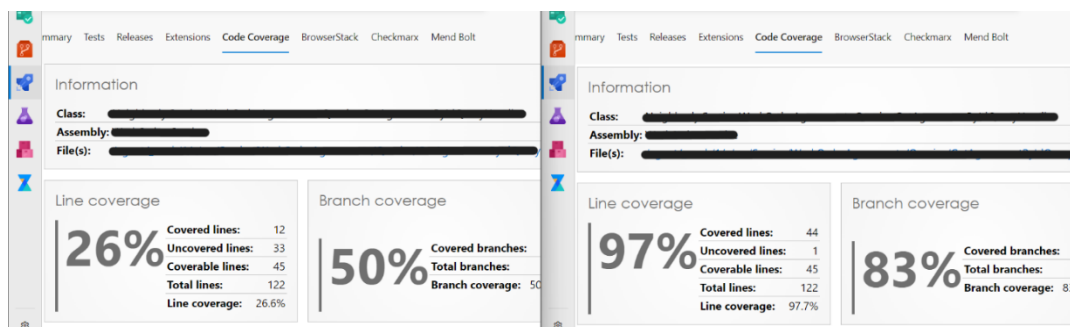
Tests created by developers did not cover some methods and different scenarios.

Sonar Cloud Quality Gate Result of Test file Generated by Github Copilot



Github Copilot covered all the methods present in FILE 2 and generated test cases for various scenarios and also handled exceptions. As a result it improved the line, branch coverage and added value to the test case file. It did not cover all the lines and branches but definitely improved it

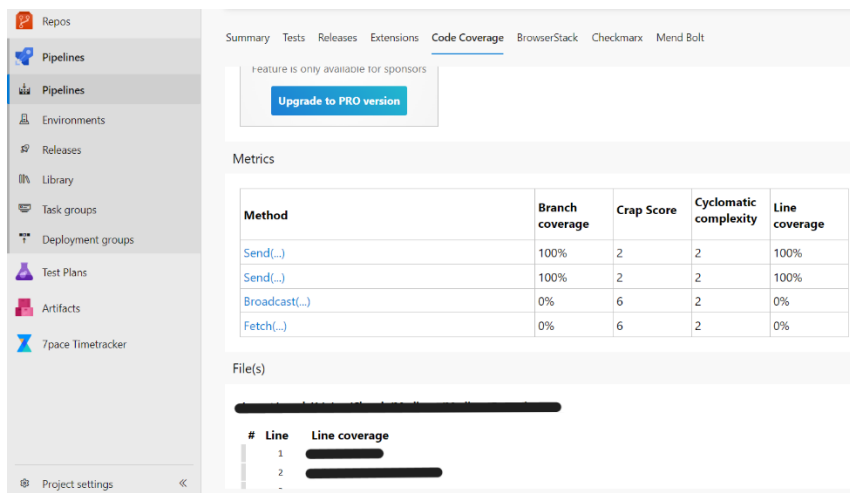
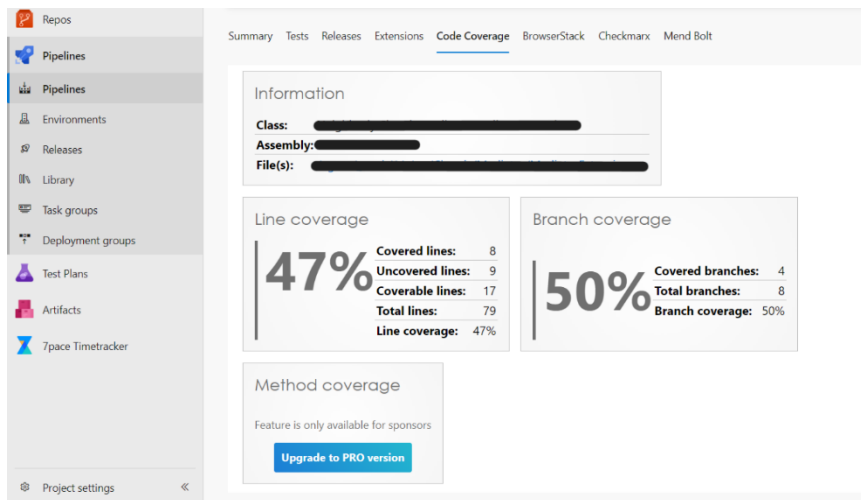
Comparison



Side by side comparison of the test files created by developers with the one generated by copilot

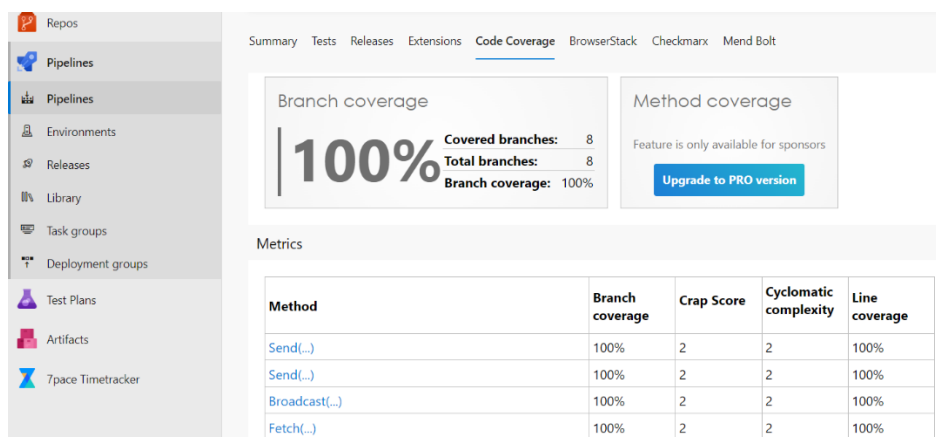
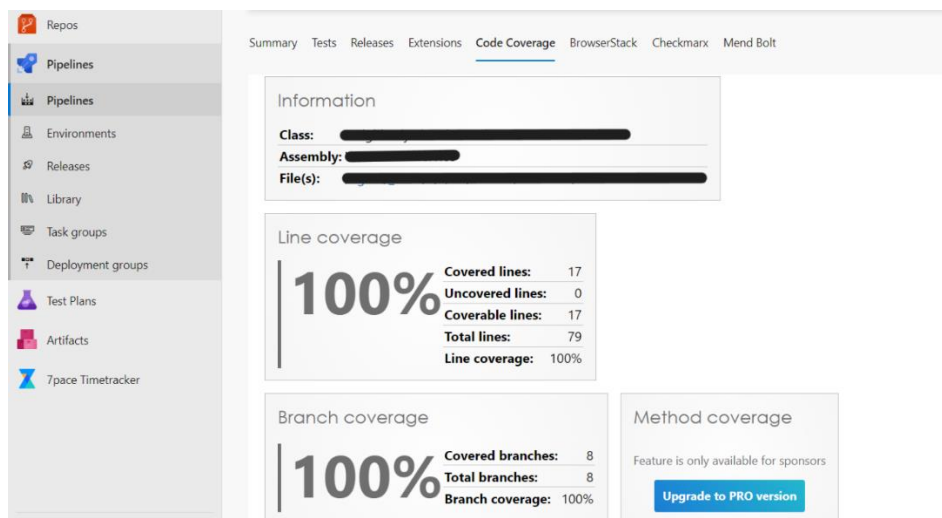
4.1.3 Comparative Analysis 2

Sonar Cloud Quality Gate Result of Test file Written By Developers



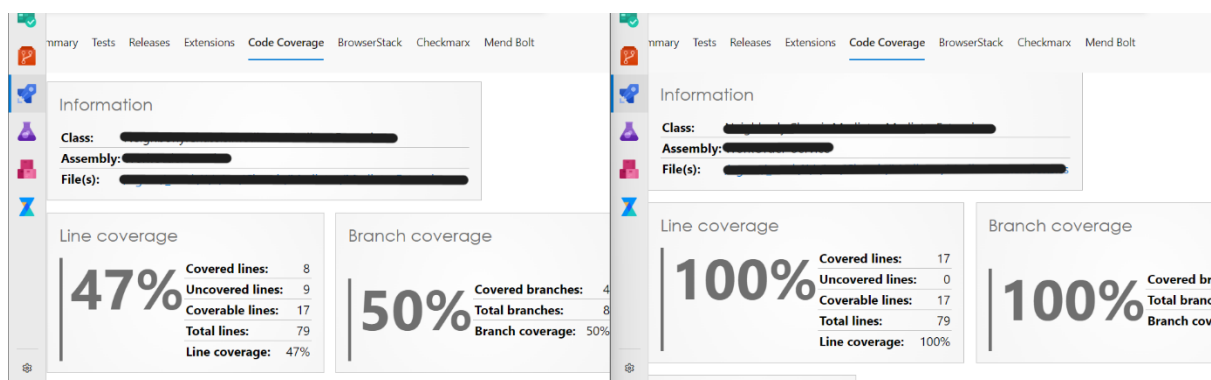
Tests created by developers did not cover some methods and different scenarios

Sonar Cloud Quality Gate Result of Test file Generated by Github Copilot



Github Copilot worked very well with FILE 3 and covered all the methods present in it, generated test cases for various scenarios and also handled exceptions. As a result it improved the overall quality of the test suit

COMPARISON



Side by side comparison of the test files created by developers with the one generated by copilot

Appendix IV :

Working in the Organization



Figure 8 :Presentation to NBLY Leadership Team



Figure 9: Discussion with colleagues



Figure 10 &9 : Neighborly Team Engagement Activities