Silver Console – Hospitality Business Portals and Automated Testing

N SHASHANK MADHAV[1], T. MAMATHA[2]
[1] *UG Scholar, CSE Department, SNIST, Hyderabad, Telangana, India*
[2] *Assistant Professor, Department of Computer Science and Engineering, SNIST, Hyderabad, Telangana, India*

Abstract— Silver Console is a Software-As-a-Service (SaaS) based product which fetches data through timely batch jobs from various data sources for reporting purposes. Silver Console majorly focuses on the following: Sales reports across single or multiple stores, Easy to use with graphical dashboard and deep drill down capability, Stay informed with easy-to-use auto alerts and communication tools, Keep track of detailed sales trends in the form of intuitive graphs and charts and interactive drilldowns, all of which make the complex seem simple, See how your sales are doing against goal or past performance, for single or multiple sites, See sales mix information right down to the employees that are selling the items you want to push, Supports 4 different POS systems (14 deprecated): Aloha (Pulse RT), Silver / SPRE / Silver Essentials (SBO), CBS, Treatware. Supports 10000+ sites/stores and 8000+ brands/clients

I. INTRODUCTION

The hospitality business is one of the major business sectors. Every day there are new restaurants, fast-food chains, drive-throughs etc. being opened as the population and the demand to eat out increases. Businesses are becoming more and more consumer centric and with that a lot of data is generated by a business. Today we will look at how the product Silver Console provides various tools to collect and report data to satisfy reporting requirements at all levels of a given business/organization.

Existing systems provide services that concentrate on satisfying customer requirements only at a single level of the business structure. They provide reporting tools either at the employee level or the store level or at the corporate level of the business but not a complete service that satisfies the business requirements at every level. Even if a service was provided to meet requirements at each business level, the services may not be very closely integrated and therefore may not work well in tandem with each other. However, with Silver Console this is not the case. We provide a closely knit service of 4 portals that take care of all the reporting requirements that an organization might have at all levels of the organization. These portals are interrelated to each other and work very closely to achieve concurrent reporting of large amounts of data to many clients. Silver Console also provides a very high level of customizability of what is being reported and how it is reported.

II. LITERATURE SURVEY

.NET and .NET Framework [2]
.NET is a developer platform made up of tools, programming languages, and libraries for building many different types of applications. There are various implementations of .NET. Each implementation allows .NET code to execute in different places—Linux, macOS, Windows, iOS, Android, and many more.

.NET Framework is the original implementation of .NET. It supports running websites, services, desktop apps, and more on Windows. .NET is a cross-platform implementation for running websites, services, and console apps on Windows, Linux, and macOS. .NET is open source on GitHub. .NET was previously called .NET Core. Xamarin/Mono is a .NET implementation for running apps on all the major mobile operating systems, including iOS and Android. .NET Standard is a formal specification of the APIs that are common across .NET implementations. This allows the same code and libraries to run on different implementations.

C# [3]

C# (pronounced "See Sharp") is a modern, object-oriented, and type-safe programming language. C# enables developers to build many types of secure and robust applications that run in .NET. C# has its roots in the C family of languages and will be immediately familiar to C, C++, Java, and JavaScript programmers.

Java

Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc.

Selenium [4]

Selenium is a free (open source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc. to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

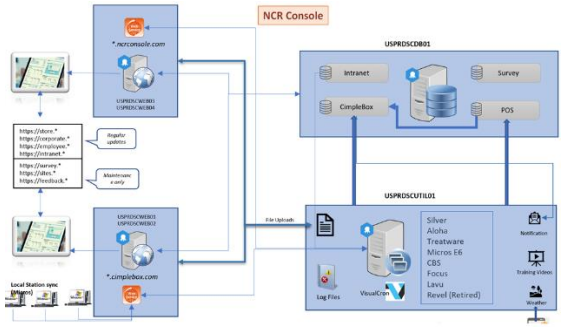
TestNG [5]

TestNG is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired by JUnit which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make end-to-end testing easy.

GitHub Actions [6]

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository or deploy merged pull requests to production.

III. NCR CONSOLE SYSTEM ARCHITECTURE



The main components of the architecture are Point Of Sale (POS) devices, databases and our main product-NCR console. Sales data is populated through POS devices. This data is then fetched by batch jobs and then loaded into two main databases which are the cimplebox database and the POS database. From this raw sales data various sales reports, graphical dashboards, summary reports, sales trends, intuitive charts, and interactive drilldowns etc. are generated and are reported to the client through the store, corporate, employee, and intranet portals

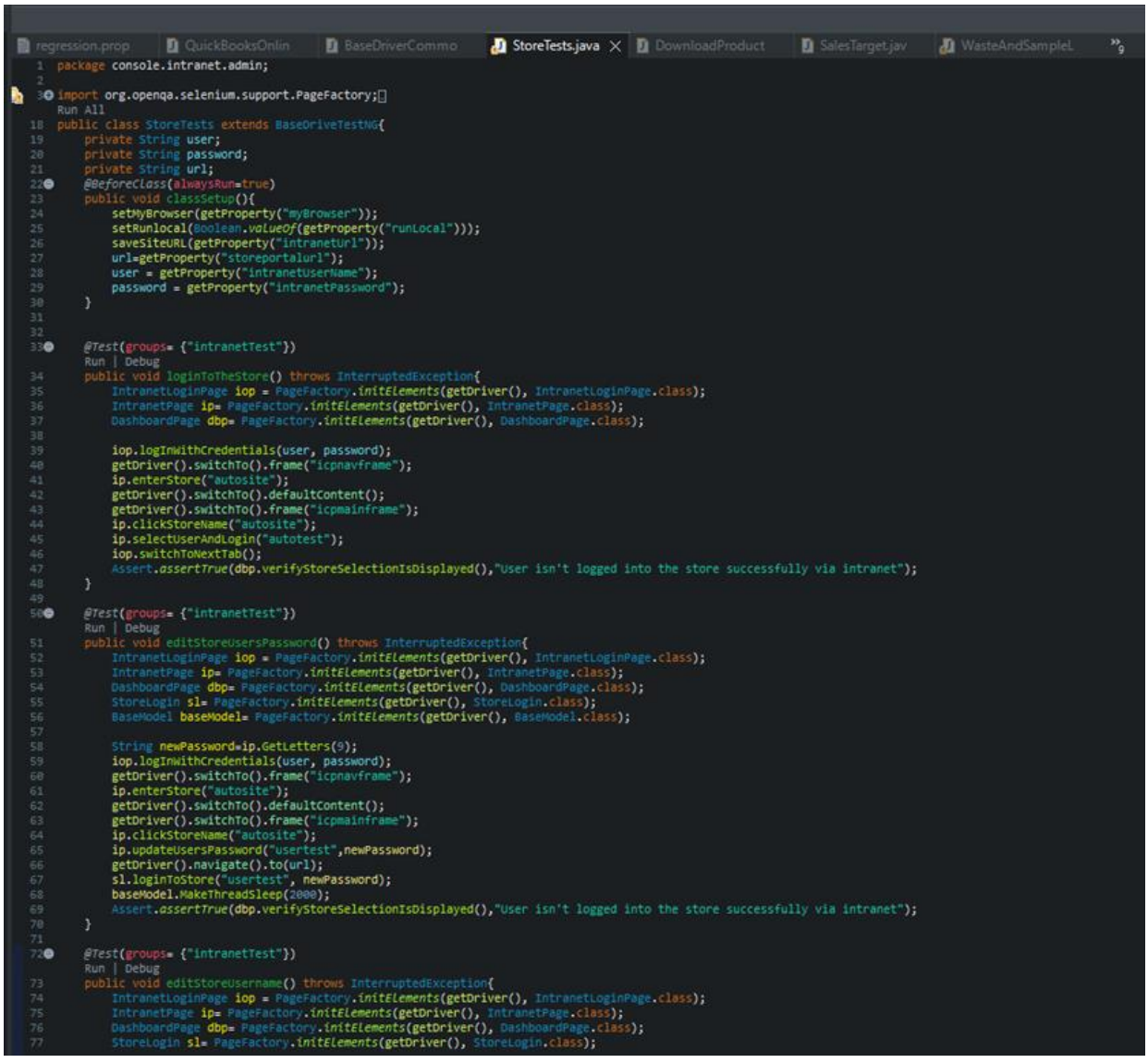
IV. AUTOMATION SYSTEM IMPLEMENTATION

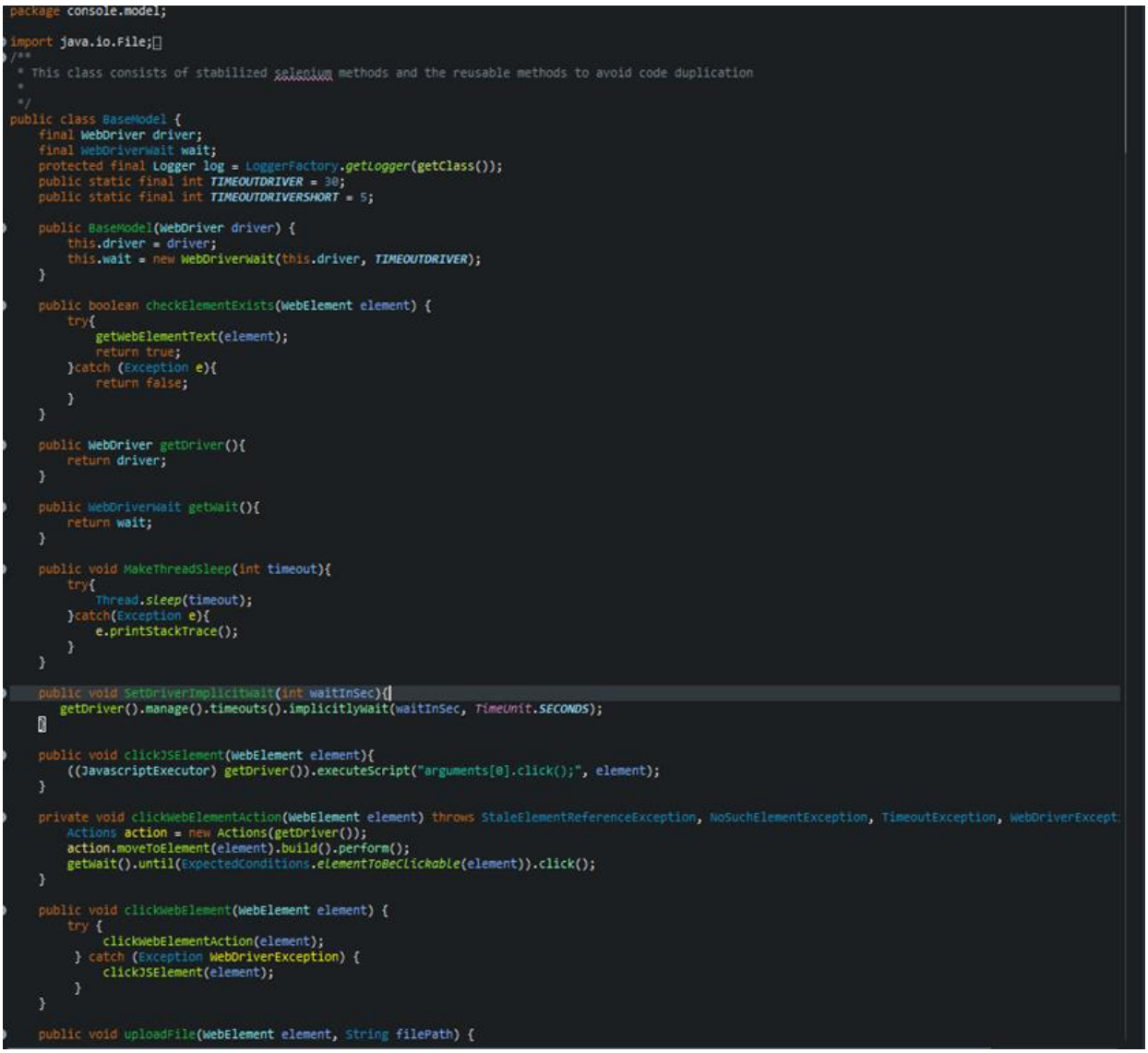
Silver Console has a separate sub-project for its automated testing. Automated testing for silver console is achieved using Selenium with Java along with various other frameworks like TestNG, Junit, log4j, Gradle for the purposes of test suite automation, result logging and build automation. All the project and test dependencies are mentioned in the build. gradle file which installs all the required dependencies and resources at run time and also sets up the environment for the test runs.

The test cases are categorized into smoke tests, regression tests and functional tests. As part of my internship, I have mostly worked on smoke tests and regression tests. The main framework used on top of selenium is TestNg. This framework makes building test suites much easier and also displays useful test run details that produce useful insights about why a test is not behaving as intended or why it is failing. It also allows us to use special methods that allow us to customize what runs before and after each test cases, method, and class. It provides grouping mechanisms which allows us to group testcases to later run them all

at the same time. It is also a very useful tool when it comes to generating reports of all the test runs and how many tests failed and how many passes. I have also used page objects to achieve higher level of abstraction and test reusability. The framework logj4 is used to create logs of the tests that are run and these logs are then saved in a specified location

One other major component of our automation project is CI/CD integration using GitHub Actions. You can configure a GitHub Actions workflow to be triggered when an event occurs in your repository, such as a pull request being opened, or an issue being created. Your workflow contains one or more jobs which can run in sequential order or in parallel. Each job will run inside its own virtual machine runner, or inside a container, and has one or more steps that either run a script that you define or run an action, which is a reusable extension that can simplify your workflow. I have used GitHub actions workflows to make it so that the test suite (specified group of testcases) can be triggered remotely using a curl command [7]. These tests, when triggered, are run on a virtual machine called GitHub runner. This runner can be configured in various ways such as its operating system, the versions of different software, etc. Using the workflow file and GitHub runners I have also implemented an automated slack channel test reporting system which automatically generates an emailable test report and sends it to a slack channel of our choice

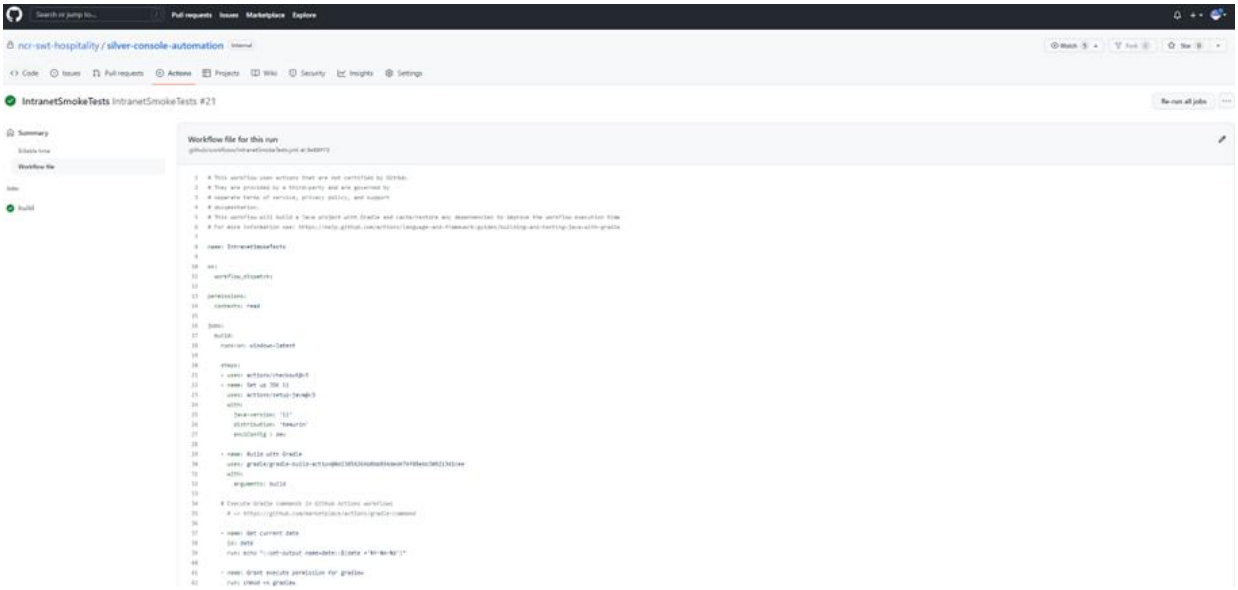


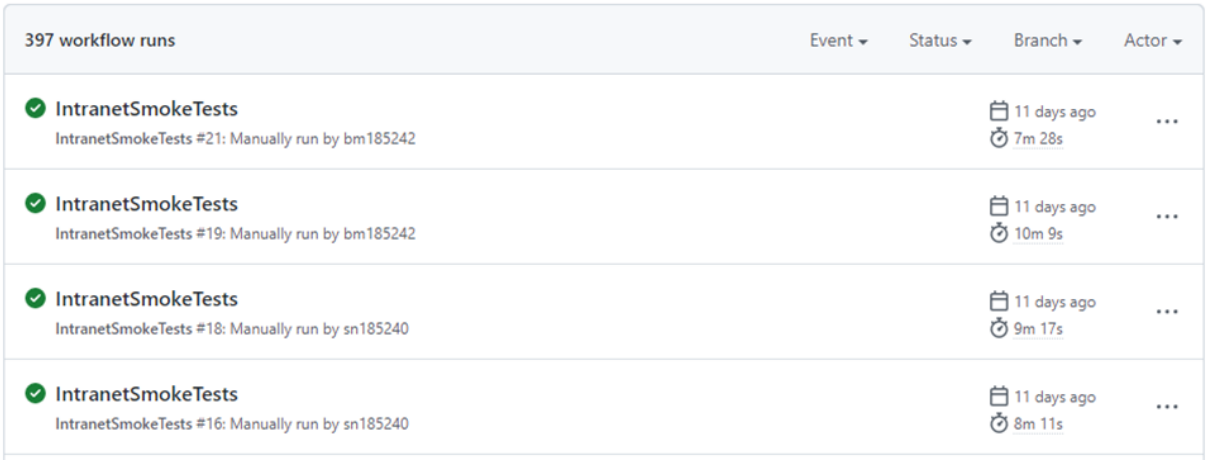


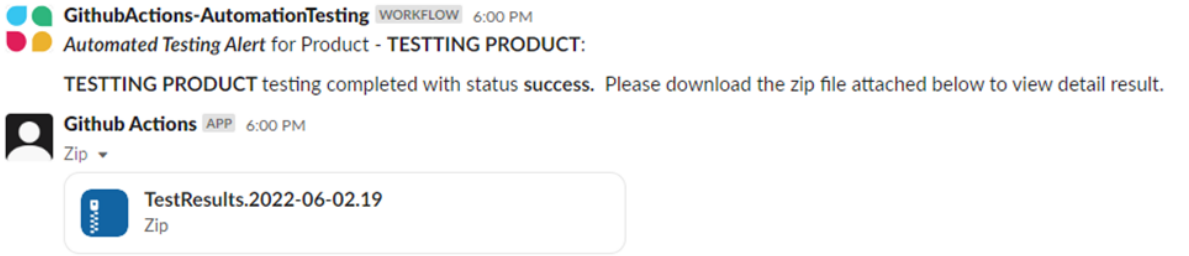
```java
package console.model;

import java.io.File;
/**
 * This class consists of stabilized selenium methods and the reusable methods to avoid code duplication
 *
 */
public class BaseModel {
    final WebDriver driver;
    final WebDriverWait wait;
    protected final Logger log = LoggerFactory.getLogger(getClass());
    public static final int TIMEOUTDRIVER = 30;
    public static final int TIMEOUTDRIVERSHORT = 5;

    public BaseModel(WebDriver driver) {
        this.driver = driver;
        this.wait = new WebDriverWait(this.driver, TIMEOUTDRIVER);
    }

    public boolean checkElementExists(WebElement element) {
        try{
            getWebElementText(element);
            return true;
        }catch (Exception e){
            return false;
        }
    }

    public WebDriver getDriver(){
        return driver;
    }

    public WebDriverWait getWait(){
        return wait;
    }

    public void MakeThreadsSleep(int timeout){
        try{
            Thread.sleep(timeout);
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    public void SetDriverImplicitWait(int waitInSec){
       getDriver().manage().timeouts().implicitlyWait(waitInSec, TimeUnit.SECONDS);
    }

    public void clickJSElement(WebElement element){
        ((JavascriptExecutor) getDriver()).executeScript("arguments[0].click();", element);
    }

    private void clickWebElementAction(WebElement element) throws StaleElementReferenceException, NoSuchElementException, TimeoutException, WebDriverExcept
        Actions action = new Actions(getDriver());
        action.moveToElement(element).build().perform();
        getWait().until(ExpectedConditions.elementToBeClickable(element)).click();
    }

    public void clickWebElement(WebElement element) {
        try {
             clickWebElementAction(element);
         } catch (Exception WebDriverException) {
             clickJSElement(element);
         }
    }

    public void uploadFile(WebElement element, String filePath) {
```

Figures: Test case automation using Selenium, Java, TestNG and Gradle

I have used Selenium and Java along with other frameworks like TestNG, Gradle, log4j to automate testcases and to produce reports





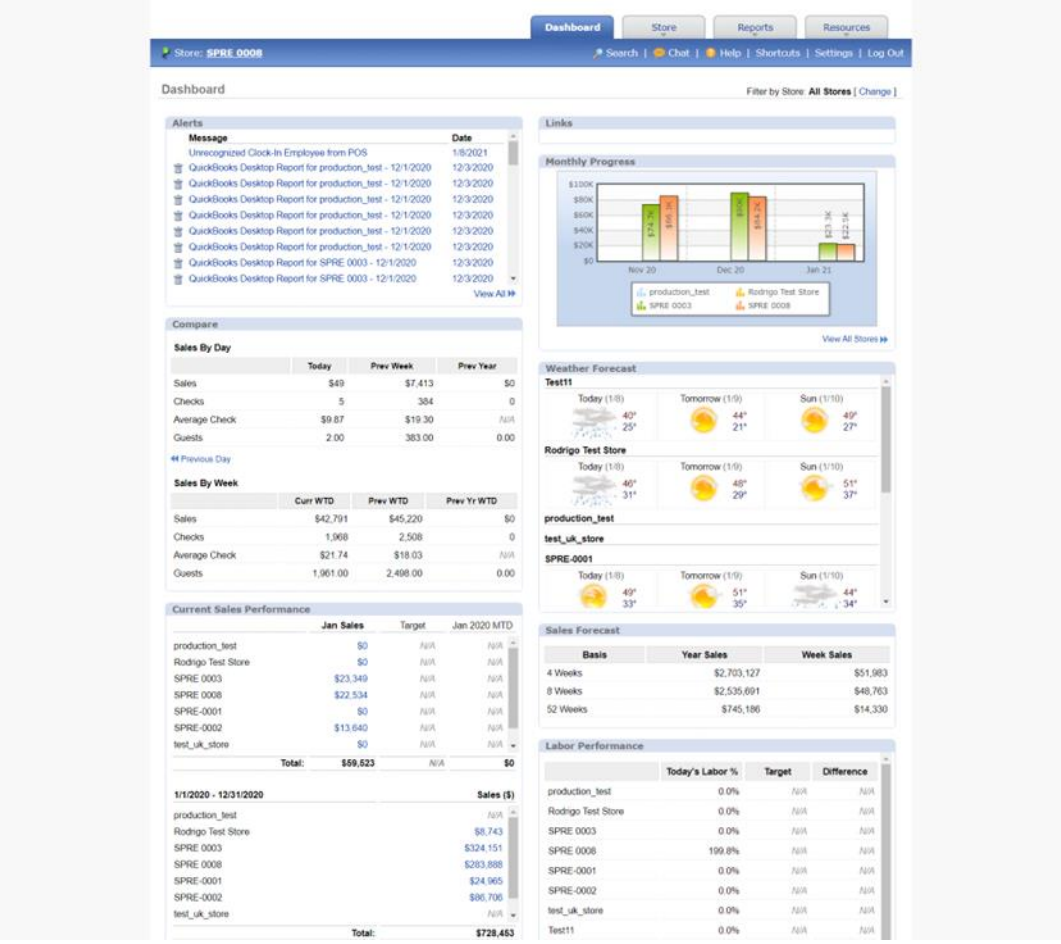


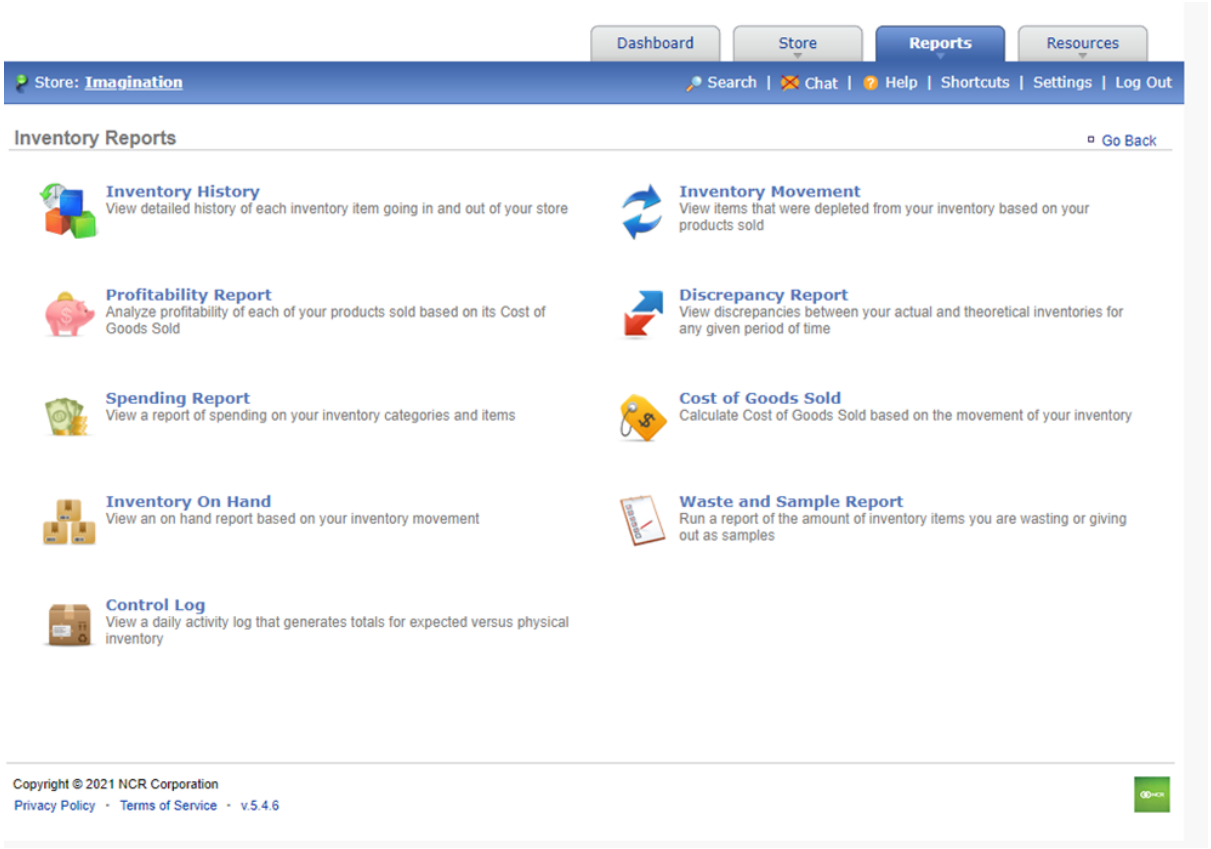
Figures: CI/CD integration using GitHub actions

I have used GitHub actions to implement CI/CD pipeline for the automation project and also to automate the sending of reports to slack channel after test suite run

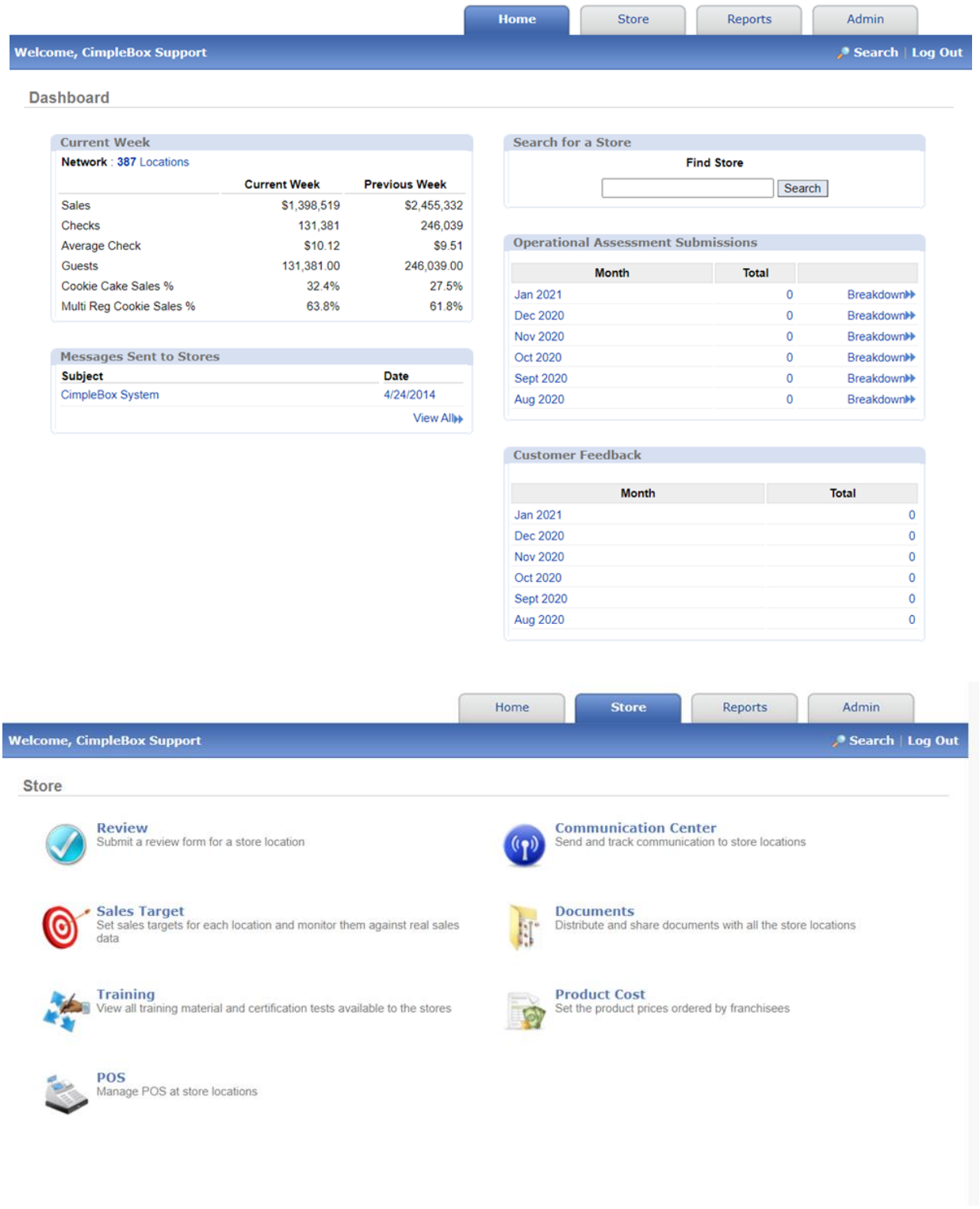
V. PRODUCT SHOWCASE

Store Portal

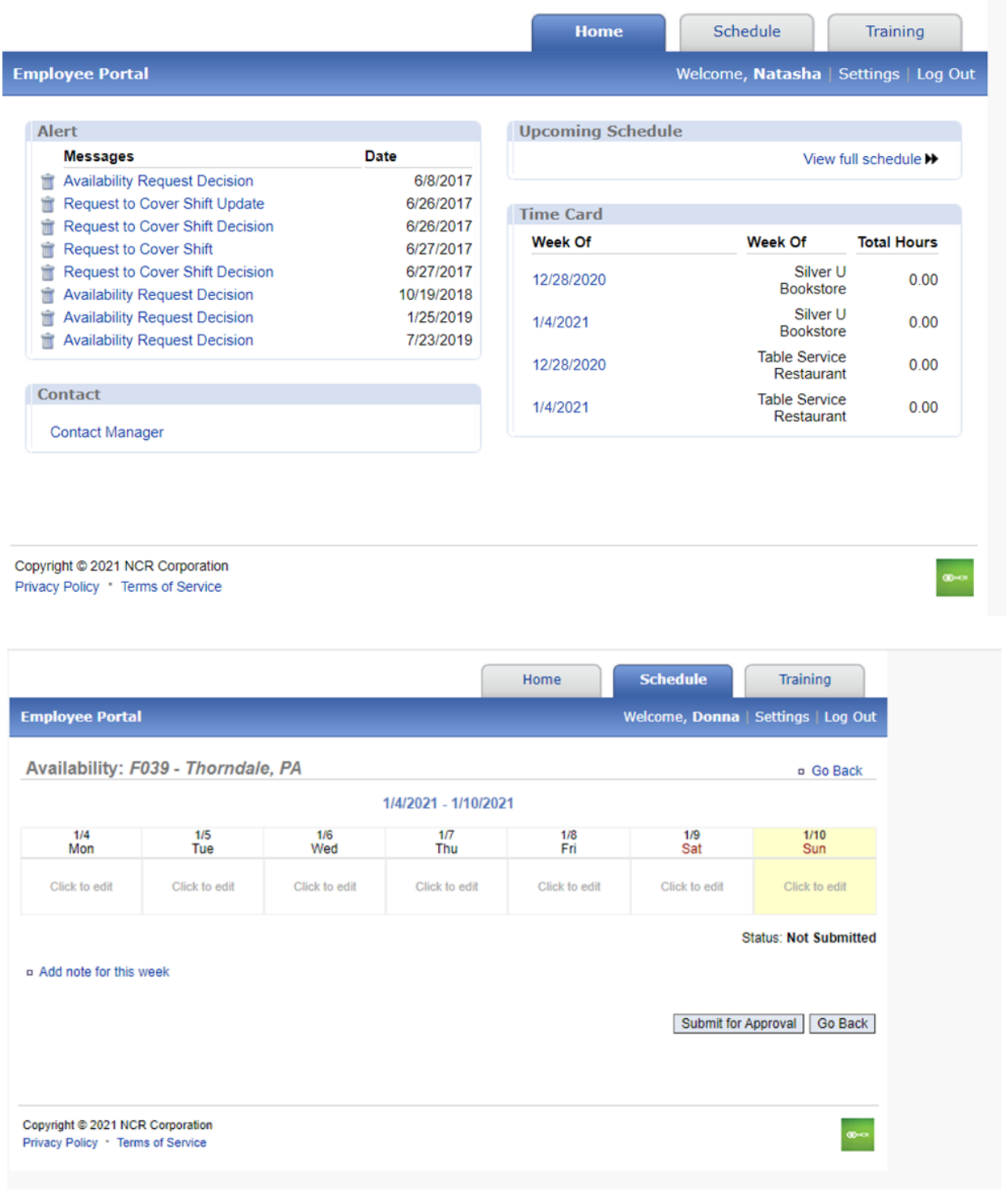

Dashboard
Store
Reports
Resources
Store: SPRE 0008
Search | Chat | Help | Shortcuts | Settings | Log Out
Dashboard
Filter by Store: All Stores [Change]
Alerts
Message
Date
Unrecognized Clock-In Employee from POS 1/8/2021
QuickBooks Desktop Report for production_test - 12/1/2020 12/3/2020
QuickBooks Desktop Report for SPRE 0003 - 12/1/2020 12/3/2020
QuickBooks Desktop Report for SPRE 0003 - 12/1/2020 12/3/2020
View All
Links
Monthly Progress
$100K
$80K
$60K
$40K
$20K
$0
Nov 20
Dec 20
Jan 21
production_test
Rodrigo Test Store
SPRE 0003
SPRE 0008
View All Stores
Compare
Sales By Day
Today
Prev Week
Prev Year
Sales $49 $7,413 $0
Checks 5 384 0
Average Check $9.87 $19.30 N/A
Guests 2.00 383.00 0.00
Previous Day
Sales By Week
Curr WTD
Prev WTD
Prev Yr WTD
Sales $42,791 $45,220 $0
Checks 1,968 2,508 0
Average Check $21.74 $18.03 N/A
Guests 1,961.00 2,498.00 0.00
Weather Forecast
Test11
Today (1/8) 40° 25°
Tomorrow (1/9) 44° 21°
Sun (1/10) 49° 27°
Rodrigo Test Store
Today (1/8) 40° 31°
Tomorrow (1/9) 48° 29°
Sun (1/10) 51° 37°
production_test
test_uk_store
SPRE-0001
Today (1/8) 49° 33°
Tomorrow (1/9) 51° 35°
Sun (1/10) 44° 34°
Current Sales Performance
Jan Sales
Target
Jan 2020 MTD
production_test $0 N/A N/A
Rodrigo Test Store $0 N/A N/A
SPRE 0003 $23,349 N/A N/A
SPRE 0008 $22,534 N/A N/A
SPRE-0001 $0 N/A N/A
SPRE-0002 $13,640 N/A N/A
test_uk_store $0 N/A N/A
Total: $59,523 N/A $0
1/1/2020 - 12/31/2020
Sales ($)
production_test N/A
Rodrigo Test Store $8,743
SPRE 0003 $324,151
SPRE 0008 $283,888
SPRE-0001 $24,965
SPRE-0002 $86,706
test_uk_store N/A
Total: $728,453
Sales Forecast
Basis
Year Sales
Week Sales
4 Weeks $2,703,127 $51,983
8 Weeks $2,535,691 $48,763
52 Weeks $745,186 $14,330
Labor Performance
Today's Labor %
Target
Difference
production_test 0.0% N/A N/A
Rodrigo Test Store 0.0% N/A N/A
SPRE 0003 0.0% N/A N/A
SPRE 0008 199.8% N/A N/A
SPRE-0001 0.0% N/A N/A
SPRE-0002 0.0% N/A N/A
test_uk_store 0.0% N/A N/A
Test11 0.0% N/A N/A



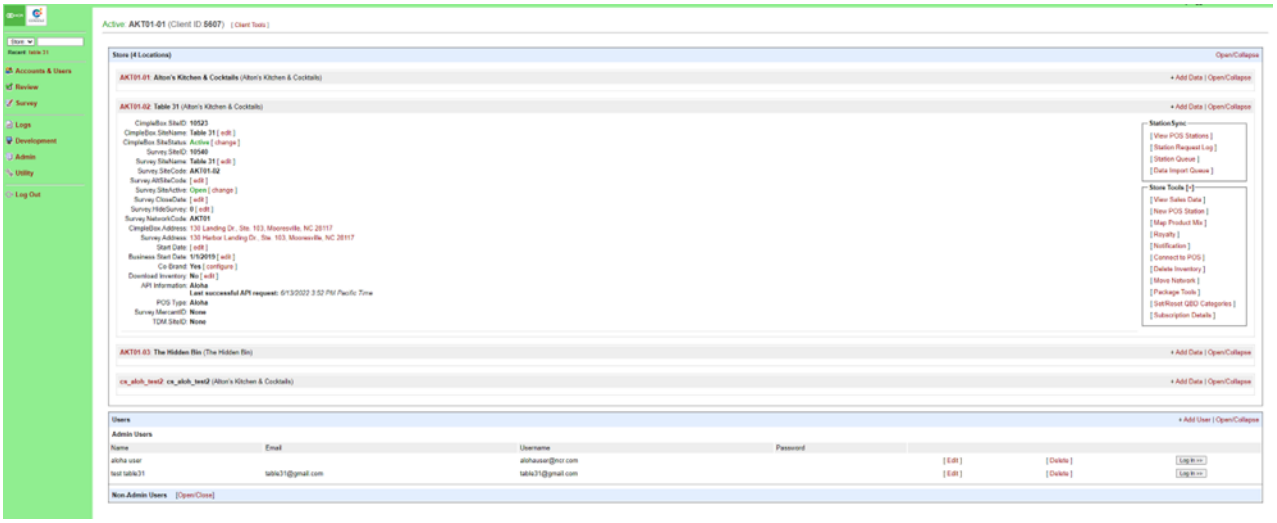
Corporate Portal



Employee Portal



Intranet Portal



VI. CONCLUSION

Moving forward there is only growth to be seen in the hospitality sector. As the customer data increases so does the need for reporting tools such as Silver Console. Trends suggest that people are more willing than ever to pay for services such as hospitality and entertainment, therefore using reporting services such as silver console will allow clients to capitalize on this increasing demand and to maximize the financial returns.

For future releases we plan to work on the front end and UI to make the portals more user friendly in general.

VII. REFERENCES

[1] https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework

[2] https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[3] https://www.browserstack.com/selenium

[4] https://testng.org/doc/

[5] https://github.com/features/actions

[6] https://www.geeksforgeeks.org/curl-command-in-linux-with-examples/

[7] Winkler D., Biffl S., Östreicher T.: „Test-Driven Automation – Adopting Test-First Development to Improve Automation Systems Engineering Processes“, Proc. of 16th EuroSPI, Industrial Track, Madrid, 2009.

[8] Broy M., Jonsson B., Katoen J-P., Leucker M.: Model-Based Testing of Reactive Systems: Advanced Lectures, Springer, 2005