WAM – Windows Automation Manager

by

Ryan Adams, Andrew Holzman, Alex Marshall

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Ryan Adams	*April 15, 2019*
Ryan Adams	Date
Andrew Holzman	*April 15, 2019*
Andrew Holzman	Date
Alex Marshall	*April 15, 2019*
Alex Marshall	Date
Abdou Fall	*April 15, 2019*
Abdou Fall, Faculty Advisor	Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2019

Prepared by
Ryan Adams, Andrew Holzman and Alex Marshall

Students of
University of Cincinnati
College of Education, Criminal Justice, and Human Services
School of Information Technology

April 15th, 2019

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Abstract - WAM (Windows Automation Manager)

In modern business/enterprise environments, automation has become the driving force to improve products, workflow, and business practices. Managing where, when, and how these automated jobs are running can become an unwieldy task for many. Windows Task Scheduler, SQL Server Agent, SQL Server Report/Integration Services, and numerous other automation tasks all have their own interfaces. WAM (Windows Automation Manager) is a web-based automation application that allows users to quickly access scheduled jobs, log information, and access a job creation interface to perform a variety of automation tasks. Through it's simplistic and to the streamlined design, WAM will allow system and network administrators to spend less time monitoring and scheduling tasks, and more time on projects and maintenance.

1. Introduction

1. 1 Problem Statement

In modern business/enterprise environments, automation has become a driving force to improve products, workflow, and business practices. A recent study by McKinsey and Company states "...about 60 percent of occupations could have 30 percent or more of their constituent activities automated. In other words, automation is likely to change the vast majority of occupations—at least to some degree—which will necessitate significant job redefinition and a transformation of business processes. (1, sc. 2). Managing where, when, and how these automated jobs are running will become more and more important, and even now that task can be an unwieldy. Windows Task Scheduler, SQL Server Agent, SQL Server Report/Integration Services, and numerous other automation tasks all have their own interfaces and tools and metrics. These management applications require an employee to constantly flip between applications to schedule, maintain, and view log data for all of their automated jobs. As system administrators automate more of their tasks, they continually add more to their maintenance workload. The whole goal of automation is efficiency - why not reduce the time and effort to maintain these jobs as well?

1.2 Description of Solution

Our product is a centrally managed application that allows for quick access to scheduled jobs, log information, and a job creation interface for a variety of automation tasks. The front end client will allow users to see all of their current jobs, what program they're running in, and when they're scheduled. They will be able to see where the jobs are storing their logs and access them. The backend server will interface with different task scheduling systems on the internal network to pull their job info, perform job creation and scheduling operations, and include a framework to schedule, and maintain custom jobs (batch scripts, vbs, etc). As job's are maintained and ran, the application will collect data and provide reporting dashboards for at-a-glance review of job history and job performance statistics.

1.3 User Profile

1.3.1 Project Title

WAM (Windows Automation Manager)

1.3.2 Potential Users

Business and Enterprise users like software developers or network administrators who are looking to run numerous automated tasks in one interface/system

1.3.3 Software and Interface Experience

- The software is designed as a web-based application that is capable of running across multiple platforms
- No special pre-install is required in order to run the application.
- Very little experience will be needed in order to operate the application. Detailed instructions will be posted within FAQ page

1.3.4 Experience with Similar Applications

- Windows Task Scheduler is only available within Windows operating systems and is limited to run only within Windows based applications. WTS will often experience permission issues and does not indicate there's been an error when it fails.
- SQL Server Agent or SQL Server Report/Integration Services are limited to scheduled tasks that can run only within SQL server

1.3.5 Task Experience

- User will need to have basic website browsing experience and understand how to schedule or setup a scheduled task

1.3.6 Frequency of Use

- Application may be used on daily basis. The frequency may vary from user to user. It depends on specialty of the user

1.3.7 Key Interface Design Requirements That the Profile Suggests

- User friendly web interface
- Status and progress of each task will be shown within the user interface
- Actual jobs will run on the server side
- Results data will be displayed once jobs finish
- Easy navigation and control

1.4 Use Case Diagram

Figure 1: W.A.M. Use Case Diagram, outlining how we envision our two main users (end users and administrators) to operate inside of WAM

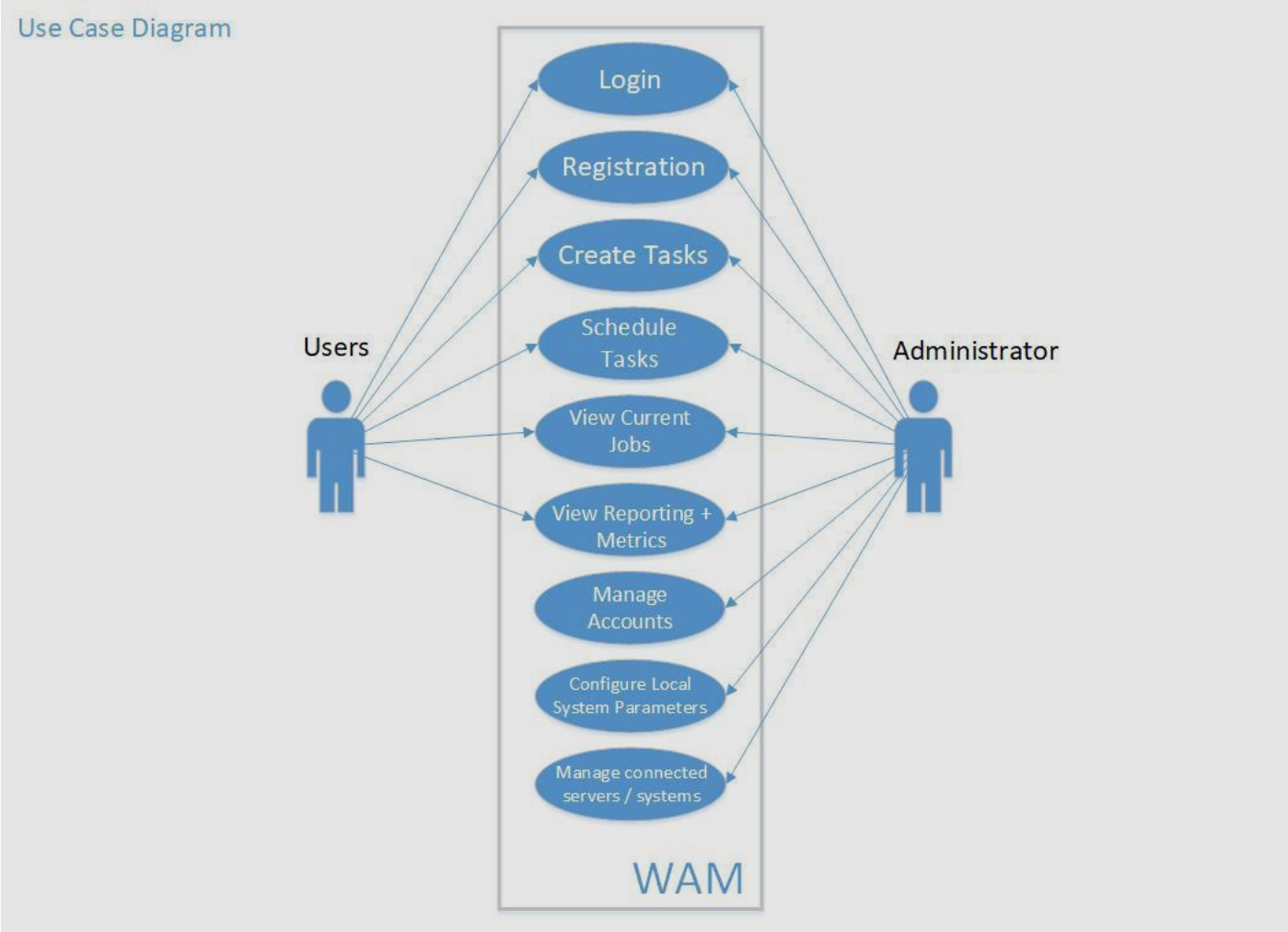

Figure 1: Use Case Diagram

2. Project Management

Table 1: Break down of the cost estimate for WAM. As it is a tool marketed at companies with existing infrastructure and software, we assume the cost of the two software requirements (Windows Server to host, and then Microsoft SQL Server) are already owned.

2.1 Budget

WAM Cost Estimate				
Line Items	**Hours/Qty Necessary**	**Rate/Price**	**Notes**	**Total**
1. Development Effort				
Labor	288	$40	18 Hrs Per Week for 16 Weeks	**$11,520.00**
2. Software				
Windows Server 2XXX	1	0	Assumed Owned (Sandbox)	**$0.00**
Microsoft SQL Server	1	0	Assumed Owned (Sandbox)	**$0.00**
Total				**$11,520.00**

Table 1: Project Budget

2.2 Objectives / Deliverables

1. **Pre-Project Planning**
   a. Course Overview
   b. Research Proposed Project
2. **Research Milestone**
   a. Team Building
   b. Team contract
3. **Design Milestone**
   a. Project Management
   b. Building a Budget
   c. Building a Gantt chart
   d. Finalizing Problem Statement
4. **Implementation Milestone**
   a. Planning Tech Expo
   b. Abstract
   c. User Profile Report
   d. Use Case Diagram
5. **Test Milestone**
   a. Elevator Speech
   b. Final Report Checklist
   c. Testing methods
6. **Software Deployment Milestone**
   a. Review Requirements
   b. Draft Report
7. **Deliverables Milestone**
   a. Preparation for presentation
8. **Presentation Milestone**
   a. Final Fall Presentations
9. **Testing Milestone**
   a. Begin testing user interface
   b. Begin testing back end / task creator
10. **Abstract Revision**
    a. revise abstract
11. **Google Charts implementation**
    a. implement and test google charts and reporting features
12. **Tech Expo Poster**
    a. Design poster
    b. Submit first draft
    c. Submit final copy
13. **Final Spring Presentation**
    a. Spring Semester Presentations
14. **Tech Expo**
    a. Setup tech expo booth
    b. Present at Tech Expo
15. **Final Paper submission**
    a. Edit and submit final paper

2.3 Project Schedule

Figure 2: Project schedule in the form of a Work Breakdown Structure and Gantt Chart.

Fall 2018:

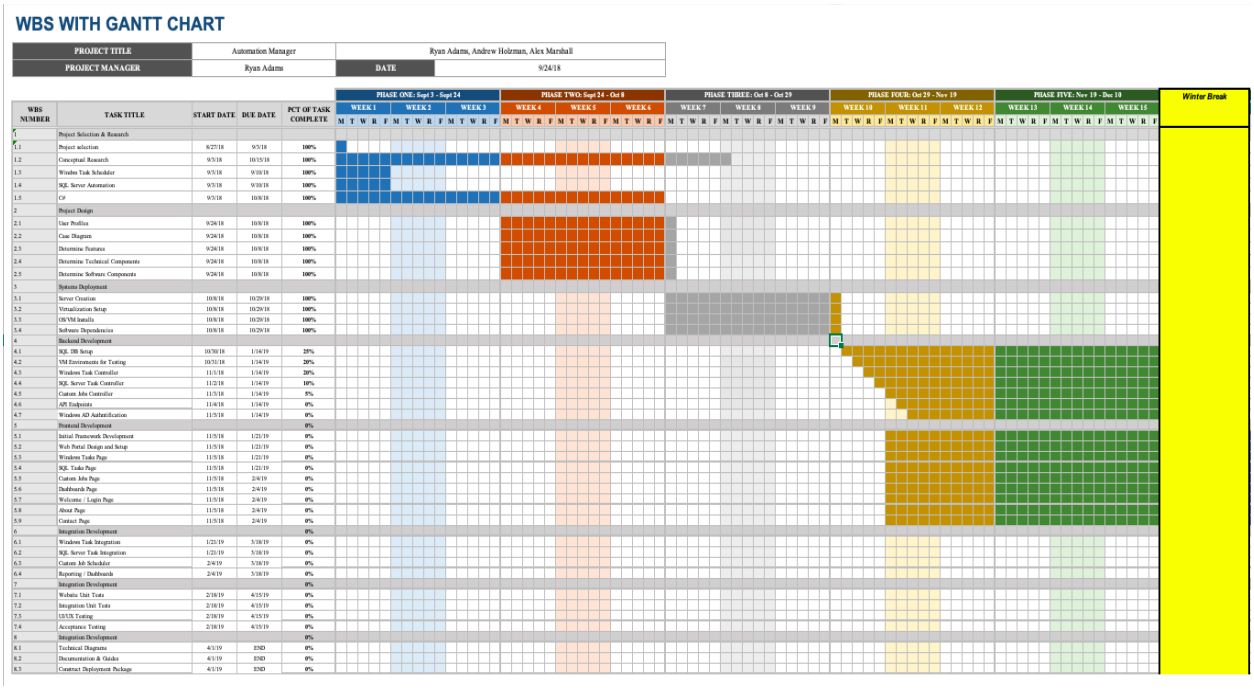

Figure 2: Work Breakdown Structure + Gantt Chart

Figure 3: Project schedule in the form of a Work Breakdown Structure and Gantt Chart.

Spring 2019:

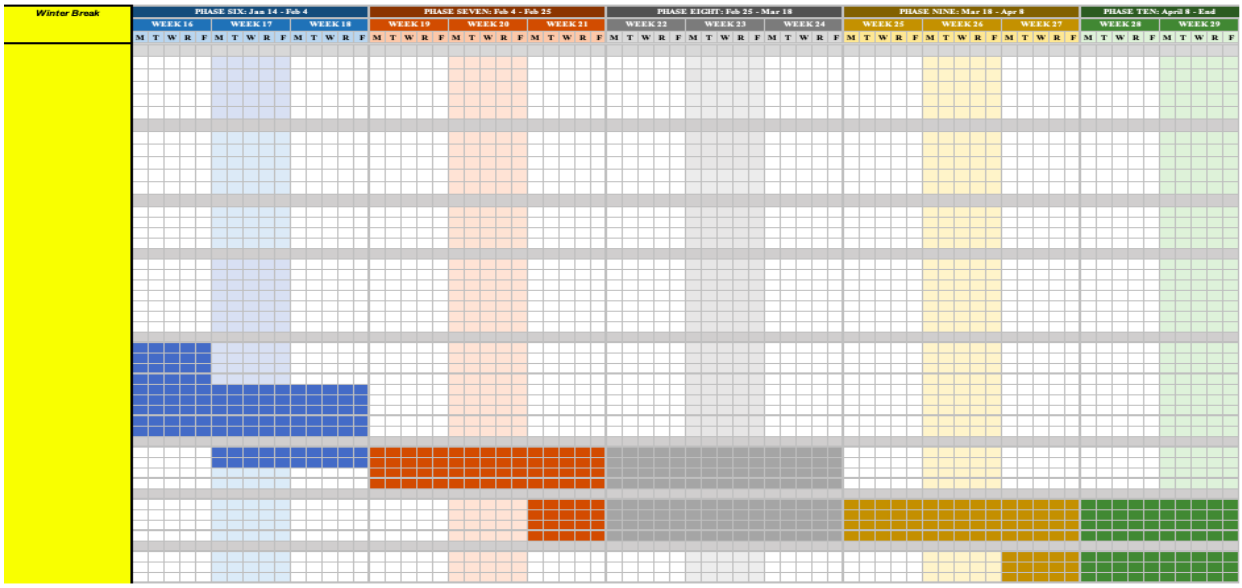

Figure 3: Work Breakdown Structure Continued

2.4 Problems / Challenges

WAM has seen its fair share of challenges as it has progressed through its lifecycle. Initially, we struggled to define a project problem statement until the inspiration for WAM hit us. We then had to determine the best way to balance project responsibilities with technical experience and external factors affecting each team member. This too proved to be challenging as our team was formed just days before the start of the Fall Semester.

3. Technical Elements

The main goal of any piece of software, or any product in general, is to achieve its stated goal efficiently and effectively. We put quite a bit of thought into the best way to design WAM, borrowing quite a bit from the principles of Test-Driven Design. We spent time considering who our target audience was, what kind of applications they'd be used to using, what they want to interface with, and the systems they'd be running this software on. We want the administrators that will rely on WAM to maintain and report on their automation tasks to instantly know how to use the application through a familiar UI and shared-experience design concepts. We put just as much effort into considering the core of the application itself and how it would run on the network, what language and frameworks we'd utilize, and the environments we thought we could best target. What follows is a breakdown of the network environment WAM will run in, an overview of the application and backend layers, and security concerns.

3.1 Network

WAM is an internally hosted web application. It will not be accessible off of the company network and as such will not require hosting on an external cloud environment. Due to the localized nature of the application - managing company automation jobs, having the site hosted internally allows the tool to communicate with machines on the network securely. As WAM targets an enterprise environment as a tool for administrators, hosting internally on a company server ensures that only internal users can access the tool and the automation task, and reduces risk of downtime due to external network pressures.

Figure 4: Illustration of network / architecture diagram of W.A.M.

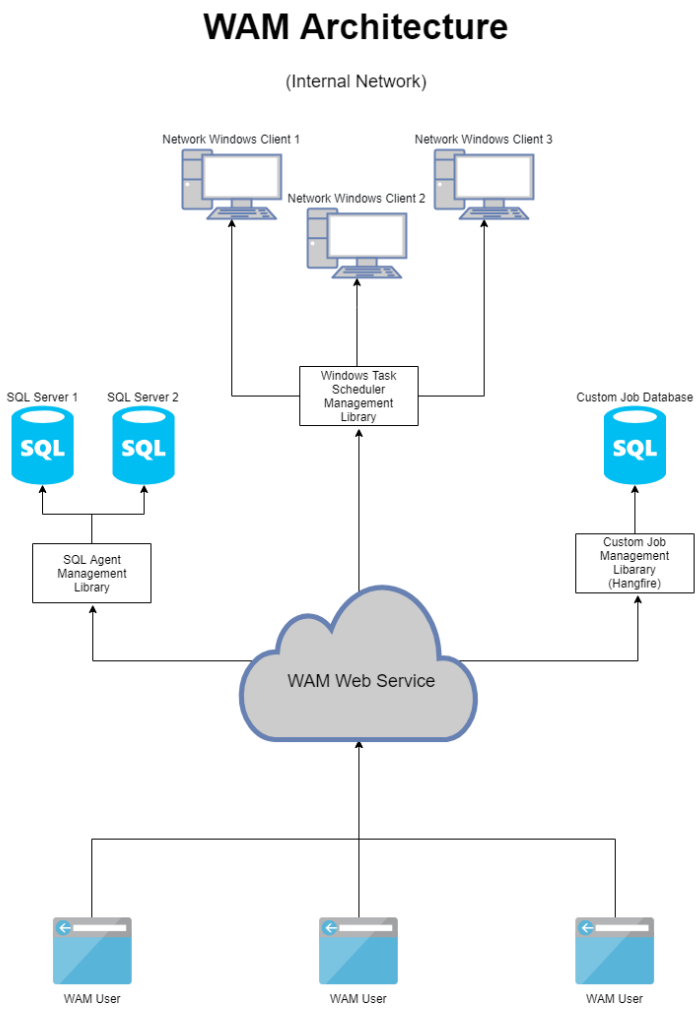

Figure 4: Network / Architecture Diagram

3.2 Application

WAM is written using C# and .NET Core 2.0, utilizing the ASP.NET Core MVC and Entity frameworks. .NET Core is the modernized, open-source redesign of the .NET framework, which allows for more streamlined and lightweight applications that are flexible, scalable and

hardware/software agnostic. The ASP.NET Core MVC framework allows for both the backend web services and the front-end client to be written in a shared language and unified development environment using the Model-View-Controller design pattern. MVC framework allows for consistent interaction between the three components that flows well and is both easily testable and scalable. MVC also allows us to expose an internal API for other applications to utilize WAM's custom job scheduling, logging and metric-based reporting systems. Utilizing Entity framework provides WAM with similarly consistent model and object-based interactions with the backend storage (see more on that below) that are secure and maintainable.

The integrations with automation task providers are written in C# using .NET Core 2.0. These are external class libraries that are decoupled from the main web server, allowing for easy testing and extendibility. Each provider has its own library to perform the data-access operations as dictated by the controllers from the web server. The custom job library (which utilizes the HangFire.IO library to maintain and schedule the custom script tasks defined inside of WAM), and our reporting / dashboards utilize their own class libraries as well.

3.3 Database

WAM stores both its personal application state as well as job history and metrics inside of a Microsoft SQL Server database. Microsoft SQL Server was chosen as it is an industry standard, the interaction design outlined by the Entity Framework, and the fact that our tool integrates with Microsoft SQL Server automation tasks.

3.4 Security

WAM authenticates its users by syncing into the network's Active Directory service. Users will login with their network login, and rights are restricted based on user permissions. The application being hosted internally helps to prevent secure data from being transmitted across the internet, as well as protecting the API that WAM will expose to external resources. The API will be secured using unique API keys issued through the WAM interface. The API key will be validated for each request sent to the system. Utilizing Entity Framework, we minimize access to the database connections being made by application, and restrictions on the model and the SQL queries themselves protect against SQL Injection attacks. In addition to these security measures, we will implement a Web Application Firewall (WAF) to restrict access to the WAM web portal. Finally, we will follow the standards set forth by The National Institute of Standards and Technology (NIST) in regard to Cyber Security

4. Project Visuals

In the following section, we will display selected screenshots from the W.A.M application itself. For brevity and illustration, we are only including the pages for Windows Scheduled Tasks. The remaining two job management sections (SQL Agent Jobs and Custom Jobs) have very similar screens adjusted to show the details for that specific job type.

4.1 Home Page

Figure 5: The home page is the first view of W.A.M. that the user will see. After navigating to the local address used for W.A.M., all users of the application will be greeted with this page. From the home page, users can quickly access a variety of pages including Job/Task Management pages, and Analysis/Reporting + Historical Data pages.

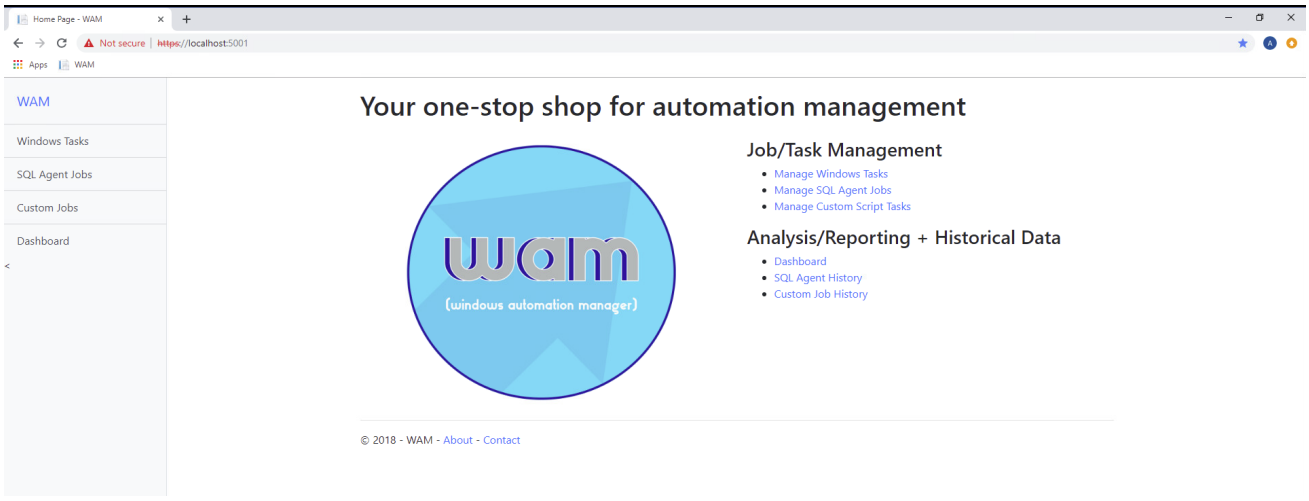

Figure 5: W.A.M. Home Page

4.2 Windows Scheduled Task List

Figure 6: The Windows Scheduled Tasks list page shows all of the Scheduled Tasks that W.A.M is pulling from client machines. From this view we can see the Name, Description, Trigger Type, Action Type, Date Created and Last Run Date of each Task.

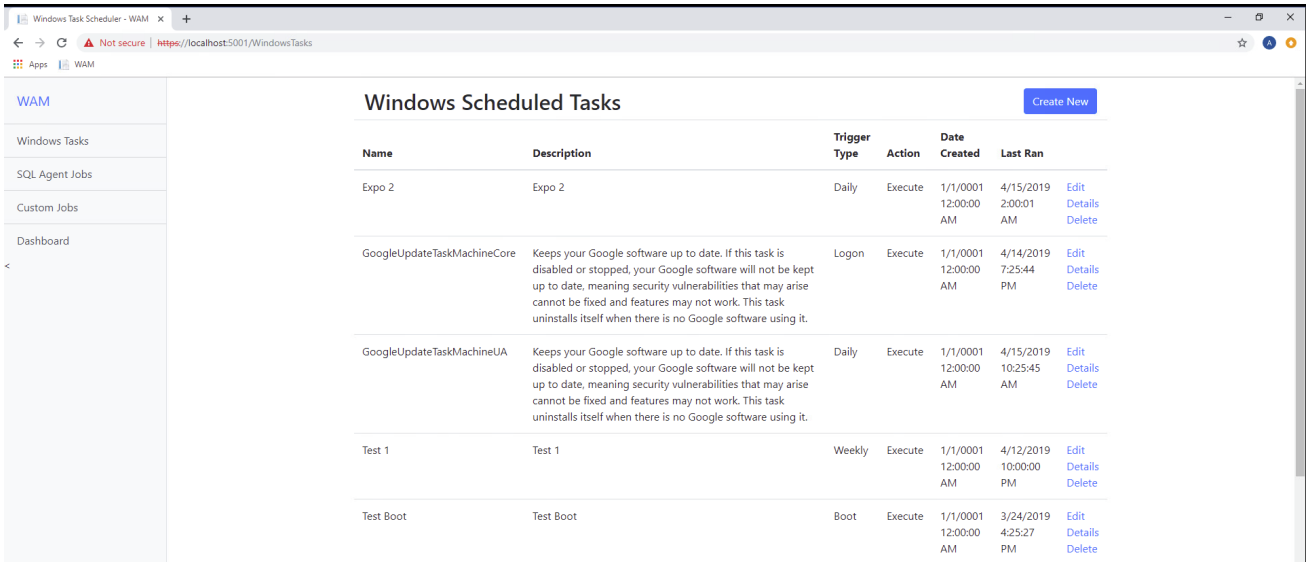

Figure 6: Windows Scheduled Task List

4.3 Windows Scheduled Task Details

Figure 7: The Windows Scheduled Task Details page shows the details of the selected Windows Scheduled Task and includes a button to jump into the Edit page, and then a button to immediately trigger the task in question.

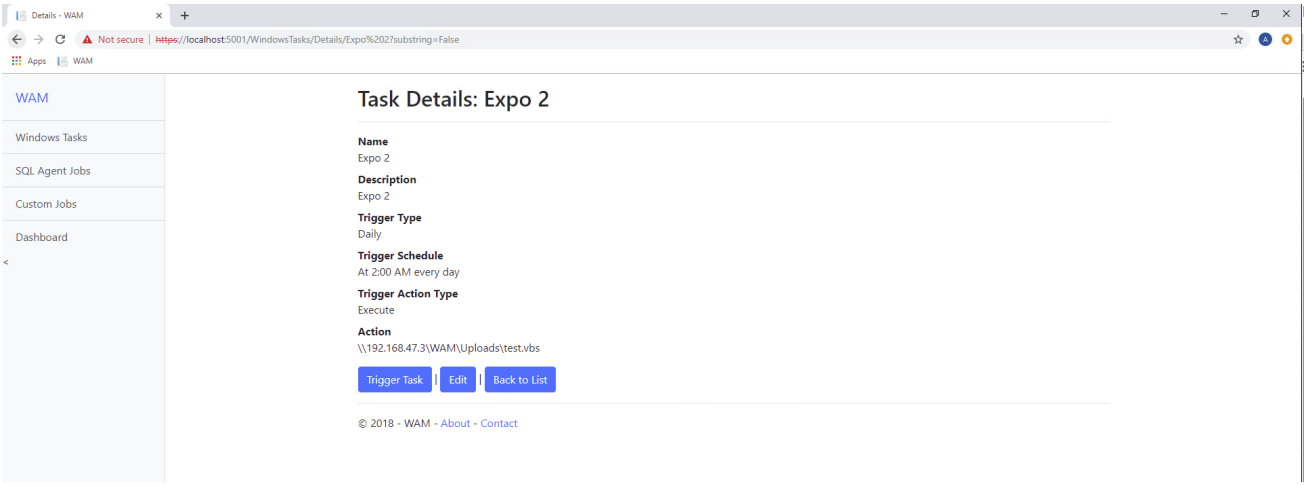

Figure 7: Windows Scheduled Task Details

4.4 Windows Scheduled Task Creation

Figure 8: The Windows Scheduled Task Creation page displays a form that the user will use to actually create the Tasks themselves. The form includes: Description, Trigger Type (Custom, Logon, and Startup), a Trigger String selector to set the schedule, and a file browser to select the file the user would like to execute.

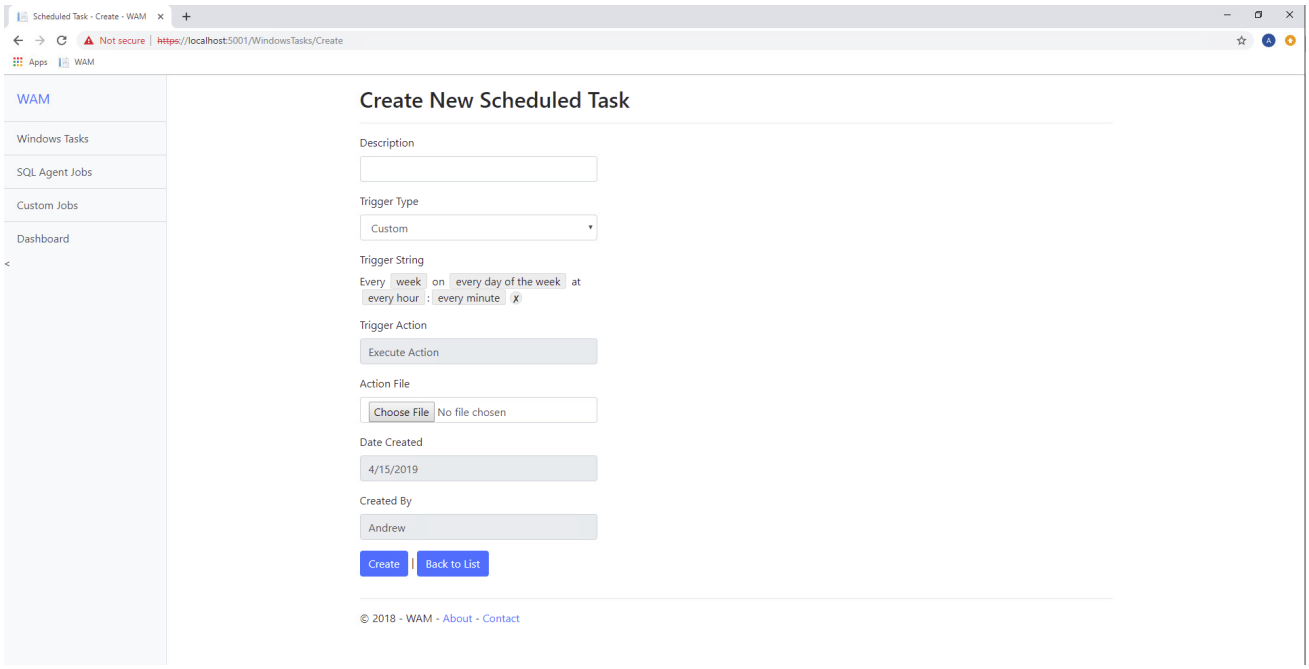

Figure 8: Windows Scheduled Task Creation

4.5 Recent Job History

Figure 9: The Recent Job History page is where you land if you select the “Dashboard”. It displays the last two weeks’ worth of job history detail including Job Name, Job Type, the Date it was ran, the Status that was logged and any Error if present. We also include a quick reference pie chart that displays the ratio of successful jobs to failed jobs, and it includes the number in each section if you mouse over it. The user can then drill down into just SQL Job History or Custom Job History by selecting one of the two buttons.

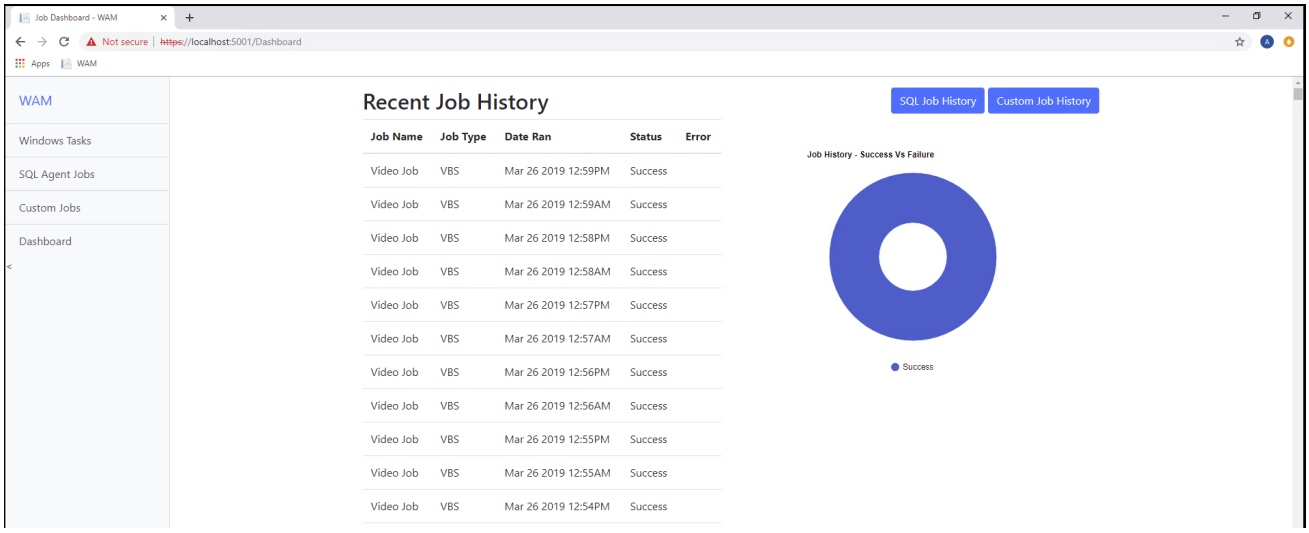

Figure 9: Recent Job History

4.6 SQL Job History

Figure 10: The SQL Job History page displays the last two weeks' worth of job history detail including Job Name, Job Type, the Date it was ran, the Status that was logged and any Error if present for only the SQL Agent Jobs. We also include a quick reference pie chart that displays the ratio of successful jobs to failed jobs, and it includes the number in each section if you mouse over it. The user can then drill down into just Successful and Failed jobs by selecting one of the two buttons.

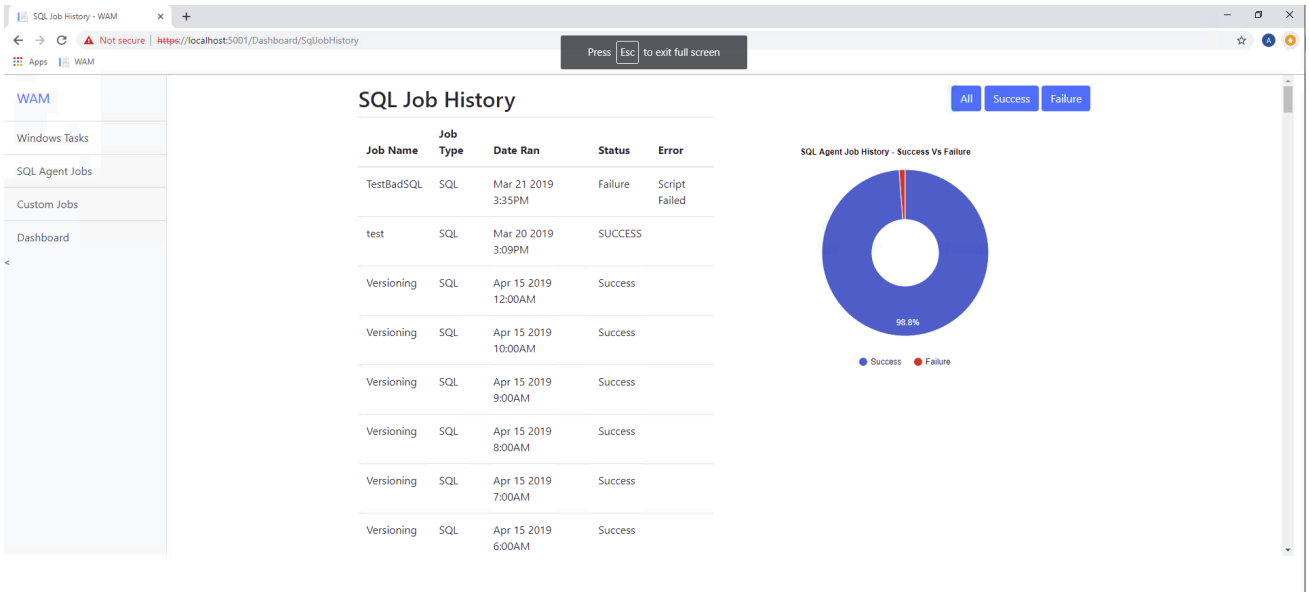

Figure 10: SQL Job History

4.7 Custom Job History

Figure 11: The Custom History page displays the last two weeks' worth of job history detail including Job Name, Job Type, the Date it was ran, the Status that was logged and any Error if present for only the Custom Jobs. We also include a quick reference pie chart that displays the ratio of successful jobs to failed jobs, and it includes the number in each section if you mouse over it. The user can then drill down into just Successful and Failed jobs by selecting one of the two buttons.

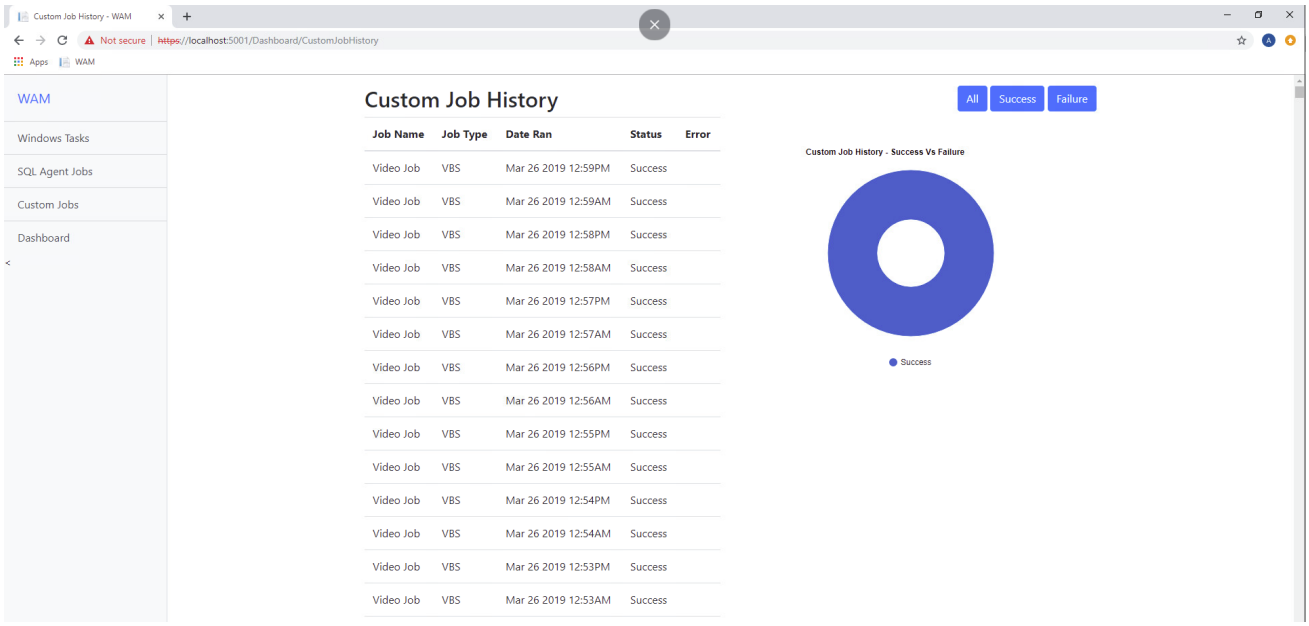

Figure 11: Custom Job History

5. Testing Plan

5.1 Overview

The purpose of this section is to provide details of test plan and strategy for WAM (Windows Automation Manager). This section details our testing methodology and the procedures we have implemented to achieve the desired outcome of the project.

The following team members will find this section beneficial:

- Project Manager
- Front End Developer
- Back End Developer
- Quality Assurance
- Security Analyst

5.2 Scope

The scope of the following tests will be to verify and confirm the stability and functionality of the web service, the job creation / management interfaces and the integrations with Windows Task Scheduler and Microsoft SQL Server.

5.3 Objectives

Our objective is to verify that the functionality of WAM (Windows Automation Manager) works as intended and according to the specifications we have set forth. The outcome of each test will help the development team correct any software bugs and ensure that we satisfy all project requirements.

It is expected that WAM will pass these tests, and if that is not the case development effort will be expected to fix and or rework the facet that failed to meet requirements.

5.4 Methodology

The methodology laid out in our test plan covers what we feel are the most important aspects of ensuring functionality and quality in our final product. Unit Tests written during the development of key functions ensures integrity throughout the development lifecycle. The Integration tests will ensure the core integration functionality is operating up to our expectations. Functional and Acceptance testing allows our test users to get a view of the product as a whole and will allow us to address any concerns from top to bottom.

5.5 Logging Test and Reporting

If during the testing process of WAM (Windows Automation Manager), a bug is found it is expected that the tester who found said bug will document it and schedule a time for developers to verify that the reported bug is an actual bug or not. Once this is determined, the development team will determine the course of action to take next.

5.6 Web Application Testing

WAM (Windows Automation Manager) will be tested as if it were a complete and fully functional web application. Testing in this manner will help the development team identify any areas within the web application that need to be addressed for fixes / tweaks to ensure we launch the best version of WAM possible

5.7 Test Procedures

1. Web Service Unit Tests - Small, bite-sized tests ensuring proper functionality for core components of the web site / web service
2. Integration Tests - Unit tests designed to test each aspect of the integration
    a. Windows Task Scheduler Tests
    b. SQL Agent Job Tests
3. System Functionality & Stability Tests - Overall stability / functionality user tests
4. Security & Assurance Tests - User testing for security and quality concerns
5. Acceptance Tests - Overall acceptance testing

5.8 Entry Criteria

- Working build is published in the test environment and accessible by testers
- Unit Tests and Integration Tests succeed

5.9 Exit Criteria

- Test procedures completed
- Unit Tests and Integration Tests still succeed
- Any bugs reported have been addressed

5.10 Schedule of Testing

Team Member	Timeline	Frequency
Developer	02/01/2019 – 4/8/2019	Weekly
Project Manager	02/01/2019 – 4/8/2019	Weekly
QA Analyst	02/01/2019 – 4/8/2019	Weekly

Table 2: Schedule of Testing

5.11 Test Case and Reporting

6. Procedure	Test Case Description	Expected Result	Actual Result	Status Pass/Fail/ Ongoing	Date Performed	Tester	Bug
1	Launch Web App	launch website, able to login	Able to see login	Pass	02/10/19	Proj Mgr	n/a
1	Host as IIS Site	Website is Accessible	Able to see login	Pass	02/10/19	Dev	n/a
2	Integration Tests	Unit Test Coverage	Unit Test Success	Ongoing		Dev	n/a
2	SQL Agent Job Tests	Unit Test Coverage	Unit Test Success	Ongoing		Dev	n/a
3	Create Task	Open “Create New Task” webpage and successfully submit task	Able to create new task	Pass	02/10/19	QA	n/a
3	Create SQL Agent Job	Open “Create New Job” webpage and successfully submit job	Able to create new Job	Pass	02/10/19	Dev	n/a
3	View Current Tasks	Open “View Current Task” webpage and view results	able to view running tasks	Pass	02/10/19	QA	n/a

4	Create User Account	Open “Create New User” webpage and successfully submit new user	able to fill out new user info and submit correctly	Pass	02/10/19	QA	n/a
4	Admin Login	Admin successful Login	able to login	Pass	02/10/19	Dev	n/a
4	User Login	User successful login	able to login	Pass	02/10/19	QA	n/a
4	Stability	Run scheduled task without stopping unexpectedly	able to run task without interruption	Ongoing		QA	n/a
5	Scheduled task and SQL Job Reports	View historical report for job / task performance		Ongoing		QA	n/a
5	Export Report	Report exports correctly		Ongoing		QA	n/a

Table 3: Test Cases and Reports

5.12 What we learned

The testing phase has been, and continues to be, enlightening to say the least. We have seen some of our design decisions (ie. page layout, in which certain variables live within the app itself cause navigation issues) come into question which have made us put some more thought into exactly what we’re wanting our users to experience when using WAM. Some changes were made, which threw off some of the integration unit tests we wrote - for example, the Windows Task integration test to create a task on a PC other than the server itself failed due to an invalid character being appended to the task type (rather than logon, it became logona). Having unit test coverage meant after changes to the integration code, we were able to simply rerun the tests and find the issue before it became a bug. So far in our testing progress, we have not yet encountered any major bugs, but the testing process continues. The benefits of having a testing plan are apparent, and process will continue until we submit our final deliverable.

6. Conclusion

6.1 Fall Semester 2018

The process of following our project from inception to realization has been both enlightening and insightful. We began the fall semester without a real idea - we threw aside a few less-than-stellar projects topics, and ultimately formed an idea we could all get behind and really see a use for within the industry. We had some struggles in the first few weeks of our project as we decided on the breakdown of work and each team member's responsibilities. We came to grips with each of our personal experiences and determined a project plan, a set of deliverables and a schedule we could hold ourselves to. We decided on the technology we'd use and set to research. After time spent researching and writing some small-scale tests, we found that we had a very strong base to fully realize this product.

Our proof of concept was constructed with relative ease after we had done our research - a few programming hills that we had to crest, but things came together. We setup an environment where we could show off the application's core functionalities - maintenance of any computer's Windows Tasks or SQL Agent Jobs on an internal network utilizing the UC Sandbox. With the application deployed in the test environment, we felt very confident in the demo we could provide for our presentation.

Moving forward, our goals are to continue to build our core application platform. We will flesh out our two core integrations and custom job engine and build out our reporting platform and then flip the switch and implement our API. We will work on a streamlined user interface and ensure that we have test coverage and have users hands-on in the system for feedback. Overall, this semester was eye-opening to the sheer scale of a fully-fledged IT project, and think we think we all have come out of it stronger in our project management skills. Looking to the future, we have full trust in our ability to deliver a fully fleshed out, in-scope project.

6.2 Spring Semester 2019

The senior design process has been one of much enjoyment and learning. The spring semester flew by in ways that we can't even begin to describe. While the fall was almost entirely focused on the how's and the what's of WAM, the spring was a time for refining and fine-tuning our project.

After getting the main architecture of the web application developed in the fall months, we used winter break and spring to start implementing the smaller things to improve the user experience. We wanted WAM to be professional looking as possible all while being simple enough to where there would be little to no learning curve for our perspective users. Other than tweaking the user interface, we also added a few user-friendly features to help make the WAM experience stand out.

We implemented the use of google charts within our web application to take our reporting and analysis pages to the next level. Adding this feature gave users an easy to understand visual to help them see how certain jobs are performing. While we did get the web application to about 99% of where we wanted it to be for IT Expo, we did run out of time to hit a few items we initially listed within the scope of our project.

Security-wise, we simply ran out of time on a couple of features. If we were to continue working on WAM in the future, the first item on our checklist would be connecting the application to active directory to enable us the ability to determine security roles and access based on group policy membership. We would have also liked to have included a web application firewall to also help prevent any possible security risks within the application.

All in all, this process has been eye-opening and full of moments of learning for each of us. I think we all can say without a shadow of doubt, that this course has helped us all learn what it is truly like to be involved in the full life-cycle of an IT project of this scale. I believe that each member of this group will cherish the lessons of the past few months for the remainder of our perspective IT careers.

Appendix A. References

1. Chui, Michael, James Manyika, and Mehdi Miremadi. "Four Fundamentals of Workplace Automation." November 15, 2015. https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/four-fundamentals-of-workplace-automation.