SOFTWARE EVALUATION REPORT

Final Version

Version: 1.0

Author(s): Jeremiah Smith (Technical Architect)

Date. 6.7.2022

PROJECT AND DELIVERABLE INFORMATION SHEET

Acme Project	**Project Title:** Acme Software Evaluation Report
	Project Website: https://acme-portal.fflf.org
	Company: Ace Solutions
	Client: Fight for Life
	Technical Architect: Jeremiah Smith
	Project Manager: Patrick Auger

DOCUMENT CONTROL SHEET

Document	**Title:** Software Evaluation Report
	Version: 1.0
	State: Draft
	Software Tool: Microsoft Word
Authorship	**Written By:** Jeremiah Smith (Technical Architect)
	Reviewed By:
	Approved By:

DOCUMENT STATUS SHEET

Version	Date	Status	Comments
0.1	7/June/2022	Draft	
1.0	1/August/2022	Final Version	

TABLE OF CONTENTS

REFERENCES AND APPLICABLE DOCUMENTS

1. Azure Portal: https://nonprofit.microsoft.com/ngoportal
2. AngularJS: https://angularjs.org/
3. AngularJS Change Log: https://github.com/angular/angular.js/blob/master/CHANGELOG.md
4. AngularJS Support: https://docs.angularjs.org/misc/version-support-status
5. Website Build Expert: https://www.websitebuilderexpert.com/building-websites/website-load-time-statistics/
6. Application Security Checklist: https://dzone.com/articles/application-security-checklist
7. OWASP Web Application Security Testing Checklist: https://github.com/OWASP/wstg
8. OWASP: https://owasp.org/
9. WSJF: https://www.scaledagileframework.com/wsjf/

LIST OF ACRONYMS AND ABBREVIATIONS

ADO	**Azure DevOps**
AI	Application Insights
CRUD	Create / Read / Update / Delete
CSS	Cascading Style Sheet
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol (HTTPS is secure version)
JS	JavaScript
OWASP	Open Web Application Security Project
SFTP	Secure File Transfer Protocol
SQL	Structured Query Language (Database Language)
TLS	Transport Layer Security
WSJF	Weighted Shortest Job First

EXECUTIVE SUMMARY

1. Introduction

This deliverable reports on the technical analysis of the web application named “Acme Portal”. In this document a presentation of this work has been prepared under 3 major topics:

- Web Application - Architecture, technologies, and performance
- System Architecture – Hosting environment, cloud components, and architecture
- Security and Compliance – Review OWASP security checklist with pass or fail determination

We will present our results and findings for each of these topics, based on which we will conclude with a set of recommendations.

2. Web Application

In this chapter, we present the results of our investigation into overall technology stack and architecture of the web application. Software stack is a term used to describe the overall combination of multiple chosen programming languages, coding libraries, data storage and management, etc. that makes up your entire application.

2.1. Overview

2.1.1. .NET Framework 4.7.2/ C#

.NET is a developer platform, developed and maintained by Microsoft, containing tools and programming libraries for building many different types of applications. .NET serves as one of topmost popular development frameworks and is a top contender in the Indianapolis area.

Acme back-end services (code and business logic that is executed at the server level) is primarily developed with this framework. The backend-services consists of the following layers:

- **Web Service Layer** – the interface that serves the front-end / client-side code from the browser.
- **Business Layer** – consists of all the business rules, calculations and actual logic within your application that makes it actually “do” things. Additionally, will retrieve information from a data layer to serve to the end user.
- **Data Layer** – primary responsibility is to create, read, update, and delete (also known as CRUD) of the application data.
- **Web Job Layer** – This is an isolated layer that is unique to this application. Primary responsibility of this layer is to encode video files for the application.

2.1.2. Code Libraries

Code libraries are collections of prewritten code that can be used to perform a task within your application. Such examples may include configuration of data, calculations, storage, reading/writing to a database etc.

Acme uses the following C# Libraries:
If highlighted, known vulnerability exists in installed version

Library / Package	**Installed Version**	**Latest Version**
Antlr	3.5.0.2	
CsvHelper	12.1.2	27.2.1
EntityFramework	6.2.0	6.4.4
Ical.NET	4.2.0	
JWT	1.3.4	9.0.0
Microsoft Application Insights	2.5.1	2.20.0
Microsoft AspNet MVC	5.2.4	5.2.9
Microsoft Azure Key Vault	1.0.0	3.0.5
Microsoft Azure Web Jobs	2.3.0	3.0.33
Microsoft Data EDM	5.8.4	5.8.5
Microsoft Data OData	5.8.4	5.8.5
Microsoft Services Client	5.8.4	5.8.5
Microsoft Identity Model	1.1.2	6.20.0
Microsoft TPL Dataflow	4.5.24	*Deprecated*
Microsoft SlowCheetah	3.2.26	4.0.8
Microsoft Windows Azure Configuration Manager	3.2.1	3.2.3
Ncrontab	3.3.0	3.3.1
Newtonsoft.Json	12.0.2	13.0.1
NodaTime	3.0.0	3.1.0
NReco.PdfGenerator	1.2.0	
QRCoder	1.4.1	1.4.3
System.Spatial	5.8.4	5.8.5

TransientfaultHandling	5.1.1209.1	
Twilio	5.52.1	5.75.3
WebGrease	1.6.0	
Windows Azure Media Services	4.2.0	*Deprecated*
Windows Azure Storage	9.3.3	*Deprecated*

2.1.3. AngularJS (Version 1.7.6)

AngularJS is a client-side (web browser) framework for dynamic web applications. This language extends your HTML web pages into templates. These templates allow for simple data binding of the data to the web page displayed to the end user.

Additionally, this library consists of other useful tools such as form-validation, page routing, deep-linking, and reusable components.

Acme primarily uses this library to map data provided by the web service layer to the HTML pages presented to the end-user and prepare the data structure that is sent back to the back-end application.

2.1.4. JavaScript Plugins

Java is a scripting language that enables you to dynamically calculate, manipulate and validate data on a web page. A JavaScript plugin is pre-written pieces of code, written in JavaScript, that provides useful features to your web page. This reduces developer's time by not having to develop the functionality themselves and focus on the implementation.

Acme uses the following JavaScript plugins:

Library / Package	**Installed Version**	**Latest Version**
Angular Block UI	0.2.1	
Angular Data tables	0.6.5	13.1.0
Angular Drag and Drop lists	2.1.0	
Angular Fitvids	0.1.0	0.1.1
Angular Notify	1.2.0	
Angular UI Validate	N/A	
Angular Upload	N/A	
Azure Media Player	2.3.5	2.3.10
Data Tables	1.10.16	1.12.1
Date Range Picker	3.0.3	3.1

Google Analytics	*Latest*	
iCheck	1.0.2	1.0.3
jQuery	3.3.1	3.6.0
Mobiscroll	4.10.9	5.17.1
Ng-tags-input	3.2.0	
Select2	4.0.6	4.1.0
Spectrum	1.8.0	
Tagify	2.9.9	4.12.0
Text-Angular	N/A	
UI-Mask	1.8.7	
Zxcvbn	N/A	
Font-Awesome	5.14.0	6.1.1
Jquery Modernizer	2.6.2	3.5.0

2.1.5. CSS Framework

CSS stands for Cascading Style Sheet and is responsible to instruct the browser how web elements should be displayed on your web page such as size, position, and color. A CSS framework is a collection of CSS stylesheets that prepped and ready to use. This reduces the developer's time by allowing them to focus on the implementation.

Acme uses the following CSS frameworks:

Library / Package	**Installed Version**	**Latest Version**
Bootstrap	4.6.0	5.2
Google Fonts	N/A	
Data Tables	1.10.16	1.12.1
Select2	4.0.6	4.1.0
Titatoggle	1.2.15	
Angular Notify	1.2.0	
Angular Block UI	0.2.1	
Ng-tag-input	3.2.0	
Spectrum	1.8.0	

Text Angular	N/A	
Date Range Picker	3.0.3	3.1
Azure Media Player	2.3.5	2.3.10
AngularJS Mobiscroll	4.10.9	5.17.1

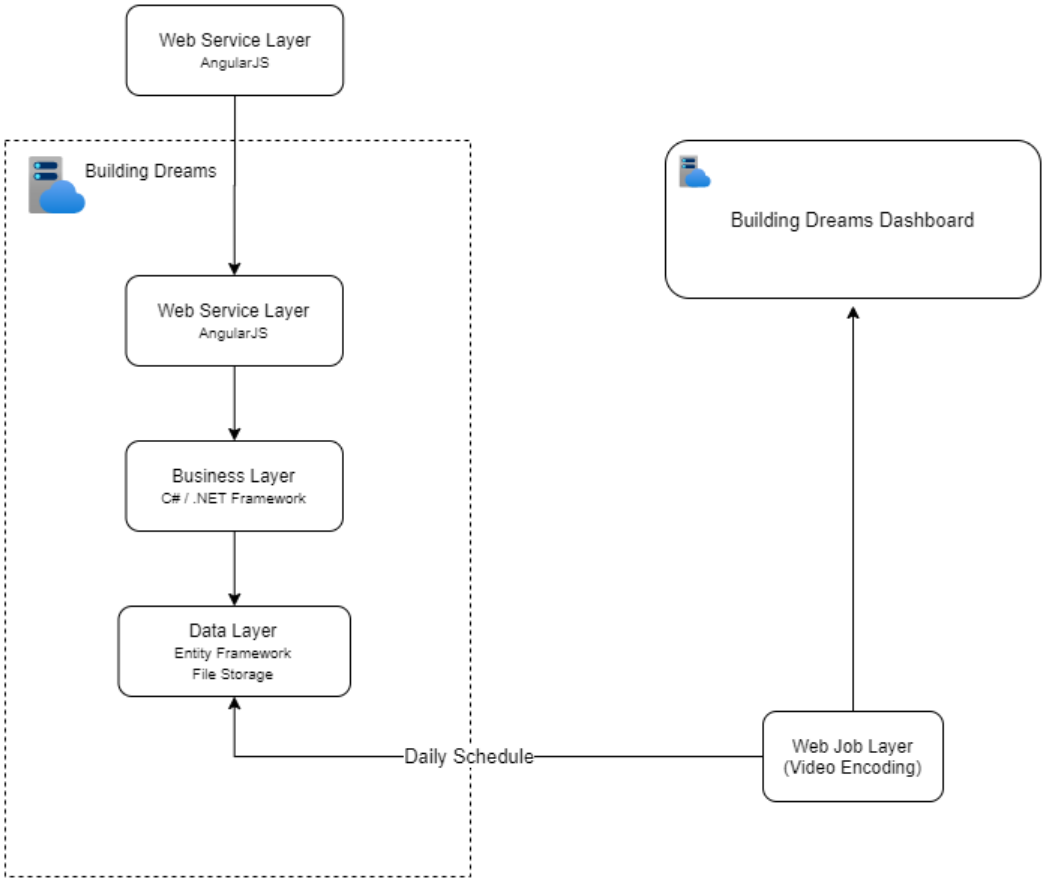
2.1.6. Data Persistence

A database is a collection of data stored in an electronic form. Data is the most valuable piece within most web applications. It is essential to a business and contains the pertinent details about the company such as employee and user records etc.

Acme utilizes two types of data persistence and file storage:

- Microsoft SQL Database
- Azure Storage Account

2.1.7. Software Architecture Diagram



2.2. Performance

Speed and performance play a big role in a user's experience. To investigate the performance of the Acme web application, we analyzed the existing Azure resource called Application Insights. This tool provides analytics and telemetry of the web application running in the production environment. Application Insights is configured to track a sample set of requests to get an overview of the application's performance and failures.

We'll focus on operations, failures, and dependencies to study the overall application health and potential user experiences. We'll be analyzing data from the last 90 days, ranging from March 17th, 2022 to June 14th, 2022.

2.2.1. Operations

In 2022 study, the 47% of users expects a web application's page to load within 2 seconds. The same study showed that 40% of people will abandon a website of web applications that take more than 3 seconds to load. A slow user experience is not only bad for business but is a red flag that the slow operation will only continue to degrade as the overall application load grows.

The below operations are demonstrating the top 10 operations sorted by the number of Requests made in the mentioned time frame.

Operation Name	**Requests**	**Avg Duration**	**Duration 50th Percentile**	**Duration 95th Percentile**	**Duration 99th Percentile**
POST Message/Status	89741	0.02 sec	0.02 sec	0.05 sec	0.17 sec
POST Announcement/Status	51596	0.03 sec	0.02 sec	0.07 sec	0.2 sec
GET Home/Index	26413	0.01 sec	0.01 sec	0.01 sec	0.02 sec
GET Account/Login	2000	0.12 sec	0.04 sec	0.15 sec	0.48 sec
GET File/Get	1175	0.12 sec	0.08 sec	0.3 sec	1.01 sec
POST Classroom/AssessmentResponses	682	0.1 sec	0.08 sec	0.19 sec	0.4 sec
POST Classroom/Profile	510	1.86 sec	1.46 sec	3.82 sec	4.36 sec
POST Student/BehaviorInfo	498	0.05 sec	0.03 sec	0.08 sec	0.36 sec
GET Admin/Index	484	0.32 sec	0.06 sec	0.17 sec	12.56 sec
POST Login/CheckLogin	373	1.28 sec	.03 sec	5.06sec	5.68 sec

As shown in the fore-mentioned chart above, the top operations have an average duration within 2 seconds. All operations above a request count of 1000 are within the recommended time span in all percentiles.

Consideration: Review the **Post Classroom/Profile** operation as it's nearing the 2 second recommended load time with a higher percentile exceeding that amount. Secondarily, review the **POST Login/CheckLogin** operation for rare long load times of 5 seconds or more. Lastly, to review the 12.56 second load time of **GET Admin/Index** in the 99th percentile as a lower priority.

2.2.2.Failures

A failure occurs when an error or exception occurs in the web application unexpected and causes the user's action to fail due to the cause.

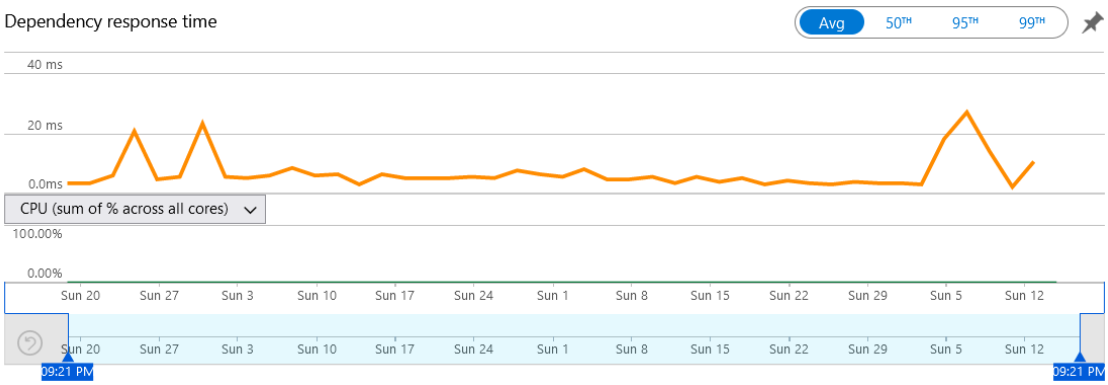
Based on the below chart, failure in your operations is a rare case over a 90-day period. Over the time, there was a total of 162 failures over 180,051 operations.

Here are the top 10 failures in the web application:

Operation Name	Failed Count	Total Count
POST Message/Status	41	89737
POST Announcement/Status	37	51592
GET External/Validate	9	39
GET .git/config	8	8
GET sitemap/Index	4	4
HEAD main/Index	3	3
HEAD bc/Index	3	3
HEAD wp/Index	3	3
GET wp-content/uploads	3	3
HEAD old/Index	3	3

2.2.3. SQL Database

Over the 90-day period and a sample set of 373,000 data operations, the average duration of each execution is 5.06ms. Reviewing the 99th percentile, the average duration continues to be 225ms. Overall, the database is in a very healthy and performant state.



3. System Architecture

3.1. Overview

This section focuses on the various components that are related to the Acme overall system architecture. This chapter is separated into four sections to “Host and System Management”, “Cloud Resources”, “Data Management”, “Conclusion”.

3.1.1. Cloud Resources

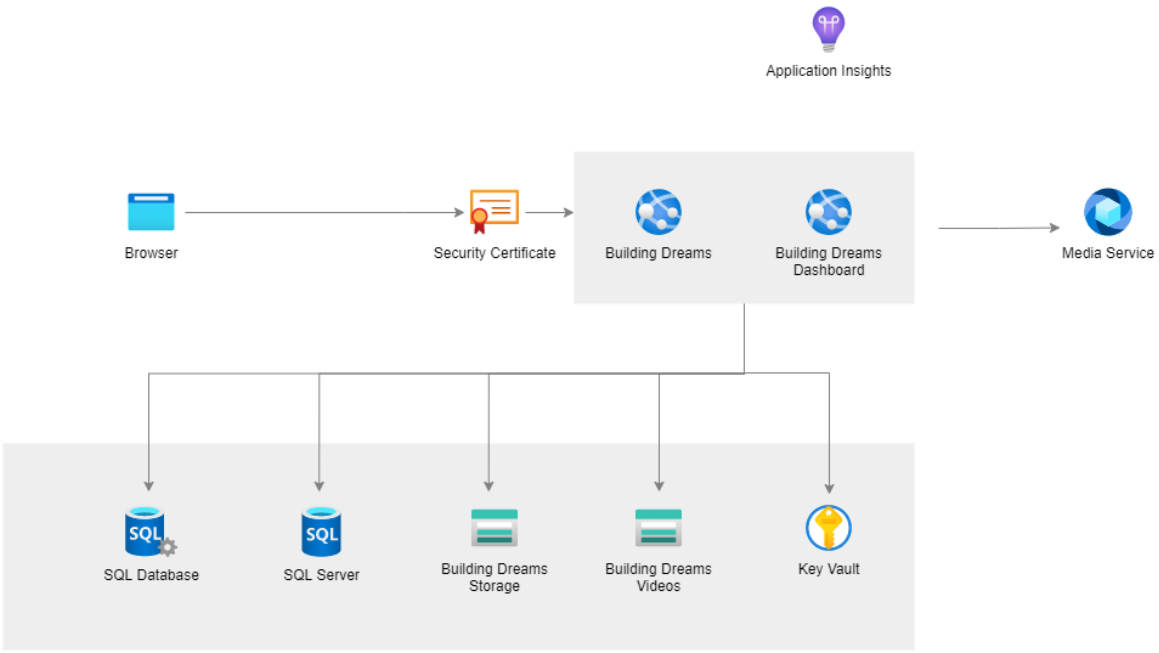
Acme uses a cloud environment called Azure. Azure is a cloud platform that offers various platform, infrastructure, software services and resources.

Fight for Life foundation utilizes many of these cloud services to enable their web application listed below:

Type	Resource Name	Description
Application Insights	AI-buildingdreams-a84f	Track analytics and telemetry of Acme app service
App Service Plan	ASP-buildingdreams-b6d4	Defines compute resources for Acme web app
App Service Plan	ASP-buildingdreamspython-a889	Defines compute resources for Acme Dashboard web app
App Service Plan	ASP-buildingdreamspython-b35d	Defines compute resources for Acme Dashboard web app (test environment)
SQL Server	Acme-portal	Hosts SQL database instances

SQL Database	Acme-portal/acme-portal	SQL database instance for Acme web app
App Service	Acme-portal	Web Application
App Service	Acme-portal-dashboard	Reporting tool for Acme web app
App Service Certificate	Acme-portal-cert	Security Certificate to provide HTTPS and secure connection for visitors via web browser
Key vault	Acme-portal-keys	Encrypt key and small secrets used by web app services
Key vault	Acme-portal-vault	Encrypt key and small secrets used by web app services
Media Service	Buildingdreams	Provides low latency streaming content to viewers of any device
Storage account	Buildingdreamsstorage	Additional file storage for Acme web app
Storage account	Buildingdreamsvideos	Video file storage for media services
Streaming Endpoint	Default	Delivers on-demand content directly to client player app
App service	Test-bd-dashboard	Test environment for Acme Dashboard web app

3.1.2. Cloud Resources Diagram



4. Security (OWASP Application Security Testing)

The objective of this chapter is to evaluate the existing security controls with the analysis of vulnerabilities of Acme web application. Below is the process and items that are reviewed to verify the effectiveness of the current security controls

4.1. Information Gathering

4.1.1. Fingerprint Web Server

Acme Web App Service

Configuration	**Pass/Fail**	**Description**
Test for HTTP Response Header Fields	Fail	X-Powered-By mentions ASP.NET
Review Webpage Content for Information Leakage	Pass	

4.2. Configuration Deployment and Management

4.2.1. Test Network Infrastructure Configuration

Acme Web App Service

Configuration	**Pass/Fail**	**Description**
HTTPS Only	Pass	
TLS Version 1.2	Pass	
SFTP Only	Fail	
Remote Debugging Off	Pass	
Unneeded Virtual Directories	Pass	
Sensitive App Setting Values	Pass	
Behind Firewall	Fail	

Acme Dashboard Web App Service

4.2.2. Test Application Platform Configuration

Acme Web App Service

Configuration	Pass/Fail	Description
Sample and Known Files and Directories	Fail	Web directory contains hostingstart.html file in root
Logging – Sensitive Information	Pass	No application logs
Logging – Isolation	Pass	No application logs
Logging – Rotation	Pass	No application logs
Logging – Preservation	Pass	No application logs
Log reviewed for attackers or malformed requests	Fail	No application logs

4.2.3. Test File Extensions Handling for Sensitive Information

Acme Web App Service

Configuration	Pass/Fail	Description
Forced Browsing – Files that should not be served by browser	Pass	
File Upload Validation	Fail	No validation logic associated to file upload controls

4.2.4. Review Unreferenced Files for Sensitive Information

Acme Web App Service

Configuration	Pass/Fail	Description
Sensitive information/clues in comments of published content	Pass	
Directory listing in browser	Pass	

4.2.5. Test HTTP Methods

Acme Web App Service

Configuration	Pass/Fail	Description
Enumerate supported HTTP methods	Pass	
Test for access control bypass	Pass	
Test HTTP method overriding techniques	Pass	

4.2.6. Test HTTP Strict Transport Security

Acme Web App Service

Configuration	Pass/Fail	Description
Review the HSTS header and its validity	Fail	

4.2.7. Test for Content Security Policy

Acme Web App Service

Configuration	Pass/Fail	Description
Review the Content-Security-Policy header	Fail	

4.3. Identity and Access Management

4.3.1. Test Role Definitions

Acme Web App Service

Configuration	Pass/Fail	Description
Identify roles used by application	Pass	
Attempt to switch or change roles	Pass	
Review permissions given to role	Pass	

4.3.2. Test User Registration Process

Acme Web App Service

Configuration	Pass/Fail	Description
Verify requirements for user registration	Pass	
Validate the registration process	Pass	

4.3.3. Test Account Provisioning

Acme Web App Service

Configuration	Pass/Fail	Description
Verify which accounts may provision other accounts and of what type	Pass	

4.3.4. Test for Account Enumeration and Guessable User Account

Acme Web App Service

Configuration	Pass/Fail	Description
Process for user identification	Pass	Email Addresses are used

4.4. Authentication Testing

4.4.1. Test for Default Credentials

Acme Web App Service

Configuration	Pass/Fail	Description
Any user accounts with default passwords	Pass	
New user accounts are created with weak or predictable passwords	Fail	Student accounts are created using a 5-digit pin/code. This is a weak authentication factor and easily brute forced.

4.4.2. Test for Weak Lock Out Mechanism

Acme Web App Service

Configuration	Pass/Fail	Description
Account lockout mechanism to mitigate brute force password guessing	Fail	No limit of login attempts, and no lockout strategy implemented

4.4.3. Test for Bypassing Authentication Schema

Acme Web App Service

Configuration	Pass/Fail	Description
Forced Browsing	Pass	
Parameter modification	Pass	
Session ID prediction	Pass	
SQL injection	Pass	

4.4.4. Test for Vulnerable Remember Password

Acme Web App Service

Configuration	Pass/Fail	Description
Validate generated session is managed securely	Pass	

4.4.5. Test for Browser Cache Weakness

Acme Web App Service

Configuration	Pass/Fail	Description
Application store sensitive information on client-side	Pass	
Access to cache information without authorization	Pass	

4.4.6. Test for Weak Password Policy

Acme Web App Service

Configuration	Pass/Fail	Description
Password Complexity Rules	Fail	No password requirements are set

4.4.7. Test for Weak Password Change or Reset Functionalities

Acme Web App Service

Configuration	Pass/Fail	Description
Test for weak password change and reset functionality	Pass	

4.5. Authorization Testing

4.5.1. Test Directory Traversal File Include

Acme Web App Service

Configuration	Pass/Fail	Description
Identify injection points that pertain to path traversal	Pass	
Assess bypassing techniques and identify the extent of path traversal	Pass	

4.5.2. Test for Bypassing Authorization Schema

Acme Web App Service

Configuration	Pass/Fail	Description
Access resources even when not authorized	Pass	
Access resources after log-out	Pass	
Access functions or resources from an invalid role or privilege	Pass	
Access admin functions when logged in as a non-admin user	Pass	

4.5.3. Test for Privilege Escalation

Acme Web App Service

Configuration	Pass/Fail	Description
Identify injection points related to privilege manipulation	Pass	
Attempt to bypass security measures	Pass	

4.5.4. Test for Insecure Direct Object References

Acme Web App Service

Configuration	Pass/Fail	Description
Value of a parameter is used directly to retrieve a database record	Pass	
Value of a parameter is used to directly perform an operation in the system	Pass	
Value of a parameter is used directly to retrieve a file system resource	Pass	
Value of a parameter is used directly to access application functionality	Pass	

4.6. Session Management Testing

4.6.1. Test for Session Management Schema

Acme Web App Service

Configuration	Pass/Fail	Description
Are all Set-Cookie directives tagged as secure	Pass	
Any cookie operations take over unencrypted transport	Pass	
Cookie be forced over unencrypted transport	Pass	
Cookie persisted	Pass	
Cookie expiration reasonable	Pass	

4.6.2. Test for Cookie Attributes

Acme Web App Service

Configuration	Pass/Fail	Description
Are all Set-Cookie directives tagged as secure	Pass	
Any cookie operations take over unencrypted transport	Pass	
Cookie be forced over unencrypted transport	Pass	
Cookie persisted	Pass	
Cookie expiration reasonable	Pass	

4.6.3. Test for Session Fixation

Acme Web App Service

Configuration	Pass/Fail	Description
Analyze the authentication mechanism	Pass	
Force cookies and assess the impact	Pass	

4.6.4. Test for Exposed Session Variables

Acme Web App Service

Configuration	Pass/Fail	Description
Ensure that proper encryption is implemented	Pass	
Review the caching configuration	Pass	
Assess the channel and methods' security	Pass	

4.6.5. Test for Cross Site Request Forgery

Acme Web App Service

Configuration	Pass/Fail	Description
Absence of Anti-CSRF Tokens	Fail	No Anti-CSRF Token are found in session cookie for all web forms

4.6.6. Test for Logout Functionality

Acme Web App Service

Configuration	Pass/Fail	Description
User interface controls that allow the user to manually log out	Pass	
Session termination after a given amount of time without activity	Pass	20-minute timeout period

4.6.7. Test for Session Timeout

Acme Web App Service

Configuration	Pass/Fail	Description
Validate that a hard session timeout exists	Fail	No timeout and redirect exist

4.6.8. Test for Session Hijacking

Acme Web App Service

Configuration	Pass/Fail	Description
Identify vulnerable session cookies	Fail	User session cookie is not marked with secure attribute

4.7. Input Validation Testing

4.7.1. Test for Reflected Cross Site Scripting

Acme Web App Service

Configuration	Pass/Fail	Description
Identify variables that are reflected in responses	Pass	
Assess the input type the form fields accept	Pass	

4.7.2. Test for Stored Cross Site Scripting

Acme Web App Service

Configuration	Pass/Fail	Description
Identify stored input that is reflected on client side	Pass	None identified

4.7.3. Test for SQL Injection

Acme Web App Service

Configuration	Pass/Fail	Description
Identify SQL injection points	Pass	

4.7.4. Test for Format String Injection

Acme Web App Service

Configuration	Pass/Fail	Description
Assess whether injecting format string conversion exists	Pass	

4.7.5. Test for Incubated Vulnerability

Acme Web App Service

Configuration	Pass/Fail	Description
Identify upload file controls that are restricted to file type	Fail	File upload controls are not controlled by any specific types

4.7.6. Test for HTTP Spitting Smuggling

Acme Web App Service

Configuration	Pass/Fail	Description
Assess if application vulnerable to spitting	Pass	
Assess if chain of communication is vulnerable to smuggling	Pass	

4.7.7. Test for Mass Assignment

Acme Web App Service

Configuration	Pass/Fail	Description
Permission-related properties set only by privileged users	Pass	e.g contactType, isAdmin, etc.
Process-dependent properties should be set internally	Pass	e.g. status, emailVerified, etc.
Internal properties should be only set internally	Pass	e.g. createdDate, updatedDate, etc.

4.8. Test Error Handling

4.8.1. Test for Improper Error Handling

Acme Web App Service

Configuration	Pass/Fail	Description
Random file path returns 404	Pass	
Request folder names in URL should not display the directory listing	Pass	
Internal properties should be only set internally	Pass	
Internal server exceptions should return general error message notification or page	Fail	Internal stack trace page is displayed in the production environment

4.9. Testing for Weak Cryptography

4.9.1. Test for Weak Transport Layer Security

Acme Web App Service

Configuration	Pass/Fail	Description
Validate the service configuration	Pass	
Review digital certificate cryptographic strength and validity	Pass	

4.9.2. Test for Sensitive Information Sent via Unencrypted Channels

Acme Web App Service

Configuration	Pass/Fail	Description
Identify sensitive information transmitted through the various channels	Pass	
Assess the privacy and security of the channels used	Pass	

4.9.3. Test for Weak Encryption

Acme Web App Service

Configuration	Pass/Fail	Description
Identify weak encryption or hashing uses or implementation	Pass	

4.10. Business Logic Testing

4.10.1. Test for Business Logic Data Validation

Acme Web App Service

Configuration	Pass/Fail	Description
Validate all checks are occurring on the back end and can't be bypassed	Fail	Back-end validation messages are coming returned from database calls. This includes internal application information that should not be returned

4.11. Client-Side Testing

This chapter was mainly tested using an automated tool called OWASP Zap. See document asset **Zap-Report.pdf.**

4.11.1. Test for Front-end Vulnerabilities

Acme Web App Service

Configuration	**Pass/Fail**	**Description**
Test for DOM-based cross site scripting	Pass	
Test for Self-based cross site scripting	Pass	
Test for JavaScript execution	Pass	
Test for HTML Injection	Pass	
Test for Client-side URL redirect	Fail	Login contains ReturnURL query parameter that sends user to external website
Test for CSS injection	Pass	
Test for client-side resource manipulation	Pass	
Test cross origin resource sharing	Fail	Access-Control-Allow-Origin is set to all
Test for cross site flashing	Pass	
Test for clickjacking	Fail	Missing anti-clickjacking header
Test web sockets	Pass	
Test web messaging	Pass	
Test browser storage	Pass	
Test for cross site scripting	Pass	
Test for reverse tab nabbing	Pass	
Test for vulnerable JavaScript libraries	Fail	AngularJS 1.7.6 is vulnerable. Recommend upgrade to latest version 1.8.2. Additionally AngularJS, as a whole framework, is no longer supported as of January 2022.

5. Recommendations

The recommendations below utilize the information found in the web application architecture, performance and security analysis done prior in the document. Each recommendation is labeled with an estimation of time to develop and the impact level it would make (such as lowering vulnerability risk and/or user experience or protection).

Therefore, a calculation of the WSJF (Weighted Shortest Job First) score is determined. This algorithm assists in sorting the priority of tasks that should be completed first.

Priority #	**Recommendation**	**Estimated Time** < 16 Hours = 1 16 – 40 Hours = 2 40+ = 3	**Impact Level** Low = 1 Medium = 2 High = 3	**WSJF Score** Impact Level / Estimated Time
1	Increase Security on Student Passcode Login Currently the application's student login is utilizing the teacher's email address and a 5-digit pin that is unique for each student. In the modern world of attackers, this would be an easy brute force attempt. **Recommend that we:** • Add reCAPTCHA challenge before login • Add birthdate for student login (or other easily identifiable piece) • Change passwords to alphanumeric as minimum	16 Hours	High	3
2	Create Firewall for Applications All application web services are protected by their own programmatic authentication strategy. Azure Firewall Standard provides L3-L7 filtering and threat intelligence feeds directly from Microsoft Cyber Security. Threat intelligence-based filtering can alert and deny traffic from/to known malicious IP addresses and domains which are updated in real time to protect against new and emerging attacks. **Important**: This requires a monthly Azure bill increase of $1277	4 Hours	High	3
3	Display General Error Page When Error/Exception Occurs Currently, Internal stack trace page is displayed in the production environment. This is sensitive information that any attacker could use to decipher and break down the application for other vulnerabilities	1 Hour	Medium	2
4	SQL Server Audit Enable SQL Server auditing in the Azure Portal	4 Hours	Medium	2
5	File Upload Validation If a workbook/form submission requires a file upload of any kind, the web application does not check for the extension of that file. This means your susceptible of having potential malware, or other harmful files, in your file storage within the cloud. Although the cloud's architecture should protect you from this, there is still risk from a user revisiting or reviewing a "student's" input and executing the harmful file on their machine.	6 Hours	Medium	2
6	Remove or Modify ReturnURL Query Parameter on Login Screen The login page will send any user to external website if the Return URL parameter is supplied a valid URL, internal and external. This could lead	8 Hours	Medium	2

	your users to a phishing attack by receiving an email from an attacker with a copycat website in the redirect URL parameter.			
7	Enable Only SFTP In Azure Portal, SFTP and FTP are allowed forms of deployment to the Application Web Service. It's recommended that deployments should always be secured utilizing SFTP	1 Hour	Medium	2
8	New User Accounts are Created with Weak or Predictable Passwords Recommend adding complexity rule set for new users. This would enforce non-student users to have strong passwords. This would increase security at a high level and assist in preventing brute force attacks.	24 Hours	High	1.5
9	Add Account Lockout Mechanism This recommend is to mitigate brute force password guessing by limiting the user so many attempts to login. After X number of invalid credentials, the account may become locked. Therefore, an email is sent out to user account to reset the password or unlock the account	24 Hours	High	1.5
10	AngularJS to Latest Angular Front-End Framework AngularJS Framework was officially announced support would discontinue in January 2022.	12 weeks	High	1
11	Implement Application Logging No logging framework was found in the application. This can be a crucial element for all types of troubleshooting issues and reviewing for potential malformed requests from attackers	40 Hours	Medium	1
12	Add the Following HTTP Header Attributes: • HSTS Header • Content-Security-Policy • Access-Control-Policy • Access-Control-Allow-Origin • Anti-Clickjacking	8 Hours	Low	1
13	Remove HTTP Header Attribute of X-Powered-By In every web page request, the HTTP Response contains a HTTP Header attribute of X-Powered-By mentions ASP.NET. This is an information gathering tool for attackers to know the tech stack they are targeting	1 Hour	Low	1
14	Add Anti-CSRF Tokens in Session Cookies This would provide the user/browser with a secret token that is sent with each form submission. In the attempt that a session is hijacked, this must be sent with a form submission for the web application to trust and react to the submission	8 Hours	Low	1
15	Mark User Session Cookie with Secure Attribute This ensures that session cookies for each user are sent over a secure channel. The browser will prevent the transmission of a cookie over a n unencrypted channel	2 Hours	Low	1
16	Add Timeout and Redirect to Page Recommend forcing user to login page if the user is inactive for X amount time on the web application. This is especially important in public areas with public computers in case the user makes a mistake of not closing out of their account before leaving the machine.	4 Hours	Low	1
17	Return Friendly Validation Messages from API Responses	8 Hours	Low	1

	Back-end validation messages are being returned from the SQL server when attempting to add or modify a record. This could include sensitive information about the application			
18	Enable SQL Auditing There is no accountability or record keeping of changes that are occurring in the SQL database. This is usually a big piece in any compliance documents along with resolving internal application issues. **Important**: This requires a monthly Azure bill increase of unknown amount of storage	1 Hour	Low	1
19	Update Third-Party Packages with Known Vulnerabilities Update all third-party packages with vulnerabilities to their latest version to secure any known vulnerabilities	16 Hours	Medium	1
20	Update All Third-Party Packages Update all third-party packages to their latest version	60 Hours	Low	0.5

6. Conclusions

The web application's software and technology stack are a common and logical structure. Although Microsoft has begun deprecating their .NET Framework technology, this should not be a concern in the near to midterm future. .NET Core / 6 will the future for Microsoft and their software development atmosphere.

The biggest focus in technology choice is the AngularJS framework. Since this library is no longer supported since January 2022, there will be an on-going concern of vulnerabilities that arise during its existence in the web application. This will take a large effort and duration due to this technology being responsible for the entire front-end website. Although the cost will be well rewarded for the lifespan of the web application.

The performance of the application is well suited for the current needs and well beyond into the future. We predict that the application will have the ability to scale with the growth of the company and the near future size of the user base. We recommend monitoring the performance metrics on a bi-annual basis and review if performance is a future concern to being taking actions and stay ahead.

Security and compliance of the application is well suited overall. We discovered several potential vulnerabilities within the architecture and the workflow of the application. It is unlikely currently, with the size of your user base and company growth, that you have become a target for any attackers. It's highly recommended to review vulnerabilities on regular basis as new security recommendations and software patches occur daily.

Overall, the Acme application is a top-quality web application with pure fundament software development concepts. The application uses software design patterns and technologies that you would expect for an application, of this scale, and purpose. Acme will have a long and promising future ahead of itself.