|  |
| --- |
| Merrilton Bank |
| Cloud Infrastructure Design |
| D088 Final Assessment |
| John Kowalsky  5/22/21  Version 2 |

Contents

[A. Authentication Process 3](#_heading=h.gjdgxs)

[B. Remote Access](#_heading=h.30j0zll) 4

[C. Application Security](#_heading=h.1fob9te) 4

[D. Network Security](#_heading=h.3znysh7) 4

[E. Internal APIs](#_heading=h.2et92p0) 5

[F. External APIs](#_heading=h.tyjcwt) 5

[G. Deployment Plan](#_heading=h.3dy6vkm) 6

[H. Maintenance Strategy](#_heading=h.1t3h5sf) 7

[I. Disaster Recovery Plan](#_heading=h.4d34og8) 8

[J. Regulatory Compliance](#_heading=h.2s8eyo1) 9

K[. Sources](#_heading=h.17dp8vu) 9

# Authentication Process

The authentication process for Merrilton Bank’s cloud application will be handled by Microsoft Azure Authentication.

First, users will be issued a unique numerical username and asked to create a secure password for their account. Next they will be prompted to set up Multi-Factor Authentication for their account. Users can choose to receive the required MFA code in their email or by text message to their mobile device.

Along with Multi-Factor Authentication, a user’s account will be associated with their sign-in device upon first login. Any new sign in from a different or unknown sign-in device or browser will require Multi-Factor Authentication before the user can access their account.

Azure AD Conditional Access Policy will be used to allow no more than ten devices or browsers to be associated with any one user’s account. It will cross reference what IP address the user’s device is attempting to access the platform on as well as what type of browser the user is using.

Azure AD Conditional Access Policy will use the IP address and browser type to uniquely identify a user’s account and ensure that access is only granted if the Conditional Access Policy’s security requirements are met. The following device and browser types will be allowed: Android, iOS, Windows Phone, Windows, MacOS, Microsoft Edge, Internet Explorer, Chrome, and Safari.

Fraud alerts will be configured to notify administrators of any potential suspicious activity. For instance, if a user usually logs in from their home city and then suddenly logs in from another state, Azure will flag the attempt as a Risky-User and an alert will be sent to the security administrators for further scrutiny to determine the risk/threat level.

Azure’s password hash synchronization, along with seamless SSO will allow users to sign in and be authenticated from anywhere.

# Remote Access

Employees will remotely authenticate to the cloud environment using Multi-Factor Authentication. First they will sign in with their username and password. After successfully entering their password, they will be sent a unique one-time six digit code to either their email or by text message to their mobile device. They will be prompted to enter the MFA code and select Verify. Once Azure MFA has authenticated their credentials, the employee will be able to access the cloud environment.

Access will occur through an encrypted VPN gateway, using Azure’s VPN Gateway service. User’s will be able to send encrypted traffic between their device/browser and the banking application’s virtual and on-premise network over the public internet.

# Application Security

To provide end-to-end encryption for communication between the application and the cloud, we will use Microsoft Azure Application Gateway. Application Gateway will provide features such as load balancing HTTP traffic, web application firewall, and support for SSL encryption of your data. For end-to-end encryption, Application Gateway will decrypt any traffic on the gateway through the use of a private key. It will then re-encrypt it again with the public key of the service running in the backend pool.

# Network Security

For encryption between the home office and the cloud, we will use Microsoft Azure VPN Gateway. VPN Gateway is a virtual network gateway that will be used to send encrypted traffic between an Azure virtual network and an on-premises network location.

# Internal APIs

The banking mobile application will use Splunk’s REST API to communicate with Azure Event Hub and Stream Analytics to run the bank fraud services and deliver notifications to the end user if suspicious or fraudulent activity is detected. The JSON code will run in Azure Functions and be invoked when the parameters for potential fraud have been detected. Once invoked the api will trigger a push notification to be sent out using Azure Notification Hub.

For branch location information, the mobile banking application will use the phone’s GPS location information and the Google Map API to provide directions to the nearest branch location. The JSON code will run in Azure Functions and be invoked when the request for branch location is activated within the banking application. Using Azure API Management, the application will interface with the Google Map API to detect the user’s location and return the branch locations nearest to the present location of the user.

All log data for the banking mobile application will use Splunk’s REST API to return the log data which will be stored and accessed with Azure Monitor using Azure’s API Gateway. The JSON code will run in Azure Functions and be invoked on an automated daily basis to ensure all log data is backed up. Once invoked, the api will send all log data to Azure Monitor for storage and later analysis.

# External APIs

The banking mobile application will use Azure AD and Seamless SSO for single sign-on. This will be managed by Azure API Management. The JSON code will run in Azure Functions and follow the following invocation rules: If the user is already signed in to the application, they will be authenticated. If not, Seamless SSO will use JavaScript in the background to request an authentication ticket from a Kerberos server. Once a ticket is provided, MFA will be required to further satisfy the authentication requirements of the mobile application.

In order to provide the user with their credit score, the banking mobile application will integrate with Experian Connect API to return the user's credit score from the Experian Credit Bureau. This will be managed with Azure API Management. The JSON code will run in Azure Functions and be invoked when a user requests a credit score check from within the application. Once invoked, the application will send a request to Experian through their Experian Connect API and receive back the reply which will pass through Azure API Management and then be displayed to the user within the application.

The banking application will utilize the Geolocation API from the Google Map platform to acquire the phone’s location data and send it to an Azure SQL database where the data will be stored. The JSON code will run in Azure Functions and be invoked when a user clicks/activates the location look-up function within the banking application. Once invoked, the application will connect to the phone’s location through the Geolocation API and then return the data to an Azure SQL database using Azure’s internal API Gateway to connect.

# Deployment Plan

Developing the cloud backend will consist of the following:

* Data Storage build out using Azure Storage for permanent uploaded files storage, static files, static website storage, and Azure SQL DB. Estimated time to completion for the Data Storage phase is 13 days at a cost of $4,880 for backend engineer labor.
* User Management build out using Azure Active Directory and creating all associated security policies for user accounts and security groups. Estimated time to completion for this phase is 10 days at a cost of $3,360 for engineer and administrative labor.
* Server-Side Logic build out in Azure Functions Apps and Azure Notification Hub. Estimated time to completion is 12 days at a cost of $6,240 for software engineer labor.
* Data Integration build out with Azure SQL DB, Azure Storage, Azure Functions, and Azure API Management. Estimated time to complete is 13 days at a cost of $7,488 for data and backend engineer labor.
* Total estimated costs = $21,968

The deployment for the cloud application will consist of using Azure Container Registry that will allow for the storing and management of private Docker container images. Azure Container Registry integrates easily with Azure Kubernetes Service. Azure Kubernetes Service will allow for easy deployment of new microservices or virtual machines. In the event that an application running in a Docker container fails, Kubernetes will replace it and start a new instance automatically (Krzyczkowski, 2021).

From the time the request for new backend services is initiated, it will take between ten and fifteen minutes for the new instances to be fully functional and operational.

The back-end resources and services required for the mobile banking application will consist of Azure Container Registry, Azure Kubernetes Service, Azure API Management, Azure Notification Hub, Azure Active Directory B2C, Azure SQL Database, Azure Storage Account, Azure Function Apps, and Azure Key Vault.

The estimated cost to run the application on Azure is $1,643.12/month as figured with Azure pricing calculator (*Pricing Calculator*, n.d.).

Redundancy will be handled by Azure Kubernete Service to ensure that should an application or microservice fail, it is instantly replaced to ensure the least amount of downtime possible.

The application itself will be deployed to both of the major application stores, Apple Store and Google Play Store, for both Android and iOS distribution.

# Maintenance Strategy

For short-term patch management, this will be reserved for the discovery of any zero-day security threats that are discovered for any of our application’s various systems. In the event of a zero-day discovery, a patch will be applied as soon as it is available.

For long-term patch management, we will use a delayed quarterly update schedule. When new patches become available, they will be tested in a sandbox environment for a full quarter year and only be applied on the production application if no conflicts or errors are found. If any errors or bugs are found, they will be sent as an issue to be dealt with by change management which will fall under the long-term update/redevelopment policy.

For short-term updates or redevelopment of the application, we will again reserve this for any zero-day or previously unknown security threats to the infrastructure or one of the services that our application runs on. In the event such a threat becomes known, we will take immediate action to update or redevelop the application as soon as possible.

For long-term updates or redevelopment of the application, we will use standard change management practices. All issues or proposed changes to the application will need to be brought before a change advisory board and approved before those updates or redevelopments are implemented.

# Disaster Recovery Plan

The banking application is built to be redundant and withstand the failure of any one or multiple points of failure. In the event that a service or virtual machine fails, Azure Monitoring will report the failure and Azure Function will use the Azure Kubernetes Service to launch a new identical replacement of the failed machine or service immediately.

An alert from Azure Monitor will send a report to the engineering team to address any and all causes for the failure and take steps to ensure the fault does not occur again.

In the event of a regional catastrophe or power outage, the banking application will be protected by Azure Backup and Site Recovery.

All data will be replicated and stored in different regions with Azure Backup ensuring the highest availability possible.

Azure Site Recovery will host duplicate machines and services in a separate region that will run in real-time and takeover should anything disrupt the current production services.

The system is designed so that in the event of a regional catastrophe, RTO would be less than 15 minutes in the worst case scenario, and a matter of seconds in most cases.

# Regulatory Compliance

Azure maintains certification from the majority of worldwide financial regulatory and compliance organizations such as PCI DSS, PCI 3DS, SOX, and FFIEC.

All data will be encrypted in transit and at rest. Azure Policy definitions will be used to enforce network security groups and firewalls. Azure Security Center will monitor missing endpoint protection and deploy Microsoft IaaSAntimalware for Windows Server along with Threat Detection on SQL Servers. Azure Policy will also be used to enforce Separation of Duties and management of Privileged Access Rights. Azure AD and Authentication will enforce passwords and a unique ID for each person with computer access. Azure Monitor will be used for logging and log management.

# Sources

Krzyczkowski, D. (2021, March 10). *Building a banking app in the cloud made simple with Azure*. Predica. https://www.predicagroup.com/blog/cloud-mobile-banking-application/?utm\_source=youtube

*Pricing Calculator*. (n.d.). Microsoft Azure. Retrieved May 21, 2021, from https://azure.microsoft.com/en-us/pricing/calculator/