Course Project – Implementation Documentation

Shaun Pritchard

Rasmussen College

CDA3315C

Alan Dennis

July,30 2020

**Clickable Index:**

***Ctrl + Click*** *on the index section number below to avoid having to scroll and instantly jump to the newest updated section. I will add a new link each week*

1. [Project Documentation](#_Project_Documentation)
2. [Analysis Documentation](#_Analysis_Documentation:)
3. [Design Documentation](#_Design_Documentation:)
4. [Implementation Documentation](#_Implementation_Documentation:)

|  |  |  |  |
| --- | --- | --- | --- |
| Enterprise Architecture Use-Case Change History – Rasmussen Web Application | | | |
| Name | Change description | Date of Change | Version |
| Shaun Pritchard | Added design Documentation | July 23, 2020 | 1.0.3 |
| Shaun Pritchard | Added Implementation documentation | July 30, 2020 | 1.0.4 |

# Project Documentation:

**Use case:** Project Use-Case-1.0.0

**Created by:** S.P

**Date Created:** 07-10-2020

**Actors:** Lead, Users, Subject Matter Experts (SME)

**Priority:** 1

**Project Summary:**

This project will facilitate a cloud web-based inventory system that will inventory database structures with output responses based on user input. I would like to abstract the database modeling to output the design structure for input queries based on the relationships of the modeling and layered database architecture for enterprise systems. This will be implemented throughout the project schedule.(See Project Schedule)

**Project Schedule:**

6 weeks is proposed based on the following schedule to complete the proposed enterprise architecture system implementation.

|  |  |  |
| --- | --- | --- |
|  | **Project Schedule** | |
| **Description** | **Action** |
| **Week 1** | Use-Case | Create New use-case and define application purpose. |
| **Week2** | Modeling baseline & Database | Define software & system requirements. Model database |
| **Week3** | Web & interface development | Create web components and interfaces to communicate with backend |
| **Week4** | Testing | Document the user acceptance testing environment |
| **Week 5** | Deployment | Document the definition of scope creep create maintenance documentation |
| **Week 6** | Security & reporting | Enhance the project documentation |

**Project Components**

Will explain and consider the risks and analysis for software requirements, hardware requirements, staffing, timelines, and project costs of the cloud-web based inventory system.

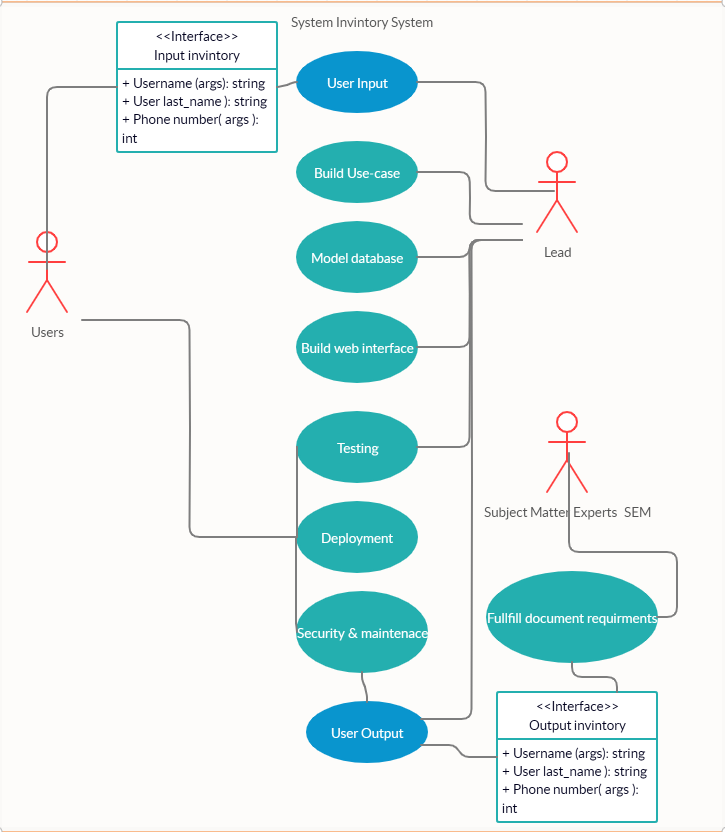
**Risk**

Time could be a factor that would threaten the projects success variants of details and complexity could be cause. Also, unseen changes and constraints could pose factors that would hinder the timeline in question for the project’s completion. This complexity and constraints could come from database modeling and building interfaces to the front-end web application.

**Analysis**

Overall, there is not enough data to propose a full analysis of the overall project. Based on system constraints and abstraction the web application seems to be a fairly easy task with minimum milestones which can be easily implemented.

**Use case diagram:**



# Analysis Documentation:

**Business Rules:**

Business rule is meant to help everyone understand the logic of each area of the service or product that is being developed. These rules will apply to a CRUD based inventory system for student registration at Rasmussen College. Business rules are denoted with (**BR**-#)

**BR-1** Users will be able to access a responsive web-based interface capturing user input from 3 input field types  First\_Name, Last\_Name and Telephone\_number to register to Rasmussen College.

**BR-2** First\_Name, Last\_Name and Telephone number input data will be stored in MySQL database backend hosted on an Azure server.

**BR-3** PHP scripting will implement database access, query, and logic between backend MySQL database and the user input data.

**BR-4** An HTML form input “Submit” button will be accessible below the main user input fields to capture input user data to facilitate the query of the MySQL database where user input data will be stored via the PHP logic.

**BR-5** Below the input form and “Submit” button on the user form a 4-column output table will be generated based on the PHP code logic that will access the stored data from the MySQL database; displaying the data fields plus the active MySQL database the current data is stored in for users to visualize.

**BR-6** Below the output table area of the interface there will be a red JavaScript input button called “Delete” implemented through PHP scripting that will delete the latest row from the given output table and the MySQL data base to Unregister a submitted student, omitting their name from the list.

**BR-7** Below the “Delete” button will be another script bas generated table that will show all current and active tables in the MySQL data base through logic that will read and update the actual table programmatically. Any new tables added to the database will be added to this table after page. This will occur from Update asynchronous logic that will check and update the current tables through a procedurally script set Time out function. This will facilitate the “Update proponents of the CRUD implementation.

**BR-8** Below the Active database table there will be another form button that will generate an active alert with current database input data of registered users for READ Only implementation of the CRUD app.

**BR-9** Interface will be styled with bootstrap being responsive to mobile devices and have a clean appeal to all users.

**System Requirements:**

**Functional requirements** *(denoted as* ***F****- plus number)*

1. Setup and install web app service with MySQL on Azure service.
2. Create new MySQL database tables.
3. Build reporting table and load data.
4. Create a new PHP based web page user interface.
5. Create PHP connection strings.
6. Write a PHP process database activity.
7. Write CSS syntax to format web user interface.
8. Publish PHP based web application to Azure cloud services.
9. Use developer tools to modify and fix inefficiencies with the web application.
10. Code base used for user interface will consist of HTML, CSS, JavaScript, PHP and MySQL
11. Web server will be developed as a service using Azure
12. MySQL database will be implemented and set up through Azure service.

**Behavioral requirements** *(denoted as* ***B****- plus number)*

1. Users form to register students **First\_Name** input will be stored in MySQL web server database back end first table field.
2. Users form to register students **Last\_Name** input will be stored in MySQL web server database back end first table field.
3. Users form to register students **Telephone\_number to** input will be stored in MySQL web server database back end first table field.
4. User will press “Submit button to query database and store data input values into the MySQL database selected table.
5. After user submission is handled by the “Submit” button. the output table below the button will populate with current **active\_database\_table** name value and the submitted user input form values, per row.
6. User will be able to press the “Delete button will be under the output table with the ability to delete the latest submission of data in the output table and the active database table in MySQL.
7. User will be able to view the Active database Table directory below the “Delete” button to show all active tables in the current MySQL database back-end implementation.
8. User will be able to press a readme submit button to output all database values in the output table to read only an alert message as read only data.

**Requirement Priorities:**

This section will outline the business requirements as well as their importance to the project. Priority ratings should include critical, high, medium, low, and future. Begin with critical needs and work down to low and future requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| Priority | Description | Rationale | Stakeholders |
| *Priority level* | *Description of the requirement* | *Why is requirement included?* | *Departments or teams impacted* |
| High | **BR-1** | Register users | IT admin |
| High | **BR-2** | Register users | IT admin |
| High | **BR-3** | Register users | IT admin |
| Med | **BR-5** | Register users | IT admin |
| Med | **BR-7** | User interface | IT admin |
| High | **F-1** | Implement input data | Developers |
| High | **F-4** | Implement input data | Developers |
| Med | **F-5** | Implement input data | Developers |
| Low | **F-7** | Database implementation | Developers |
| High | **F-8** | Frontend development | Developers |
| High | **B-1** | Inventory create data | User & Admin |
| High | **B-2** | Inventory create data | User & Admin |
| High | **B-3** | Inventory create data | User & Admin |
| Med | **B-5** | Inventory update data | User & Admin |
| Med | **B-6** | Inventory delete data | User & Admin |
| Low | **B-8** | Inventory read data | User & Admin |

# Design Documentation:

**Design Documentation:**

1. Cloud services needed for this project
2. Flow chart of the course lab application
3. Implement a historical change grid at the top of the document

**Cloud service implementation requirements:**

The system components needed for this web application are listed below denoted by (CS) for cloud services. These tools will allow for testing through setting up virtual environments, component-based testing, API testing, performance testing, user integration, authentication, mock services and user data for testing ETP, and other enterprise services to facilitate the end product.

CS-1 Azure cloud database service – for this project we will need to use integrated cloud tools for testing and distributing data for the web application.

CS-2 Azure DevTest Lab – used to provide virtual machine access for testing, managing cost, enabling self-service, automation, scaling and proof of concept need for the web application.

CS-3 Azure Files - offers fully managed file shares in the cloud that are accessible via the industry standard Server Message Block (SMB) protocol. Azure file shares can be mounted concurrently by cloud or on-premises deployments of Windows, Linux, and macOS.

CS-4 Azure Active Directory B2D - (Azure AD) enterprise identity service provides single sign-on and multi-factor authentication to create test users.

CS-5 Microsoft Graph - will allows the management of user accounts in your Azure AD B2C directory providing create, read, update, and delete methods in the Microsoft Graph API.

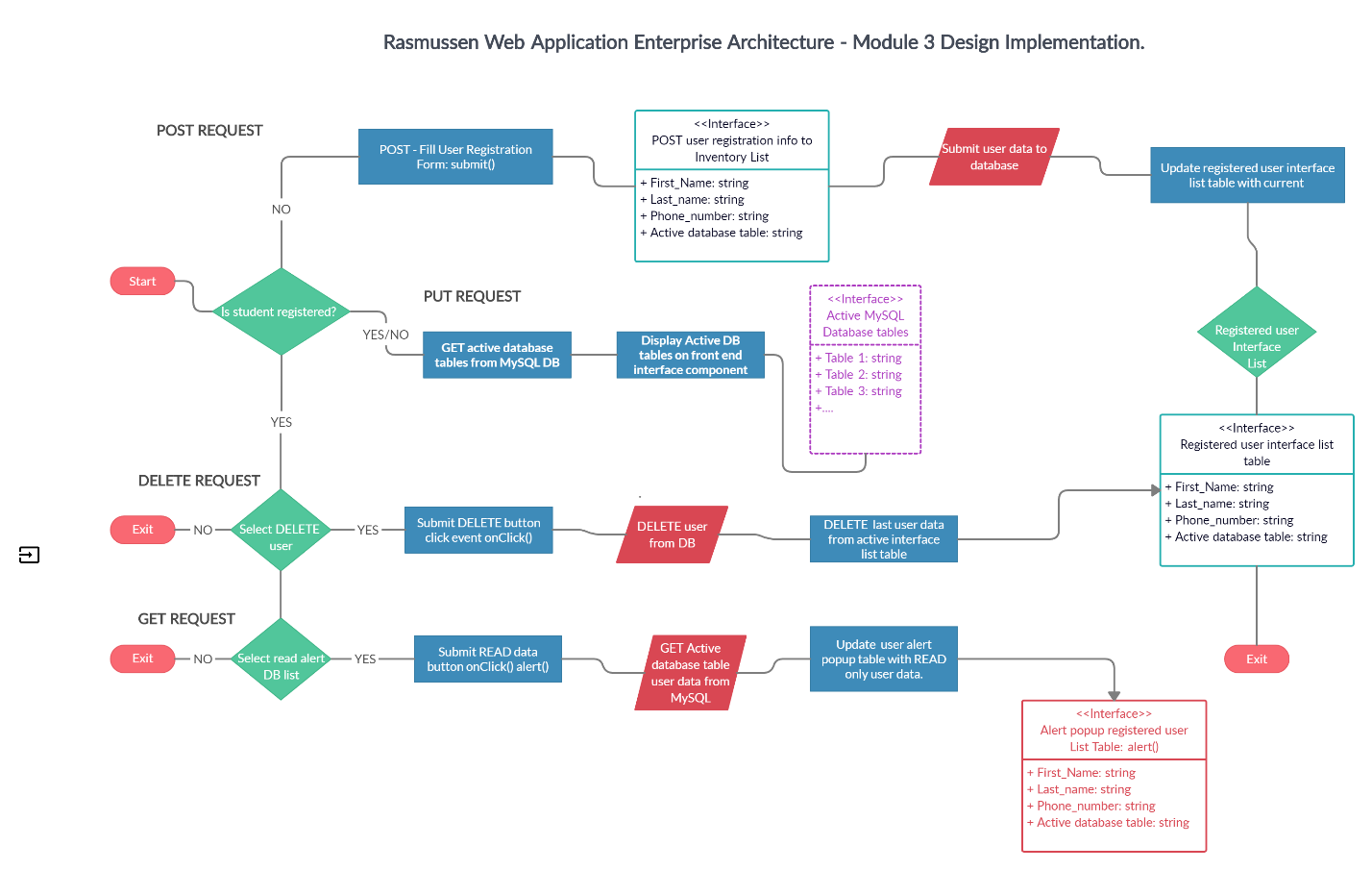
CS-6 Google Cloud Identity – Another authentication API to use for testing provisions of up unlimited email, groups, and user request.

CS-7 Blaze meter - provides components needed for the entire continuous testing process with mock services, performance testing, API monitoring, and functional testing

CS-8 User testing – a cloud-based test platform used to create real time feedback from real users to implement feedback data on the overall design aspects such as interfaces and functionality of the application.

[[1]](#footnote-1)

**Flow chart of the course lab application:**



# Implementation Documentation:

**Developer testing environment requirements:**

Here are the developer design and implementation specifications for local testing environment needed for the current project. Specifications are denoted by (DT-1…)

DT-1 Initial logic test - test script procedure will need to validate through console.log frontend registration form input values are being submitted through script logic component.

DT-2 Initial design test -Browser test using dev tools to determine if user interface and data modules are responsive

DT-3 Initial logic test - the user interface registration form test will need to show connectivity to database submission posting input values to specified labeled inputs testing persistence POST functionality to the database tables being queried.

DT-4 Initial logic test - Test procedure to test persistence and DELETE data from user submissions from the backend.

DT-5 Initial logic test - Test procedure to test READ capabilities of persisted data allocated in user database and its tables.

DT-6 Initial logic test - Test procedure to test UPDATE capabilities of persisted data allocated in user database and its tables.

DT-7 Initial design test -Browser test functionality of Delete buttons to determine if user interface and data modules connecting to logic.

DT-8 Initial design test -Browser test functionality of Read buttons to determine if user interface and data modules connecting to logic. Test should include console log or alert event to display persisted user data.

DT-9 Initial design test -Browser test functionality of user data should be displayed in company modules to show list table of registered users.

DT-10 Initial design test -Browser test functionality of user data should be displayed in company modules to show active database tables.

**Developer UAT (User acceptance) requirements:**

Here are the procedures for the team developers to outline the UAT systems of the user acceptance testing environment to implement duplication of production application for user test and acceptance of the current progress and shareholder acceptance of the project. Denoted as (UAT-1).

UAT-1 Initial design user test – will be a replica of the functioning interface input form. Developer should push user interface module to get shareholder approval.

UAT-2 Initial design user test – will be a replica of the functioning interface input form plus the list data. Developer should push user interface module to get shareholder approval.

UAT-3 Initial design user test – will be a replica of the functioning interface input form plus the table references and display active database tables. Developer should push user interface module to get shareholder approval.

UAT-4 Initial design user test – will be a replica of the functioning full interface input form, list tables, and in its entirety. Developer should push full mockup of production ready user interface module to get shareholder approval.

UAT-5 Initial persistence test – display the functionality of user persistence input data should be submitted to the backend then displayed in the list table on the front end based on the full mockup of production ready user interface module to get shareholder approval.

UAT-6 Initial logic test – display the functionality of user persistence of all input data fields and all button events action to show the creation, read, update, and delete functionality of the application. This should consist of a full mockup of the production ready user full interface to get shareholder approval on workflow and usability. Testing will not include browser load time capabilities or refresh rate and optimized performance.

**Four Week project Schedule:**

This project schedule is defined for System administrator and his/her team to delegate the determine timelines, expectations, and other project details

**Week1:**

|  |  |  |  |
| --- | --- | --- | --- |
| System Task | Description of task | Timeline | Project details |
| Software application design | Develop full schema, design, and lifecycle of application | 5 days | Rasmussen inventory application Software architect |
| Services and resources | Set up all services and resources to host application | 3 days | Rasmussen inventory application Systems administrator. |
| Use-case | Design use case for project | 2 days | Rasmussen inventory application Software architect |
| Database design | Complete setup of MySQL data base with service to host application and delegated tables for user registration and backup will need to be created | 2days | Rasmussen inventory application database development |
| Form design | Complete frontend form design | 1 days | Rasmussen inventory application design |
| Form logic modules | Write logic to connect form and persist to database | 2 days | Rasmussen inventory application logic development |
| Test scripts | Build test scripts to define logic modules | 1day | Rasmussen inventory application test scripts |
| Tables design | Design tables for registered user output display | 1 day | Rasmussen inventory application frontend design |
| Module output design | Create the frontend modules to display the active user databases | 1 day | Rasmussen inventory application script logic module for output to database table |
| 1-week project overview | Ensures all task delegated to team are complete and on track and coordinates with system administrator for application requirements. | 4 days | Rasmussen inventory application project management. |

**Week2:**

|  |  |  |  |
| --- | --- | --- | --- |
| System Task | Description of task | Timeline | Project details |
| Frontend CRUD buttons | Frontend development and styling of the buttons on the front user end | 1 day | Rasmussen inventory application design |
| Module output logic | Modular scripting logic will need developed to output current register user data to the frontend interface | 1 day | Rasmussen inventory application logic development |
| Module user test | Use test scripts will need to be written for the module output logic | 1 day | Rasmussen inventory application logic development |
| Tables logic | Modular scripting logic will need developed to output current active database tables to frontend interface | 2 day | Rasmussen inventory application logic development |
| 2-week project overview | Ensures all task delegated to team are complete and on track | 4 days | Rasmussen inventory application project management. |

**Week3:**

|  |  |  |  |
| --- | --- | --- | --- |
| System Task | Description of task | Timeline | Project details |
| Read alert | Develop alert function to capture and READ data from the database | 1 day | Rasmussen inventory application logic development |
| Update module | Create update module logic to update current user with PUT | 1 day | Rasmussen inventory application logic database development |
| Delete module | Create DELETE functionality module to delete registered user from the data base. | 2 day | Rasmussen inventory application logic database development |
| Delete testing | Test script for test bench to validate persistence and delete of registered user. | 1 day | Rasmussen inventory application logic database development |
| Testing database | Full testing data base will need to be duplicated for UAT with clear copy of all functional tables | 2 days | Rasmussen inventory application logic database development |
| 3-week project overview | Ensures all task delegated to team are complete and on track | 4 days | Rasmussen inventory application project management. |
| Insights | Ensure all team members are in compliance and overview current status of project with software architect | 3 days | Rasmussen inventory application Software architect |

**Week4:**

|  |  |  |  |
| --- | --- | --- | --- |
| System Task | Description of task | Timeline | Project details |
| final frontend task | Frontend developers will need to meet with project manager review final notes and make sure all design guidelines are met | 1 day | Rasmussen inventory application frontend development & design |
| Final testing logic | Database developers will run test scripts to ensure database is being persisted properly and efficiently. | 1 day | Rasmussen inventory application logic database development |
| Database replication test | Systems administrator will need to verify the database has been copied and ready for UAT testing | 2 day | Rasmussen inventory application System Administrator |
| Full application bed test | Developers will need to review all project manager and WBS task to allocate final completion on the project. | 1 day | Rasmussen inventory application developers |
| Submit & review for production release | Submit to project manager and scrum final notes before application is deployed | 3 days | Rasmussen inventory application System Administrator, project manager, and Software architect |

**WBS (Work break down structure):**

This WBS will outline the deliverables for the current project and break down the task need to complete the project and designated timeframes and cost for each delegated task. Please see the table below

|  |  |  |
| --- | --- | --- |
| **Task Name** | **Duration** | **Assigned** |
| Develop application lifecycle and design. | 5 days | Software architect |
| Develop Use case and PM documentation. | 2 days | Software architect |
| Set up resources and services to host application. | 5 days | Systems administrator |
| Design the Form UI. | 4 days | Frontend developer |
| Build UI for output registered users. | 2 days | Frontend developer |
| Build UI modules to display active databases. | 3 days | Frontend developer |
| Build UI to handle CRUD capabilities. | 3 days | Frontend developer |
| Ensure deign is on schedule. | 2 days | Project manager |
| Develop and Design database for application and set up DB resources. | 6 days | Database developer |
| Implement logic to submit user inputs from UI. | 5 days | Backend developer |
| Implement logic to read back submitted users to a list table on the UI. | 4 days | Backend developer |
| Implement logic to persist and update user data from the UI. | 3 days | Backend developer |
| Ensure project is on track and UAT test have been developed. | 2 days | Systems administrator |
| Implement modular logic to GET active database tables. | 3 days | Backend developer |
| Develop UI alert pop up and logic to display READ data to user of registered and deleted users. | 4 days | Frontend developer |
| Add active databased new user submit is stored in in output UI list. | 5 days | Database developer |
| Ensures all task delegated to team are complete and on track. | 6 days | Project Manager |
| Collaborate with developers to go over flaws and over new findings or abilities in the software. | 5 days | Software architect |
| Ensure database has been replicated for UAT testing. | 4 days | Systems administrator |
| Ensure application meets full CRUD specifications of the user . | 2 days | Backend developer |
| Ensure test bed is set up. | 4 days | Backend developer |
| Ensure all workbench test have been completed. | 3 days | Project Manager |
| Ensure UAT is ready for user to demonstrate. | 4 days | Systems administrator |
| Ensure all task have been complete for application deployment. | 6 days | Project Manager |
| Complete final analysis on application based on team data. | 4 days | Software architect |
| UAT testing Final | 1 day | User |

**Project users and roles**

Here we will define the specific roles and purpose being performed on this current project user roles for specific variation and responsibilities of implementation for the current project. This will describe their roles and provide details about what they will contribute to the project.

|  |  |  |
| --- | --- | --- |
| Role | Description | Project Phase week |
| System administrator | Is responsible for the upkeep, configuration, and reliable operation of computer systems in which the application will be implemented on including its databases and resources. | 1,2,3,4 |
| Software architect | Is the lead software developer expert who makes high-level design choices and tries to enforce technical standards, including software coding standards, tools, and platforms. | 1,2,4 |
| Front end developer | Is the developer responsible for the interface and other applicable design elements of the application that users engage with. | 1,2,3,4 |
| Backend developer | Is responsible for server-side web application logic and integration of the work front-end developers do. They will implement web services and APIs used by front-end developers and develop the logic to connect to the database. | 1,2,3,4 |
| Project manager | Will be in charge of planning and leading software project. | 1,2,3,4 |
| Database Developer | Database developers ensure that database management systems (DBMS) can handle massive quantities of data and are responsible for integrating the frontend logic and maintaining logic to the database tables. | 1,2,3,4 |
| User | User is the shareholder and or the originator of the application, or other users who will test and approve each phase through the UAT of the application. The user interacts with the application and uses it. | 1,3 |

# References

Azure Active Directory B2C. (2020). *Azure Active Directory B2C*. Retrieved from https://azure.microsoft.com/en-us/services/active-directory/external-identities/b2c/: https://azure.microsoft.com/en-us/services/active-directory/external-identities/b2c/

Azure NetApp Files. (2020). *Enterprise File Storage, Powered by NetApp*. Retrieved from https://cloud.netapp.com/azure-netapp-files?: https://cloud.netapp.com/azure-netapp-files?

Blaze Meter. (2020). *Mock Services*. Retrieved from https://guide.blazemeter.com/hc/en-us: https://guide.blazemeter.com/hc/en-us

Google Cloud . (2020). *Federating Google Cloud with Azure Active Directory: Configuring provisioning and single sign-on*. Retrieved from https://cloud.google.com/: https://cloud.google.com/architecture/identity/federating-gcp-with-azure-ad-configuring-provisioning-and-single-sign-on

Microsoft Azure. (2020). *Azure Active Directory*. Retrieved from https://azure.microsoft.com/: https://azure.microsoft.com/en-us/services/active-directory/?&ef\_id=Cj0KCQjw6uT4BRD5ARIsADwJQ1\_3NPNOA0tOwSD00HuMrKxSownRFv8rXZ8l9Fuz63o2W5010yxBR50aAvMkEALw\_wcB:G:s&OCID=AID2100131\_SEM\_Cj0KCQjw6uT4BRD5ARIsADwJQ1\_3NPNOA0tOwSD00HuMrKxSownRFv8rXZ8l9Fuz63o2W5

Microsoft DevTest Labs. (2020). *DevTest Labs*. Retrieved from https://azure.microsoft.com/en-us/services/devtest-lab/: https://azure.microsoft.com/en-us/services/devtest-lab/

User Testing. (2020). *Real-time feedback.* Retrieved from https://www.usertesting.com/: https://www.usertesting.com/

Modern Analyst.(2020).Business rules Retrieved from.

<https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/1442/Use-Cases-and-Business-Rules-Can-They-Work-Together.aspx>

References

1

1. (Microsoft Azure, 2020) (Microsoft DevTest Labs, 2020) (Azure NetApp Files, 2020) (Azure Active Directory B2C, 2020) (Google Cloud , 2020) (Blaze Meter, 2020) (User Testing, 2020) [↑](#footnote-ref-1)