Capstone Title - Cloud-Based Inventory Management System for Medical Equipment Supply

Company - Ultratheraputics

Domain - Cloud Computing Azure

Expertise Level - Beginner to Intermediate

Problem Statement

Ultratheraputics needs a scalable cloud-based system for managing and tracking medical equipment inventory across multiple locations. This project involves creating an Azure-hosted inventory management application that supports secure login, data storage, and alerts for low-stock items, helping MediSupplies Inc. optimize inventory and maintain a steady supply of essential medical equipment.

Learning Objectives

- Set up basic authentication and access control with Azure Static Web Apps.
- Store and manage inventory data using Azure Table Storage.
- Handle images and files for inventory items with Azure Blob Storage.
- Develop CRUD API endpoints using Azure Functions.
- Set up automated alerts for low-stock items using Azure Logic Apps.
- Design a basic inventory dashboard interface with HTML and CSS.

Business Objectives

Streamline inventory management across multiple locations, enabling Ultratheraputics. to efficiently track stock levels, identify low-stock items, and improve overall operational efficiency, ensuring a consistent supply of critical medical equipment.

Dataset Link

No dataset provided. Students will create sample data representing inventory records and equipment details.

Prerequisites

- Basic knowledge of Azure services (Static Web Apps, Table Storage, Blob Storage, Logic Apps).
- Basic knowledge of HTML/CSS and REST APIs.

Platforms/Tools - Azure Portal, GitHub, Visual Studio Code (optional)

Number of Tasks - 8

Task Dependency - Yes

Tasks

1. **Task 1: Set Up Authentication and Access Control**
    - **Task Name: Task Name**: Authentication and Access Control with GitHub Pages and Firebase Authentication
    - **Background Description:**Managing access is crucial to ensure only authorized personnel can access and update inventory information. GitHub Pages provides a free hosting solution for static websites, while Firebase Authentication can be used to manage secure logins.
    - **Skills:** Azure Static Web Apps, Authentication
    - **Task Instruction:** n GitHub, set up a **GitHub Pages** deployment for your project repository.
    - Use **Firebase Authentication** to enable secure login for your app. Follow the setup guide to integrate Firebase Authentication with your HTML/JavaScript application.
    - Configure Firebase Authentication to allow only signed-in users to access the site.
    - Test the login to make sure users need to sign in before accessing any page.
    - **Submission Format:** Screenshots of the GitHub Pages deployment setup, Firebase Authentication settings, and a short description of how the authentication flow works.
    - **Resources:**
        - GitHub Pages Documentation
        - Firebase Authentication Documentation
    -
2. **Task 2: Create a Table Storage Schema for Inventory Data**
    - **Task Name:** Inventory Data Schema Creation with Azure Table Storage

- **Background Description:** Organizing data efficiently is essential for inventory management. Azure Table Storage provides a cost-effective way to store structured data, ideal for tracking inventory details.
- **Skills:** Azure Table Storage, Data Organization
- **Task Instruction:** Log in to Azure, navigate to Storage Accounts, and create a new storage account. In the storage account, select "Tables" to create a new table named "Inventory." Define columns such as item ID, item name, location, quantity, last updated date, and restock threshold. Add a few sample entries to test that data is being stored correctly. Ensure the schema and column names are consistent with your inventory needs.
- **Submission Format:** Screenshots of the Table Storage schema and sample entries.
- **Resources:** Azure Table Storage Documentation

3. **Task 3: Develop Basic API Endpoints for CRUD Operations**
   - **Task Name:** CRUD API Endpoints with Azure Functions
   - **Background Description:** An API enables interaction between the front end and backend data. Using Azure Functions to create CRUD endpoints will allow users to manage inventory.
   - **Skills:** Azure Functions, API Development
   - **Task Instruction:** Use Azure Functions to create endpoints that perform CRUD operations. Set up a “Create” function to add new inventory items, a “Read” function to view item data, an “Update” function to modify item details, and a “Delete” function to remove items. Each function should interact with Azure Table Storage to retrieve or update the data. Write the functions in JavaScript or Python, ensuring each function correctly handles incoming requests and returns responses. Test each function locally or in the cloud using sample requests to verify functionality.
   - **Submission Format:** API code and screenshots of test requests demonstrating each CRUD operation.
   - **Resources:** Azure Functions Documentation
4. **Task 4: Store Equipment Images and Files in Blob Storage**
   - **Task Name:** Image and File Management with Azure Blob Storage
   - **Background Description:** Storing images and manuals allows easy reference for each inventory item. Azure Blob Storage is ideal for storing large files securely.
   - **Skills:** Azure Blob Storage, File Management
   - **Task Instruction:** In Azure, open your Storage Account and navigate to “Blob Service” > “Containers.” Create a new container (folder) called “InventoryImages.” Upload sample images (such as sample product photos) to this container. For each item, create an identifier (like the item ID) to name the images or to help group images by item type. Ensure each image can be accessed by its unique URL for later use in the front end.
   - **Submission Format:** Screenshots of Blob Storage container setup, sample images uploaded, and URLs generated for these images.

    - **Resources:** Azure Blob Storage Documentation
5. **Task 5: Design a Low-Stock Alert System with Azure Logic Apps**
    - **Task Name:** Automated Low-Stock Alerts with Azure Logic Apps
    - **Background Description:** Timely restocking is essential for smooth operations. Azure Logic Apps can automate notifications for low-stock items, helping ensure continuous availability.
    - **Skills:** Azure Logic Apps, Automation
    - **Task Instruction:** Open Azure Logic Apps and create a new Logic App. Set up a workflow that connects to your Table Storage and checks inventory levels daily. Define conditions that identify "low stock" (e.g., if quantity is below the restock threshold). If a low-stock item is detected, set up an action to send an automated email notification to a specified address. Use sample email addresses and a test item to verify the alerts.
    - **Submission Format:** Screenshot of the Logic App workflow configuration and a sample email alert received.
    - **Resources:** Azure Logic Apps Documentation
6. **Task 6: Build a Simple Front-End for Viewing and Updating Inventory**
    - **Task Name:** Front-End Inventory Interface with HTML/CSS
    - **Background Description:** A user-friendly interface allows easy interaction with the inventory system. Using HTML/CSS, students will create a basic front end for viewing and managing inventory data.
    - **Skills:** HTML, CSS, Basic Front-End Development
    - **Task Instruction:** Create a basic webpage with HTML and CSS that allows users to view and update inventory items. The page should display inventory items in a simple table format, showing fields like item name, quantity, and location. Add "Edit" and "Delete" buttons that trigger the CRUD API endpoints. Use CSS to format the table for readability. Test that clicking the buttons successfully triggers the API functions created in Task 3.
    - **Submission Format:** Code for the webpage and screenshots of the interface.
    - **Resources:** HTML/CSS Basics
7. **Task 7: Implement a Basic Inventory Dashboard**
    - **Task Name:** Basic Inventory Dashboard with HTML/CSS
    - **Background Description:** A dashboard provides an overview of inventory data, helping with decision-making. Using HTML and CSS, students will create a simple dashboard that displays key metrics such as total items, low-stock items, and recent restocks.
    - **Skills:** HTML, CSS
    - **Task Instruction:** Design a dashboard using HTML/CSS to display inventory metrics like the total number of items, low-stock items, and recently restocked items. Create a basic layout, adding color and icons for an organized look. Integrate this dashboard as a separate view or page linked to your main inventory table from Task 6. Test to ensure the dashboard updates when data changes in Table Storage.
    - **Submission Format:** Code for the dashboard and screenshots of the display.

- **Resources:** HTML/CSS Basics

8. **Task 8: Create Documentation and a User Guide for Application**
    - **Task Name:** Application Documentation and User Guide
    - **Background Description:** Documentation helps users and future developers understand and maintain the application. This guide should provide instructions for navigating the app and using the API.
    - **Skills:** Documentation, Technical Writing
    - **Task Instruction:** Write a user guide that describes how to log in, view and update inventory, and understand the dashboard metrics. Include setup instructions, details about each feature, and screenshots. Explain each step with clarity for easy understanding, assuming a beginner level of familiarity with the app.
    - **Submission Format:** PDF document with the guide, including screenshots for each feature.
    - **Resources:** Technical Writing Guide