Title: File Upload and Storage in Azure DB for a 3D Building Layout Web App:

Research, Parser-Editor Linkage, and Development Initiation

Abstract:

This research paper aims to explore the utilization of Azure services for file upload and storage within a web application designed to generate 3D building layouts from multiple floor plans. The project entails developing a web application that accepts various floor plans, processes them to generate a 3D building layout, and integrates navigation features. Central to this endeavor is the storage and retrieval of uploaded files securely and efficiently within Azure's ecosystem. This paper investigates the integration of Azure services with a React application deployed on Azure, focusing on file upload, storage, and subsequent parsing, and editing functionalities.

I. Introduction:

The development of a web application capable of generating 3D building layouts from multiple floor plans is a complex task requiring robust backend infrastructure and efficient file handling mechanisms. Azure, Microsoft's cloud computing platform, offers a suite of services that can facilitate this process, from file storage to serverless computing and beyond. This paper delves into the research and development aspects of integrating Azure services with a React application deployed on Azure to achieve the desired functionality.

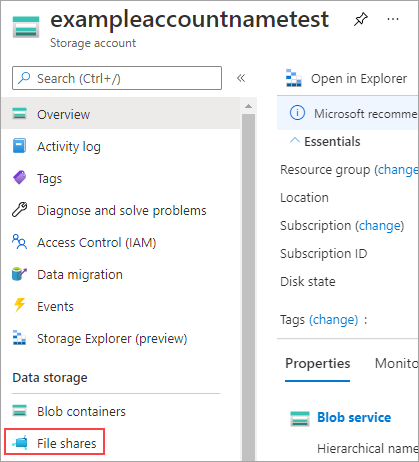
II. Azure Services for File Upload and Storage:

Azure provides several services suitable for file upload and storage, including Azure Blob Storage, Azure Files, and Azure SQL Database. Azure Blob Storage is particularly well-suited for storing large amounts of unstructured data, making it an ideal choice for hosting floor plan files. Azure Files, on the other hand, offers fully managed file shares in the cloud, providing a scalable and accessible solution for file storage. Additionally, Azure SQL Database can be utilized for structured data storage, such as metadata associated with uploaded floor plans.

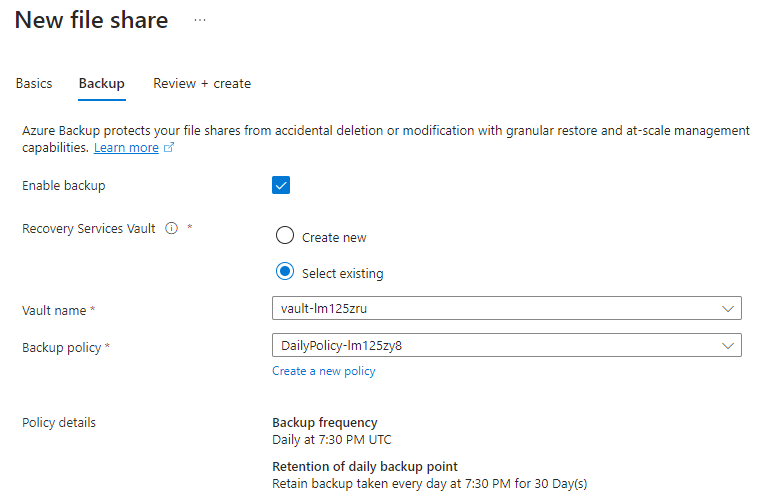
Steps: Using Azure File Share

1. Create a storage account (team already finished this step)
2. Create an Azure File Share:

* Select storage account.
* In Data storage, select File shares.



* On the menu at the top of the File shares page, select + File share. The New file share page drops down.
* In Name, type myshare. File share names must be all lower-case letters, numbers, and single hyphens, and must begin and end with a lower-case letter or number. The name can't contain two consecutive hyphens.
* Leave Transaction optimized selected for Tier.
* Select the Backup tab. By default, [backup is enabled](https://learn.microsoft.com/en-us/azure/backup/backup-azure-files) when you create an Azure file share using the Azure portal. If you want to disable backup for the file share, uncheck the Enable backupcheckbox. If you want backup enabled, you can either leave the defaults or create a new Recovery Services Vault in the same region and subscription as the storage account. To create a new backup policy, select Create a new policy.



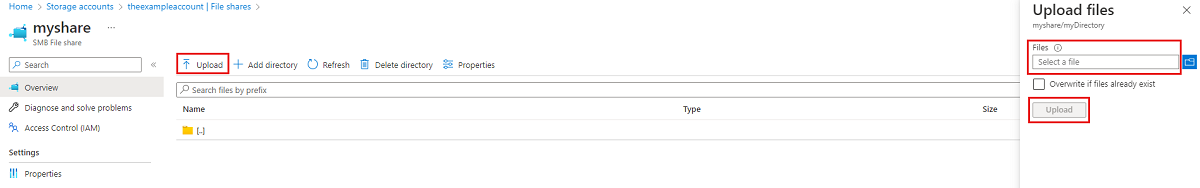
* Select Review + create and then Create to create an Azure file share.

1. Create a directory:

* create a new directory named myDirectory at the root of your Azure file share:
* On the File share settings page, select the myshare file share. The page for your file share opens, indicating no files found.
* On the menu at the top of the page, select + Add directory. The New directory page drops down.
* Type myDirectory and then select OK.

1. Upload a file:

* Create or select a file to upload.
* Select the myDirectory directory. The myDirectory panel opens.
* In the menu at the top, select Upload. The Upload files panel opens.



* Select the folder icon to open a window to browse your local files.
* Select a file and then select Open.
* In the Upload files page, verify the file name, and then select Upload.
* When finished, the file should appear in the list on the myDirectory page.

NOTE: According to the project, SMB file share might be the better option for the project file upload.

More information visit website: <https://learn.microsoft.com/en-us/azure/storage/files/storage-how-to-create-file-share?tabs=azure-portal>

III. Integration with React Application Deployed on Azure:

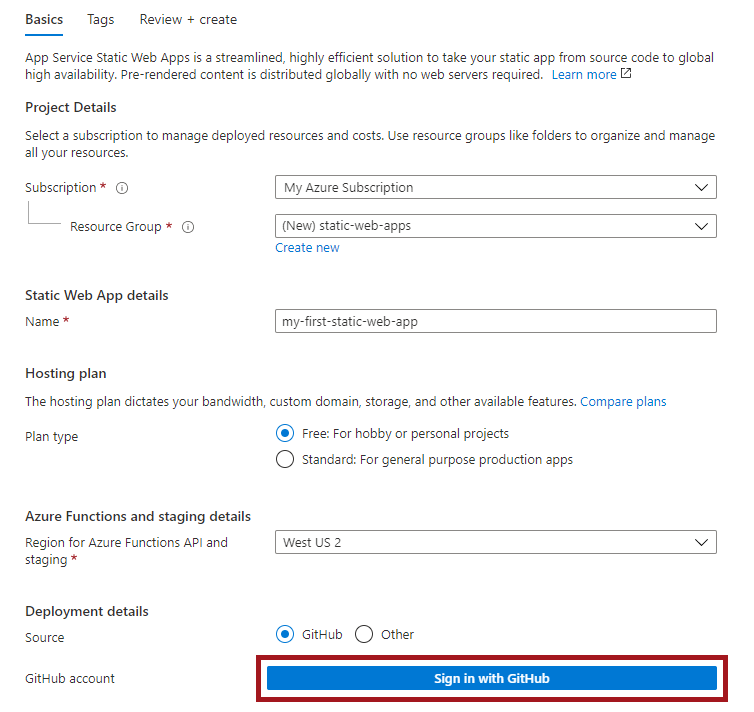
The React application, serving as the frontend of the web application, can be seamlessly integrated with Azure services for file upload and storage. Azure Blob Storage's REST API can be leveraged to upload floor plan files directly from the React application. Authentication and authorization mechanisms provided by Azure Active Directory can ensure secure access to storage resources. Furthermore, Azure Functions, a serverless compute service, can be utilized to trigger parsing and editing processes upon file upload, ensuring real-time updates to the 3D building layout.

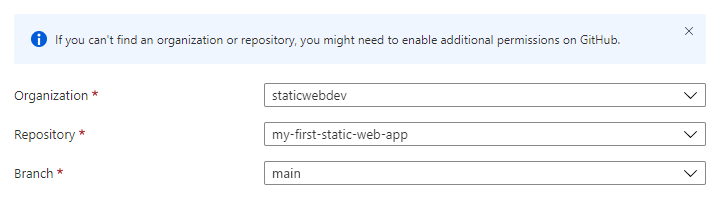
NOTE: Azure Blob Storage’s REST API resource: <https://learn.microsoft.com/en-us/rest/api/storageservices/blob-service-rest-api>

Steps to deploy React App on Azure Static Web Apps:

1. Have a repository ready.
2. Create Static Web Apps:

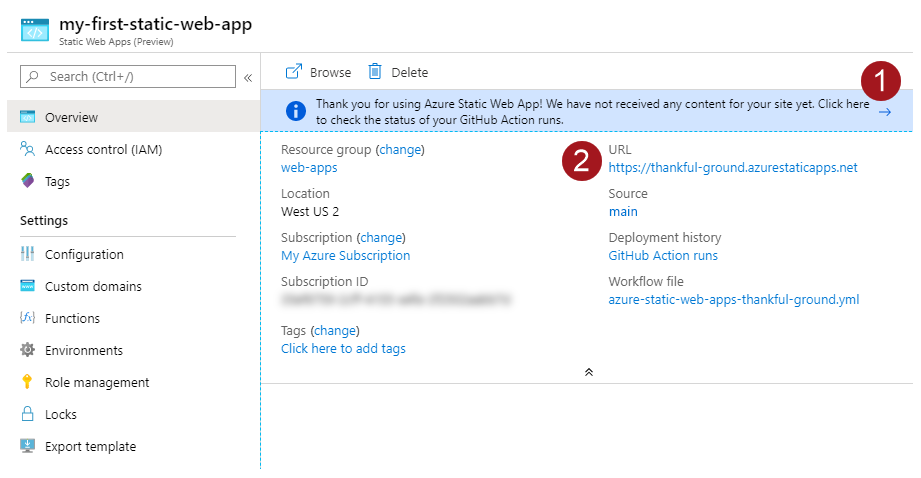
* create a static web app from the Azure portal.
* Go to the [Azure portal](https://portal.azure.com).
* Select Create a Resource.
* Search for Static Web Apps.
* Select Static Web Apps.
* Select Create.
* In the Basics section, begin by configuring your new app and linking it to a GitHub repository.





* In the Build Details section, add configuration details specific to preferred front-end framework.
* Select React from the Build Presets dropdown.
* Keep the default value in the App location box.
* Leave the Api location box empty.
* Type build in the App artifact location box.
* Select Review + create.

1. View the website:



* 3.1. Selecting on the banner that says, Select here to check the status of your GitHub Actions runs takes you to the GitHub Actions running against your repository. Once you verify the deployment job is complete, then you can go to your website via the generated URL.
* 3.2. Once GitHub Actions workflow is complete, you can select the URLlink to open the website in new tab.

IV. Parser-Editor Linkage and Development Initiation:

The parser and editor functionalities play a crucial role in processing uploaded floor plans and generating 3D building layouts. The parsed data can then be stored in Azure SQL Database for further manipulation and editing. Development initiation involves setting up Azure resources, configuring storage and database services, and implementing backend logic to handle file parsing and editing operations.

NOTE: AzureDB Resource: <https://learn.microsoft.com/en-us/azure/dms/migration-using-azure-data-studio?tabs=azure-sql-mi>

V. Conclusion:

In conclusion, Azure offers a comprehensive set of services that can facilitate the development of a web application for generating 3D building layouts from multiple floor plans. By leveraging Azure Blob Storage, Azure Functions, Azure SQL Database, and other Azure services, developers can create a scalable, secure, and efficient solution. The integration of these services with a React application deployed on Azure paves the way for seamless file upload, storage, parsing, editing, and navigation functionalities within the web application.

In summary, this research paper provides insights into the utilization of Azure services for file upload and storage within a web application aimed at generating 3D building layouts. It outlines the integration of Azure services with a React application deployed on Azure, focusing on parser-editor linkage and the initiation of development processes.