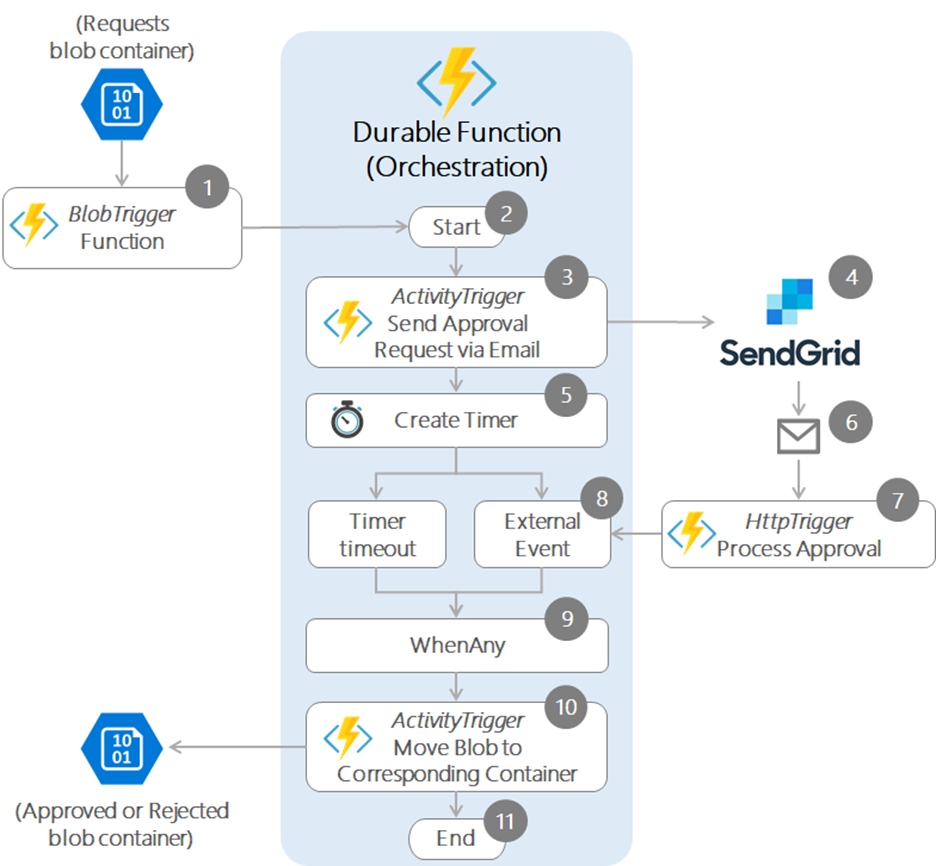
Enhanced RFP Response: Approval Workflow using Azure Durable Functions

# Architecture Diagram

Below is the updated architecture diagram illustrating the Durable Functions-based Approval Workflow:



# Enhanced Security & Access Control

✅ Use Azure AD B2C to authenticate the approver (token-based authentication from Angular frontend).  
  
✅ Use System Assigned Managed Identity for Azure Function App to securely access:  
- Azure Key Vault (for secrets and keys)  
- Cosmos DB (for approval data storage)  
- Azure Blob Storage (to move approved/rejected files)  
  
🔐 Step-by-step Implementation:  
  
1. \*\*Enable System Assigned Managed Identity:\*\*  
 - Navigate to Azure Function App → Identity → System Assigned → Turn ON → Save.  
  
2. \*\*Access Azure Key Vault:\*\*  
 - Azure Key Vault → Access Policies → + Add Access Policy  
 - Secret Permissions: Get  
 - Principal: Azure Function App  
  
 ✅ Code Example (.NET):  
 ```csharp  
 var client = new SecretClient(new Uri("https://<vault>.vault.azure.net/"), new DefaultAzureCredential());  
 var secret = await client.GetSecretAsync("SendGridApiKey");  
 var sendGridKey = secret.Value;  
 ```  
  
3. \*\*Access Cosmos DB using Managed Identity + RBAC:\*\*  
 - Cosmos DB → Access Control (IAM) → + Add Role Assignment  
 - Role: Cosmos DB Built-in Data Contributor  
 - Assign to: Function App  
  
 ✅ Code Example:  
 ```csharp  
 var credential = new DefaultAzureCredential();  
 var cosmosClient = new CosmosClient("https://<cosmos>.documents.azure.com", credential);  
 var container = cosmosClient.GetContainer("MyDB", "Approvals");  
 await container.UpsertItemAsync(item, new PartitionKey(item.UserId));  
 ```  
  
4. \*\*Access Blob Storage using Managed Identity:\*\*  
 - Storage Account → Access Control (IAM) → Assign Role: Storage Blob Data Contributor → To: Function App  
  
 ✅ Code Example:  
 ```csharp  
 var blobServiceClient = new BlobServiceClient(new Uri("https://<storage>.blob.core.windows.net"), new DefaultAzureCredential());  
 var containerClient = blobServiceClient.GetBlobContainerClient("approved");  
 await containerClient.UploadBlobAsync("myfile.txt", stream);  
 ```  
  
🧠 Note: If using BlobTrigger, the binding requires connection strings unless explicitly handled with SDK + Managed Identity.