Assignment 2 - Public Cloud

Aditya Prakash

March 17, 2025

1 Feature Added

In this project, I have implemented a **Product Catalogue** as a full-stack application deployed in a public cloud environment. The entire application runs under a single URL, handling both frontend and backend functionalities. Additionally, I have added **Image Upload for Products**, where users can upload images that are stored in **Azure Blob Storage** and displayed in the catalogue.

Implementation Steps

1. Set up an **Azure SQL Database** to store product details, including images.
2. Developed a **Flask backend** to handle API requests and interact with the database.
3. Configured **Azure Blob Storage** for storing product images and generating public URLs.
4. Built a **React frontend** to provide an interactive UI for adding and listing products.
5. Deployed the full-stack application using **Azure App Service** under a single URL.

2 Architecture Diagram

The system follows a cloud-based architecture with the following components:

- **Frontend (React)**: Manages UI and sends requests to the backend.
- **Backend (Flask)**: Handles API endpoints and communicates with the database and blob storage.
- **Azure SQL Database**: Stores product data, including image URLs.
- **Azure Blob Storage**: Stores uploaded images and provides public access URLs.
- **Azure App Service**: Hosts the full-stack application under a single URL.

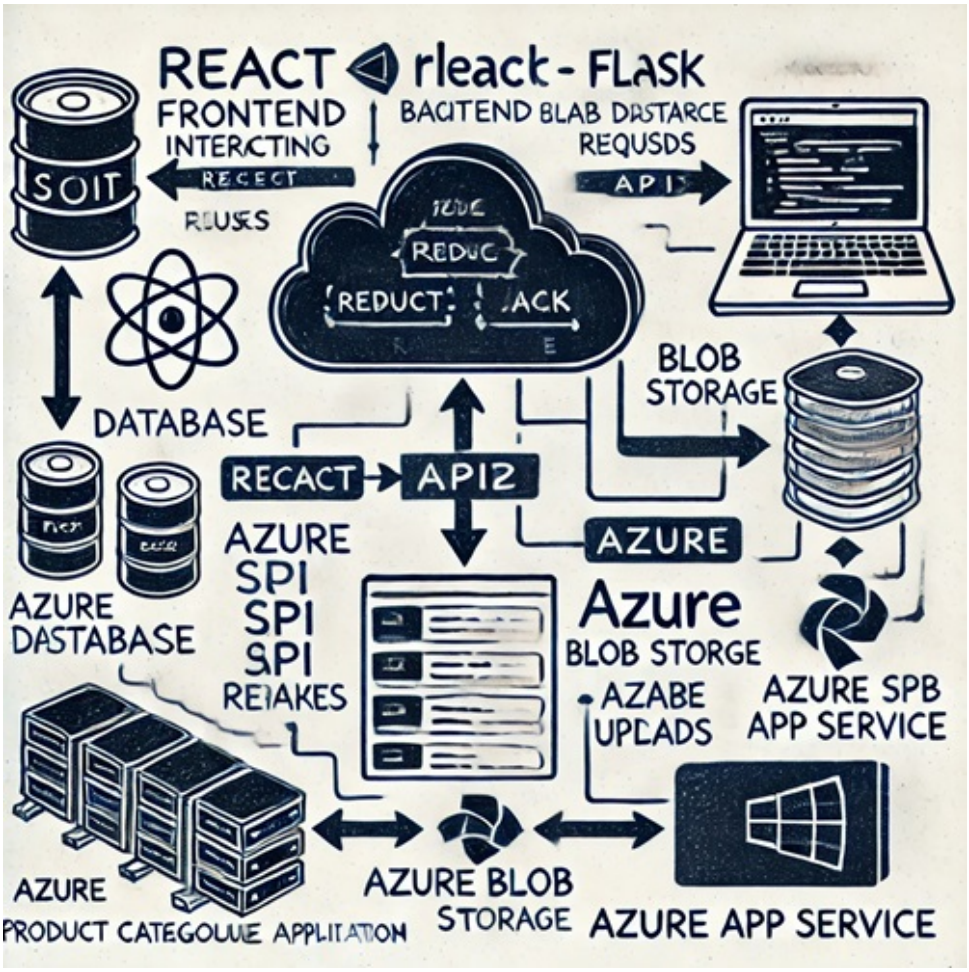

Figure 1: System Architecture Diagram

3 Architecture Decisions

Choice of Technologies

Backend (Flask)

- Lightweight and easy to integrate with Azure services.
- RESTful API for efficient client-server communication.

Frontend (React)

- Responsive and modern UI with efficient state management.
- Component-based development for modularity.

Azure SQL Database

- Fully managed relational database.
- High availability and secure data storage.

Azure Blob Storage

- Scalable and cost-effective storage for images.
- Public access URLs for easy retrieval.

4 Code Implementation

Flask Backend (API for Product Management)

```
from flask import Flask, request, jsonify
from azure.storage.blob import BlobServiceClient
import pyodbc

app = Flask(__name__)

# Database connection
conn_str = "DRIVER={ODBC Driver 17 for SQL Server};SERVER=<your-server>;DAT
conn = pyodbc.connect(conn_str)
cursor = conn.cursor()

# Azure Blob Storage
blob_service_client = BlobServiceClient.from_connection_string("<connectio
container_name = "product-images"

@app.route('/add-product', methods=['POST'])
def add_product():
    data = request.form
    file = request.files['image']
    blob_client = blob_service_client.get_blob_client(container=container_n
    blob_client.upload_blob(file)
    image_url = blob_client.url
    cursor.execute("INSERT INTO Products (name, description, image_url) VA
    conn.commit()
    return jsonify({"message": "Product added successfully!"})

if __name__ == "__main__":
    app.run(debug=True)
```

React Frontend (Product Form and Listing)

```
import React, { useState, useEffect } from 'react';

const ProductCatalogue = () => {
  const [products, setProducts] = useState([]);
  const [file, setFile] = useState(null);
  const [name, setName] = useState('');
  const [description, setDescription] = useState('');

  const handleUpload = async () => {
    const formData = new FormData();
    formData.append("name", name);
    formData.append("description", description);
    formData.append("image", file);
```

```
    await fetch("https://your-app.azurewebsites.net/add-product", {
      method: "POST",
      body: formData,
    });
  };

  return (
    <div>
      <h2>Product Catalogue</h2>
      <input type="text" placeholder="Product Name" onChange={(e) => setNam
      <input type="text" placeholder="Description" onChange={(e) => setDes
      <input type="file" onChange={(e) => setFile(e.target.files[0])} />
      <button onClick={handleUpload}>Add Product</button>
      <div>
        {products.map((product, index) => (
          <div key={index}>
            <h3>{product.name}</h3>
            <p>{product.description}</p>
            <img src={product.image_url} alt={product.name} width="200" />
          </div>
        ))}
      </div>
    </div>
  );
};

export default ProductCatalogue;
```

5 Application URL

The full-stack application is deployed under a single URL:

- **Application URL:** `https://cloudassign-aecxdeb9crgee7em.centralindia-01.azurewebsites.net/`

6 Cost Considerations

To minimize costs while using the Azure Student Offer:

- Unused resources are deleted periodically.
- The application is hosted on Azure App Service's free tier.
- Database queries and storage are optimized for efficiency.