# Project Report

## Developing a Cloud-Based Application Using Microsoft Azure Services

Course: Introduction to Cloud Computing CS 4037

Instructor: Muhammad Sudais

Semester: Fall 2024

**Group Members:**

Rohan Kumar (21K-3410)

Roain Kumar (21K-3184)

Avinash (21K-3413)

Sahil (21K-4696)

Ehan Ahmed (21K-3366)

# Table of Contents

1. Project Overview

2. Objectives

3. Application Design and Architecture

3.1 Architecture Diagram

3.2 Description of Azure Services Used

4. Functional Requirements

4.1 User Features

4.2 API Integration

4.3 Cloud Hosting and Database

4.4 Storage Integration

4.5 Monitoring and Security

5. Non-Functional Requirements

5.1 Performance and Scalability

5.2 Availability and Security

5.3 Cost Optimization

6. Implementation Details

6.1 Deployment Steps

6.2 Challenges and Solutions

7. Azure Services Configuration

7.1 App Service Setup

7.2 SQL Database Configuration

7.3 Azure Key Vault Security Setup

7.4 Monitoring Setup

8. CI/CD Pipeline

8.1 Pipeline Design and Implementation

9. Cost Management and Analysis

9.1 Resource Costs and Optimization Strategies

10. Conclusion and Key Learnings

11. References

12. Appendices

# 1. Project Overview

The Restaurant Ordering System is a cloud-based application designed to streamline customer and staff interactions with a restaurant's operations. The system supports order placement, tracking, and management in real-time while ensuring security, scalability, and performance.

Key Outcomes:

* Enhanced customer experience.
* Centralized order and menu management for restaurant staff.
* High scalability using serverless services and managed Azure solutions.

# 2. Objectives

**Improve Customer Experience**: Enable a responsive interface for browsing menus, customizing orders, and receiving updates.

**Streamline Operations**: Facilitate menu updates, order tracking, and error-free order processing.

**Ensure Scalability and Security:** Utilize Azure’s services for high availability and low latency.

# 3. Application Design and Architecture

## 3.1 Architecture Diagram

The architecture includes Azure Active Directory for authentication, Azure App Service for hosting the application, Azure SQL Database for structured data, and Azure Blob Storage for storing food item images.

A computer screen shot of a computer

Description automatically generated

## 3.2 Description of Azure Services Used

**Azure App Service:** Hosts the web application for both customers and administrators.

**Azure Functions:** Used for triggering order status updates and sending notifications to users.

**Azure Application Insights:** Ensures detailed real-time monitoring and performance analysis.

**Azure Blob Storage:** Provides efficient image storage and retrieval for food items.

**Azure SQL Database:** Manages structured order and customer data securely.

**CI/CD:** Facilitates continuous integration and deployment via GitHub. This comprehensive architecture streamlines the order lifecycle, enhances user interactions, and optimizes system performance.

# 4. Functional Requirements

**Order Status Updates**: The system must allow real-time status updates for customer orders using Azure Functions. Notifications must be triggered automatically when order status changes (e.g., "Order Placed," "In Progress," "Delivered").

**Email Notification Service:** Customers must receive email notifications for important updates related to their orders. The notification service must be integrated with Azure Functions for automated delivery.

**Customer and Restaurant Interaction:** Customers and restaurants must interact with the web app for placing and managing orders, powered by a PHP-based system.

**Data Storage and Retrieval:** Store all customer and order data in Azure SQL Database for structured management.Images of menu items must be stored and retrieved efficiently from Azure Blob Storage.

**Continuous Integration and Deployment (CI/CD):** Use GitHub for version control and CI/CD pipelines to deploy the application seamlessly.

# 5. Non-Functional Requirements

**Scalability:** The system must handle increasing user demand without degradation of performance, utilizing Azure's serverless architecture (e.g., Azure Functions).

**Performance Monitoring:** Use Azure Application Insights to monitor application performance, identify bottlenecks, and optimize system efficiency.

**Reliability:** The system should ensure high availability of services, including order processing, notifications, and database operations.

**Security:** Ensure all data stored in Azure Blob Storage and Azure SQL Database is secured with encryption. Implement user authentication for secure access to the web app.

**Cost Efficiency:** Use consumption-based pricing models for Azure services like Functions and Application Insights to optimize operational costs.

**Maintainability:** The system should support easy updates and modifications via CI/CD pipelines. All system logs and metrics must be accessible for troubleshooting and maintenance.

**User Experience:** Provide a user-friendly interface for both customers and restaurants, ensuring smooth navigation and interaction.

# 6. Implementation Details

**Backend Implementation**

**Azure Functions:** Azure Functions are implemented to handle asynchronous tasks such as order status updates and email notifications. The function is triggered by events (e.g., status change in orders) and sends notifications to customers via an integrated email service. The event-driven architecture ensures efficient and scalable processing of updates.

**Azure SQL Database:** Used for managing all structured data, including customer details, order information, and restaurant data. SQL queries are used for CRUD (Create, Read, Update, Delete) operations, ensuring data consistency and integrity. The database is optimized for handling high transaction volumes.

**Azure Blob Storage:** Blob storage is implemented to manage images of food items and other media files. Images are uploaded, stored, and retrieved using API calls, ensuring quick access and efficient storage.

**PHP Web Application:** The PHP-based backend powers the web app for customers and restaurants. The web app connects to the database to retrieve and update data and integrates with Azure Functions for order updates. A REST API is implemented for communication between the frontend and backend.

**Frontend Implementation**

**Customer and Restaurant Interfaces:** The user interface (UI) is built to support both customers and restaurants. Customers can browse menus, place orders, and track status.Restaurants can view incoming orders, update order status, and manage menu items.

**Notifications:** Real-time notifications (email) are integrated into the frontend to ensure customers receive status updates.

**Monitoring and Insights**

**Azure Application Insights:** Integrated into the web application to provide real-time monitoring and diagnostics.Tracks key performance indicators (KPIs), such as response times, error rates, and resource usage.

# 7. Azure Services Configuration

Configuration steps for App Service, SQL Database, Blob Storage , Azure Function and Azure insights.

# 8. CI/CD Pipeline

**GitHub Integration**:

* GitHub is used for version control and collaboration.
* CI/CD pipelines are implemented to automate deployment to Azure services, ensuring quick and reliable updates.

# 9. Cost Management and Analysis

The implementation of the restaurant ordering system required leveraging several Azure services, each with associated costs. Below is a breakdown of the major resource costs and strategies used to optimize expenses:

* **Azure App Service**: Hosted both the customer-facing web application and the admin portal. The service was configured to use the "Basic" pricing tier to balance cost and performance during development.
* **Azure SQL Database**: Used for storing user data, menu items, orders, and transactional records. The database was set up with a DTU-based pricing model, selecting the Basic tier for cost efficiency.
* **Azure Blob Storage**: Utilized for storing and serving food images. Costs were minimized by using access tiers such as "Hot" for frequently accessed images and "Cool" for less frequently accessed data.
* **Azure Application Insights**: Used for monitoring the application and collecting performance metrics.
* **Azure Functions :** operates on a consumption-based pricing model, making it cost-effective for handling sporadic workloads like order status updates. Application Insights adds minimal overhead while providing critical monitoring capabilities.

**Cost Optimization Strategies:**

1. **Reserved Instances**: Utilizing reserved instances for predictable workloads helped reduce costs.
2. **Autoscaling**: Configuring autoscaling for the App Service ensured resources were only allocated during peak usage periods.
3. **Access Tiers**: Leveraged Azure Blob Storage access tiers (Hot, Cool) to optimize costs based on data usage patterns.
4. **Resource Cleanup**: Regularly identified and removed unused resources to prevent unnecessary charges.

# 10. Conclusion and Key Learnings

The restaurant ordering system was successfully developed using Microsoft Azure's cloud services. The project achieved its primary goals of creating a scalable, reliable, and user-friendly platform for restaurant customers and administrators Key outcomes and learnings include:

**Project Outcomes:**

1. A fully functional web application hosted on Azure App Service, enabling customers to browse menus, place orders, and track order statuses in real time.
2. The inclusion of Azure Functions demonstrates the value of serverless computing in creating scalable, cost-efficient solutions.
3. Additionally, Application Insights offered invaluable insights into system performance.
4. An administrative portal for managing orders, menus, and restaurant operations effectively.
5. Integration of Azure SQL Database for secure and reliable data storage.
6. Efficient media delivery through Azure Blob Storage integration.

**Key Learnings:**

1. **Azure Ecosystem**: Gained hands-on experience with various Azure services, including App Service, SQL/Cosmo Database, and Blob Storage.
2. **Scalability and Optimization**: Learned to implement server configurations and optimize costs using Azure's autoscaling features and pricing tiers.
3. **Security Practices**: Understood the importance of using Azure Active Directory for authentication and Key Vault for securing sensitive data.
4. **Cloud-Based Architecture**: Developed a deeper understanding of designing and deploying cloud-based systems to meet functional and non-functional requirements.