**Proposed Solution Overview**

The system will use **Azure Functions** as the core backend for its event-driven (http, Service Bus message received), serverless backend architecture. The solution will also involve **event-driven workflows**, **scalable Azure services**, and **.NET technologies** for flexibility and rapid development.

The architecture adheres to:

1. **Cloud-native principles** for scalability and cost optimization.
2. **Incremental migration processes** to integrate the legacy monolithic system during traffic switching.
3. Support for concurrent users in the **tens of thousands with holiday surges**.
4. Effortless integration for third-party resellers, flight providers, ticket resale, social sharing, multi-currency, payments system.

**Key Architectural Components**

The architecture consists of the following:

1. **Frontend Application**:
   * Adaptive **web-based UI** hosted on **Azure Static Web Apps** to improve performance and scalability.
   * Interacts with **backend Azure Functions** via RESTful APIs exposed through **Azure API Management** or **backend Azure App Service** to provide administration functionality.
   * Enables seamless integration with guest and registered users (including loyalty rewards, price tiers, and chatbots).
2. **Azure App Service (Backend Application)**:
   * **Administration App:** Enables application administrators to manage and configure application settings dynamically, leveraging insights from traffic statistics to optimize performance and user experience.
3. **Azure Functions (Backend Application)**:
   * The backend is split into independent serverless Azure Functions, enabling modular and event-based functionality for handling various tasks:
     + **User Management Service**: Handles user registration, authentication (using **Microsoft Entra ID**), user related tickets lifecycle data, generates shareable links for tickets resale for social media.
     + **Loyalty Service**: Manages loyalty points and tier upgrades (bronze/silver/gold/platinum)
     + **Flight Pricing Engine**: Aggregates real-time data from multiple flight data providers together with the legacy platform, configures profit margins (including ones, based on loyalty and traffic statistics), and calculates the best price for users (based on loyalty data).
     + **Payment Service**: Handles multi-currency and cryptocurrency payments, refunds, and third-party payment provider integrations (e.g., PayPal, VISA, MasterCard, Bitcoin).
     + **Notification Service**: Sends notifications via email, SMS, and messaging apps for flight updates, loyalty deals, ticket resale events, etc.
     + **Advertisement Service**: Processes the Advertisement from external partners APIs.
4. **Storage**:
   * **Azure Cosmos DB**:
     + NoSQL storage for dynamic application settings data, users data, tickets lifetime history, loyalty, analytics.
   * **Azure Blob Storage**:
     + Store static ticket files (PDFs, QR codes) and media for resale tickets.
5. **Data and Workflow Automation**:
   * Use **Azure Service Bus** for real-time event processing (e.g., ticket resale changes, user data changes, tickets payment lifecycle changes).
6. **Security**:
   * **Azure Front Door** and WEfor global load balancing and traffic management with integrated failover and DDoS protection.
   * **Azure Monitor, Log Analytics, and Application Insights** for observability and performance monitoring.
   * Secure APIs through **Azure API Management**
   * Authentication via **Microsoft Entra ID**.
   * Integrate **Azure Key Vault** to store sensitive data like API keys, database credentials, payment secrets, and service account keys securely
7. **Analytics**:
   * **Azure Log Analytics** for collecting and processing site traffic and user behavior data.
   * **Power BI dashboards** to analyze user behaviour, popular flights and opportunities, popular routes, to audit and legal purposes.
8. **Incremental Migration**:
   * Setup **Azure API Management** to act as a central gateway for routing user requests between legacy and new systems
   * The purpose of **Data Factory** is to move existing premise user data, flight bookings, and loyalty to the target databases.

**CI/CD Workflow with Azure DevOps**

**Overview**

The CI/CD pipeline automates:

1. Building, testing, and packaging the Azure Functions and frontend code.
2. Deploying to **staging** for validation.
3. Promoting to **production with blue-green deployments**.

**CI Pipeline (Continuous Integration)**

**Trigger**: Code changes (commits, pull requests) on feature/\* branches.

**Steps:**

1. **Build Azure Functions, Azure App Service and Frontend**:
   * Compile Functions/Azure App Service using **.NET Core**.
   * Use **NPM/Node.js** to build the frontend.
2. **Unit Tests**:
   * Run tests for function business logic and utilities.
   * Results are stored in the pipeline workspace.
3. **Publish Artifacts**:
   * Publish deployable artifacts to the **Azure DevOps pipeline storage**.

**CD Pipeline (Continuous Deployment)**

**Trigger**: Successful completion of the CI pipeline.

**Steps:**

1. **Deploy to Staging**:
   * Deploy Azure Functions, Azure App Service and frontend assets to staging slots.
   * Automatically run **integration/e2e tests**.
2. **Approval Gate (Manual / Auto)**:
   * Require manual approval or validation tests to promote releases to production.

**Deploy to Production**:

* **Infrastructure as Code:**
  + Use Azure Resource Manager (ARM) templates, Terraform, or Bicep to define and deploy the entire infrastructure programmatically (Azure Function Apps, API Gateway, SQL, Cosmos DB).
* Use **blue-green deployment** strategy:
  + Deploy Functions/ Azure App Service to a new slot or environment and then swap.
  + Use two instances of Static Web Apps behind an Azure Front Door

**Ways for improvements**

1. **Modular microservices hosted on a Cluster**

Transition from the **serverless Azure Functions architecture** to a more sophisticated and scalable **cluster-based approach** utilizing **Azure Kubernetes Service (AKS)** or **Azure Service Fabric**.

1. **Encapsulate the Flow by Wrapping Azure Functions within a Logic App**
2. **Additional Decomposition of Microservices**

* For example, divide the **User Management Service** into specialized services, each handling specific responsibilities:
  + **User Registration and Authentication**: Focuses on user onboarding, credentials management, and authentication workflows.
  + **User-Related Ticket Lifecycle Management**: Responsible for tracking tickets associated with users, including purchase history, cancellations, and modifications.
  + **Social Media Ticket Resale Link Generation**: Generates shareable links for ticket resale across social media platforms, ensuring seamless user interaction and visibility.

1. **Leverage Azure Premium Plans for Function App**

Upgrade to an **Azure Functions Premium Plan** to:

* Avoid **cold starts** (reduce latency for infrequent executions).
* Enable **pre-warmed instances** for critical services like the Flight Pricing Engine or Payment Processing Functions.
* Scale out **across multiple regions** to handle traffic surges (e.g., during holidays)

1. **Use Distributed Caching**

Introduce **Azure Cache for Redis** to:

* Store frequently accessed data like flight pricing results, resale listings etc.
* Reduce repeated calls to **Azure Cosmos DB**, or third-party APIs, improving response times for users.

1. **Optimize API Gateway Rules**

* Add advanced **rate-limiting** rules or **geo-fencing** in **Azure API Management** to prevent sudden traffic overloads (e.g., malicious DDoS bursts).
* Use **API transformations** for seamless integration of external third-party providers without modifying downstream services.

1. **Multi-Region Deployment**

* Make the system **multi-region** (e.g., deploying Azure Functions to **multiple Azure regions**) to improve availability for global users.
* Use **Azure Front Door** for routing traffic intelligently between regions and ensuring geo-distributed load balancing.

1. **Enhance Security**

* Improve **Azure Key Vault**
  + Rotate credentials automatically to prevent unauthorized accesses.
* **Advanced Threat Protection**
  + Use **Azure Security Center** and enable **Advanced Threat Protection (ATP)** for **Azure SQL**, **Cosmos DB**, and **Blob Storage**.
  + ATP provides alerts on unusual behaviors or access that could indicate potential attacks (e.g., brute-force logins).
* Upgrade **DDoS Protection** to include **application-layer protection** alongside **Azure Front Door**:
  + Detect and mitigate higher-layer attacks targeting APIs (e.g., Flight Pricing or Ticket Resale services).

1. **Chatbot Enhancement**

* Embed the shopping assistant chatbot with **AI-powered conversational capabilities**:
  + Add AI APIs via **Azure Cognitive Services (LUIS)** to recommend flights, loyalty deals, and support FAQs.
  + Personalize responses based on **user purchase history or frequent search data**.

1. **Expand Higher Availability and Resilience**

* **Global Multi-Region** Deployment
  + Deploy services like **Flight Pricing Engine** or **Notification Functions** into **multiple Azure regions**.
  + Use **Global Traffic Manager** to handle failover between regions automatically.
* **Function-Level Resilience**
  + Add **retry logic** or **backup triggers** for Azure Functions handling critical workflows:
    - E.g., retry failed ticket resale updates or payment processing within 10 seconds.
* **Disaster Recovery (DR)**
  + Enable disaster recovery and business continuity planning with **Geo-Redundant Storage (GRS)** for databases and blobs.
  + Back up critical transactional data (loyalty points, flights) using **Azure Backup Services**.