Software Architecture – Egypt Tourism Hub

1. Architecture Style

The system will be designed using a **Microservices Architecture** to ensure scalability, flexibility, and maintainability. Each core functionality (e.g., user management, booking system, marketplace) will be developed as an independent microservice, communicating through RESTful APIs or GraphQL.

Alternatively, an **MVC (Model-View-Controller)** pattern will be used within individual microservices to separate concerns and enhance maintainability.

2. High-Level System Components

A. Frontend (User Interface Layer)

- **Technologies:** HTML5, CSS3, JavaScript (ES6+), and React.js for dynamic web interfaces.
- **Responsibilities:**
  - Provides an intuitive and interactive user experience.
  - Communicates with backend services via APIs.
  - Manages authentication state and user sessions.

B. Backend (Business Logic Layer)

- **Technologies:** .NET Core (C#) with Web API for scalable and maintainable services.
- **Microservices:**
  - **Authentication Service** – Manages user authentication & authorization (OAuth 2.0, JWT, social login support).
  - **User Management Service** – Handles user profiles, roles, and preferences.
  - **Booking Service** – Manages trip reservations, activity bookings, and tour packages.
  - **Marketplace Service** – Handles vendor listings, souvenir sales, and order management.
  - **Payment Gateway Integration** – Secure online payments using Stripe, PayPal, or local alternatives.
  - **Review & Ratings Service** – Manages customer feedback and ratings for activities and products.
  - **Notifications Service** – Sends email/SMS/app notifications for confirmations and updates.

C. Database Layer (Data Storage & Management)

- **Primary Databases:** Microsoft SQL Server for structured data.
- **NoSQL Storage:** MongoDB / Firebase for unstructured or semi-structured data (e.g., user activity logs, reviews, and analytics).
- **Caching:** Redis for session management and quick data retrieval.
- **Search Engine:** Elasticsearch for enhanced filtering and search capabilities.
-

D. API Gateway & Communication

- **API Gateway** (e.g., Ocelot for .NET Core) to handle requests, authentication, rate-limiting, and logging.
- **Internal Communication:** RESTful APIs between microservices.
- **External Integration:** Third-party APIs for flights, maps, weather, and payment gateways.

E. Infrastructure & Deployment

- **Cloud Platform:** Azure / AWS for hosting and scalability.
- **Containerization & Orchestration:** Docker + Kubernetes for deployment and scaling.
- **CI/CD Pipeline:** GitHub Actions / Azure DevOps for automated deployment and updates.
- **Logging & Monitoring:** Serilog + Application Insights for system health monitoring.
- **Security Measures:** HTTPS, SSL/TLS encryption, IAM roles, data encryption at rest and in transit.

3. System Interactions & Workflow

A. User Journey – Booking a Trip

1. **User Registration & Authentication**
    - User signs up using email, phone, or social login.
    - Authentication service validates credentials and issues JWT.
2. **Browsing & Filtering Activities**
    - User searches for trips, accommodations, or souvenirs.
    - Frontend fetches data from respective microservices via API Gateway.
    - Results are displayed with filtering and sorting options.
3. **Booking & Payment Process**
    - User selects an activity and initiates booking.
    - Booking service verifies availability and confirms the reservation.
    - Payment service processes the transaction securely.
    - Notifications service sends confirmation email/SMS.
4. **Order Management in Marketplace**
    - Vendors list souvenirs and manage orders.
    - Users add items to cart and complete purchase.
    - Marketplace service handles transactions and communicates with the payment gateway.
5. **Reviews & Ratings**
    - Users leave feedback after completing a booking or purchase.
    - Reviews are stored and displayed for future users.

4. Scalability, Performance & Resiliency Enhancements

- **Load Balancing:** Distribute traffic efficiently using Azure Load Balancer / AWS ALB.
- **Auto-Scaling:** Implement horizontal scaling to handle peak loads.
- **Rate Limiting:** Protect APIs from abuse using API Gateway policies.

- **Resiliency Strategies:** Implement failover mechanisms, redundancy, and database replication.
- **Logging & Monitoring:** Centralized logging with ELK Stack, Azure Monitor.
- **Real-Time Communication:** Utilize WebSockets or SignalR for live updates.

5. Mapping Use Cases to System Components

Use Case	Responsible Service (Microservice)	Description
User Registration & Authentication	Authentication Service	User signs up, logs in, and manages credentials.
User Profile Management	User Management Service	Users update personal information and preferences.
Browse & Filter Destinations	Trip & Activity Service	Users explore cultural activities and guided tours.
Booking Trips & Activities	Booking Service	Users book trips, receive confirmations, and manage bookings.
Shopping & Souvenir Purchases	Marketplace Service	Users browse and purchase souvenirs from vendors.
Payment Processing	Payment Service	Handles secure transactions for bookings and purchases.
Reviews & Ratings	Review & Ratings Service	Users leave feedback and rate activities and vendors.
Notifications & Alerts	Notification Service	Sends emails/SMS for bookings, payments, and promotions.
Vendor & Shop Management	Vendor Management Service	Vendors list, update, and manage souvenir shops and offerings.
Reporting & Analytics	Reporting & Analytics Service	Generates insights on bookings, purchases, and customer behavior.