a. Architectural Characteristics

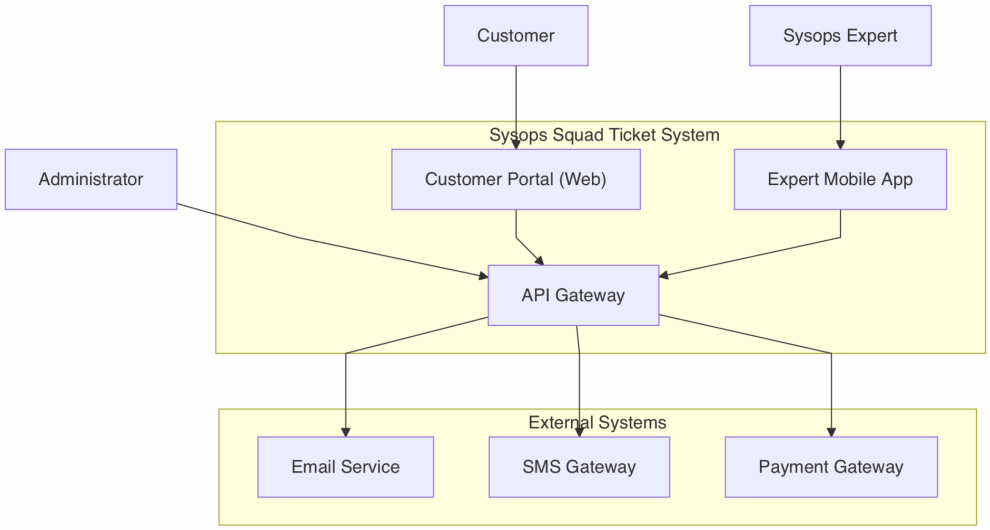
	Characteristic
Availability	The system must be available 24/7 for customers and experts to submit and track tickets.
Scalability	The system must handle spikes in ticket volume and user access during high-demand periods.
Reliability / Fault Tolerance	The system should continue operating even when one microservice or component fails.
Performance	Responses (ticket creation, assignment, updates) must happen within seconds.
Modifiability / Maintainability	The system should support frequent changes without affecting other components.
Security	Authentication and authorization for customers, experts, and admins; secure payments and personal data.
Auditability	All ticket updates, assignments, and billing transactions should be logged.
Usability	Web and mobile apps must be user-friendly for both customers and Sysops experts.
Interoperability	Integration with SMS, email notification, and payment systems.

b. Scenarios per Characteristic

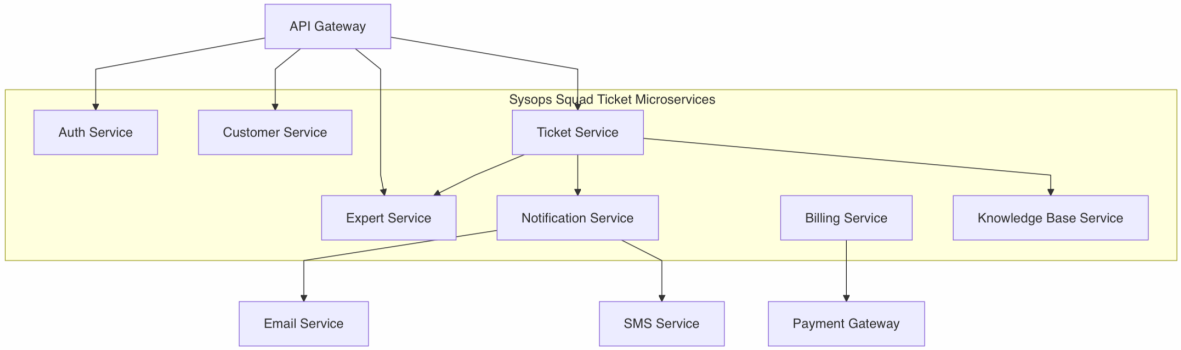
Characteristic	Scenario
Availability	If one microservice fails (ex. Notification Service), others like Ticket or Expert Service should continue running; retry mechanism for notifications.
Scalability	During a product recall, when thousands of customers submit tickets, the system auto-scales via container orchestration (Kubernetes).
Reliability	When a database replica fails, the service continues using a secondary replica.
Performance	Ticket assignment occurs within 2 seconds after a customer submits a request.
Modifiability	A new “priority level” field can be added to tickets without redeploying all services.
Security	Only registered customers can create tickets; only authorized experts can access them.
Auditability	Admin can query the system to see which expert handled which ticket on which date.
Usability	Experts access tickets easily via a responsive mobile app interface.
Interoperability	The system integrates seamlessly with Twilio (SMS) and Mailgun (email).

c. Architecture Definition (Diagrams)

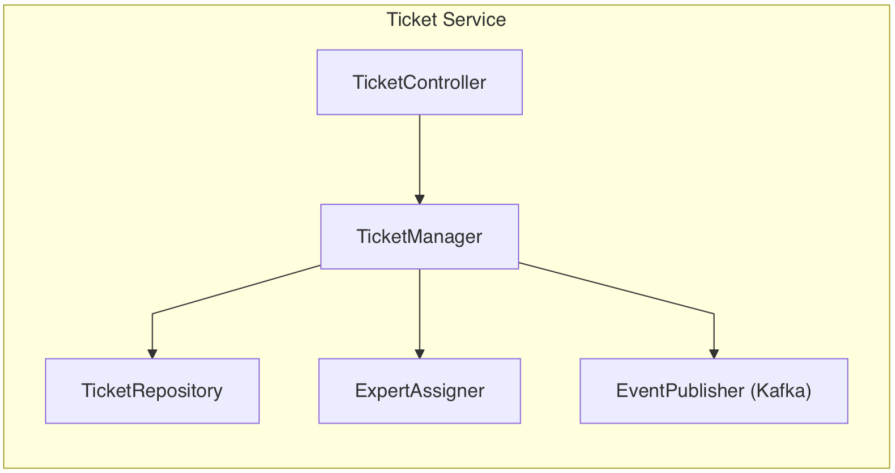
1. Context Diagram



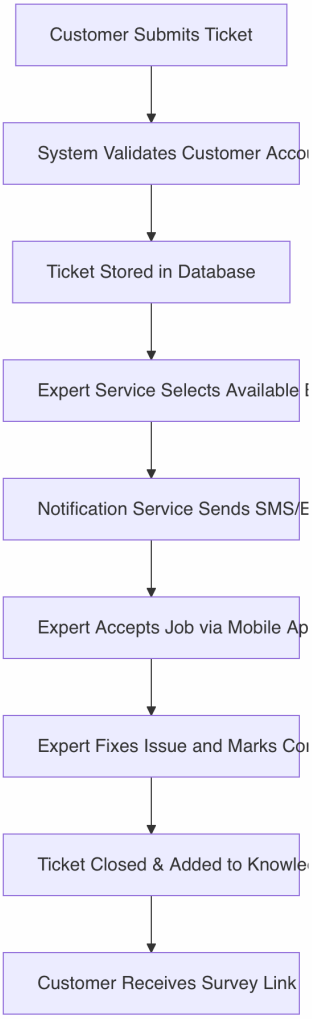
2. Container Diagram



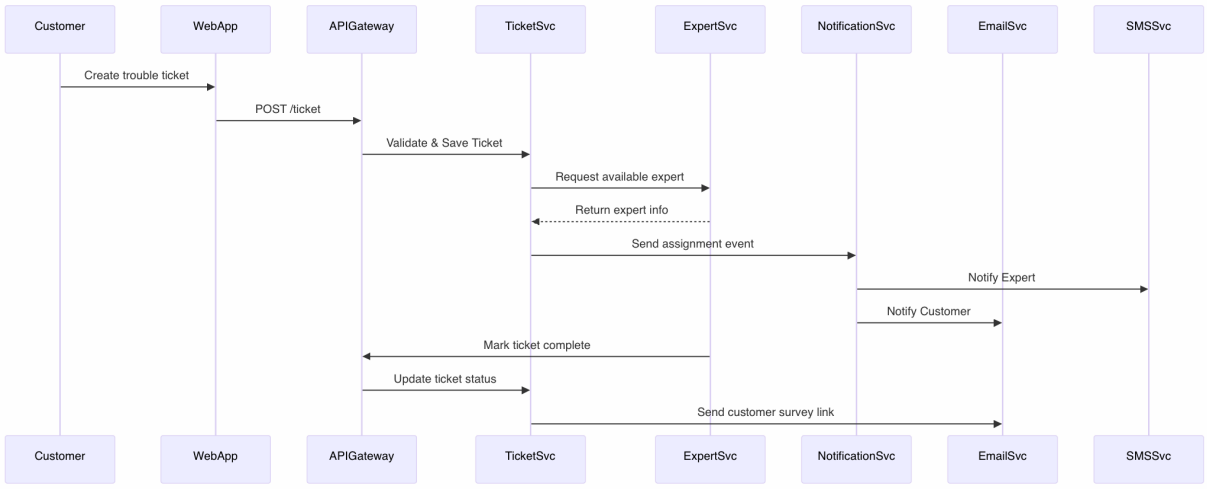
3. Component Diagram (for Ticket Service)



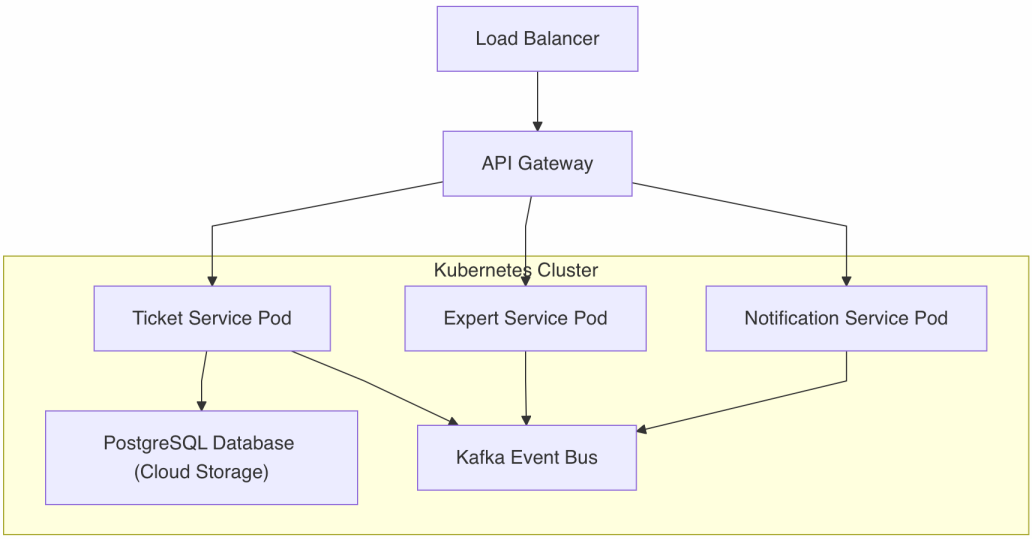
4. Activity Diagram (Ticket Lifecycle)



5. Sequence Diagram (End-to-End Ticket Flow)



6. Deployment Diagram (Optional – for completeness)



---

d. Risks of the Proposed Architecture

Risk	Description	Impact
Integration Failure	Email/SMS/Payment services may be unavailable or slow.	Missed notifications or delayed billing.
Data Inconsistency	Ticket or expert data may become inconsistent if services fail mid-transaction.	Wrong assignments or lost tickets.
Performance Bottlenecks	Improperly scaled Ticket Service could delay responses.	Customer dissatisfaction.
Security Breach	Unauthorized access or data leakage.	Legal and financial penalties.
Complex Deployment	Microservices increase deployment complexity.	DevOps overhead and potential downtime.
Message Loss in Queue	Kafka event loss or duplication.	Missed expert notifications or duplicate assignments.

e. Risk Mitigation Options

Risk	Mitigation Strategy
Integration Failure	Implement retry policies, circuit breakers, and service fallback responses.
Data Inconsistency	Use distributed transactions (Saga pattern) and event-driven consistency.
Performance Bottlenecks	Apply horizontal auto-scaling and caching layers (Redis).
Security Breach	Use OAuth2, JWT authentication, TLS encryption, and RBAC.
Complex Deployment	Automate CI/CD pipeline with Docker + Kubernetes + Helm charts.
Message Loss	Enable Kafka message persistence, acknowledgments, and dead-letter queues.