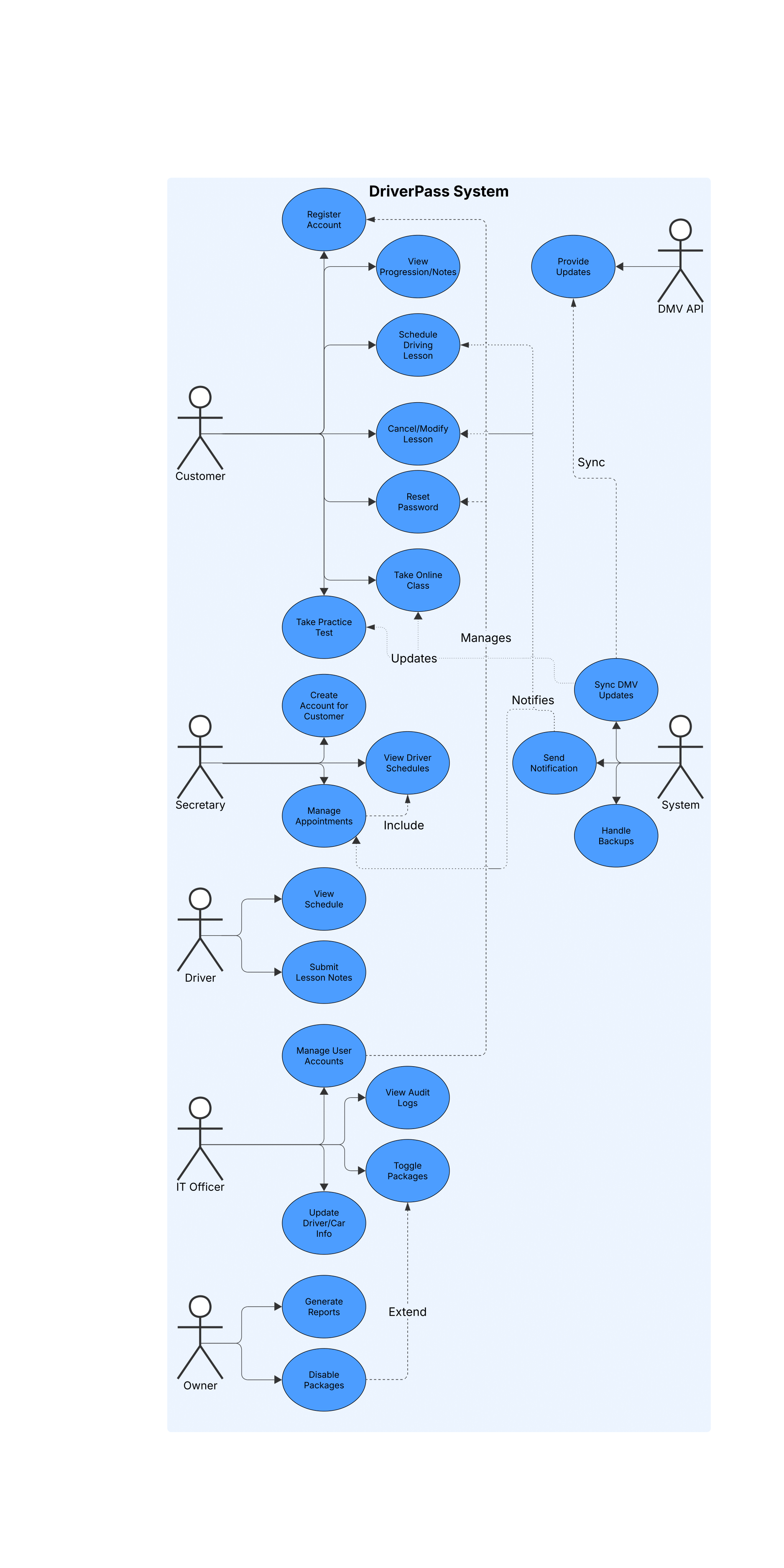
# CS 255 System Design Document Template

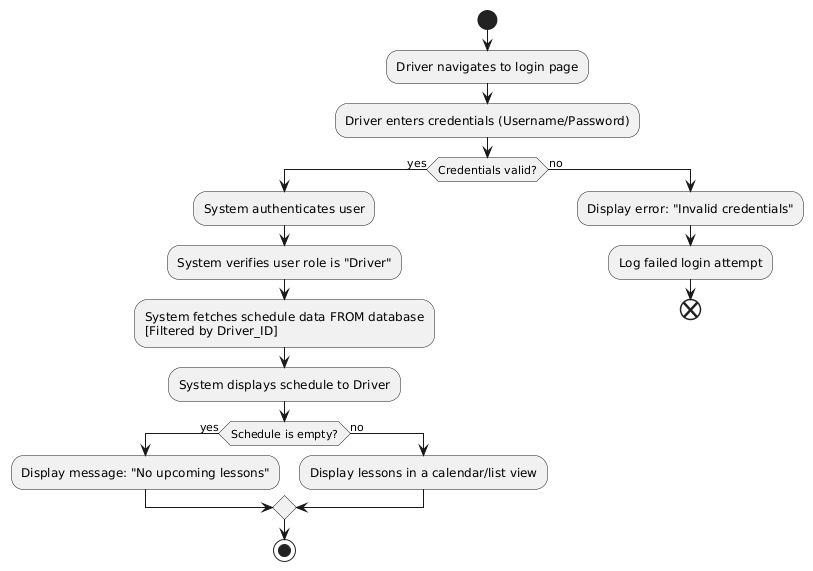
## UML Diagrams

### UML Use Case Diagram

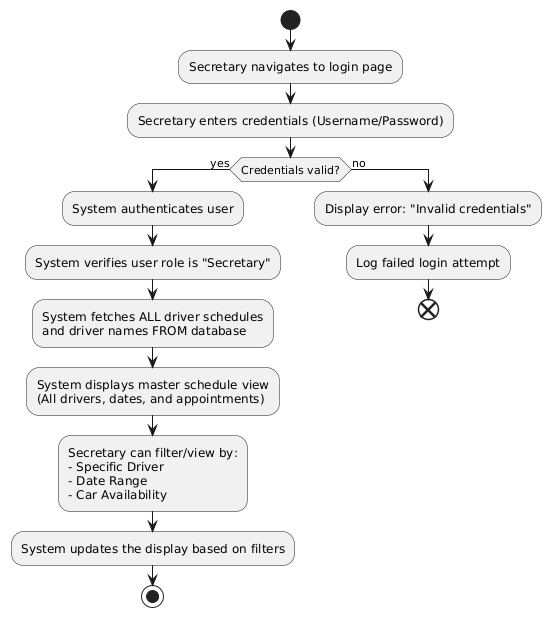
**

### UML Activity Diagrams

Driver Activity Diagram

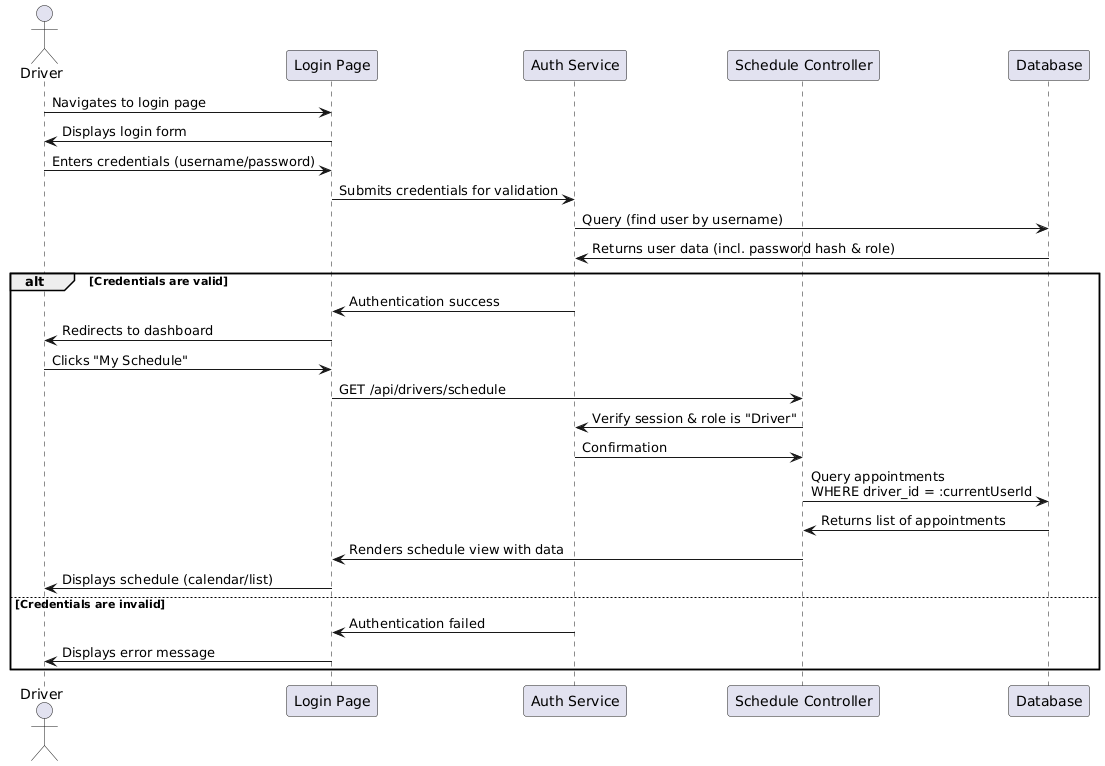
**

Secretary Activity Diagram

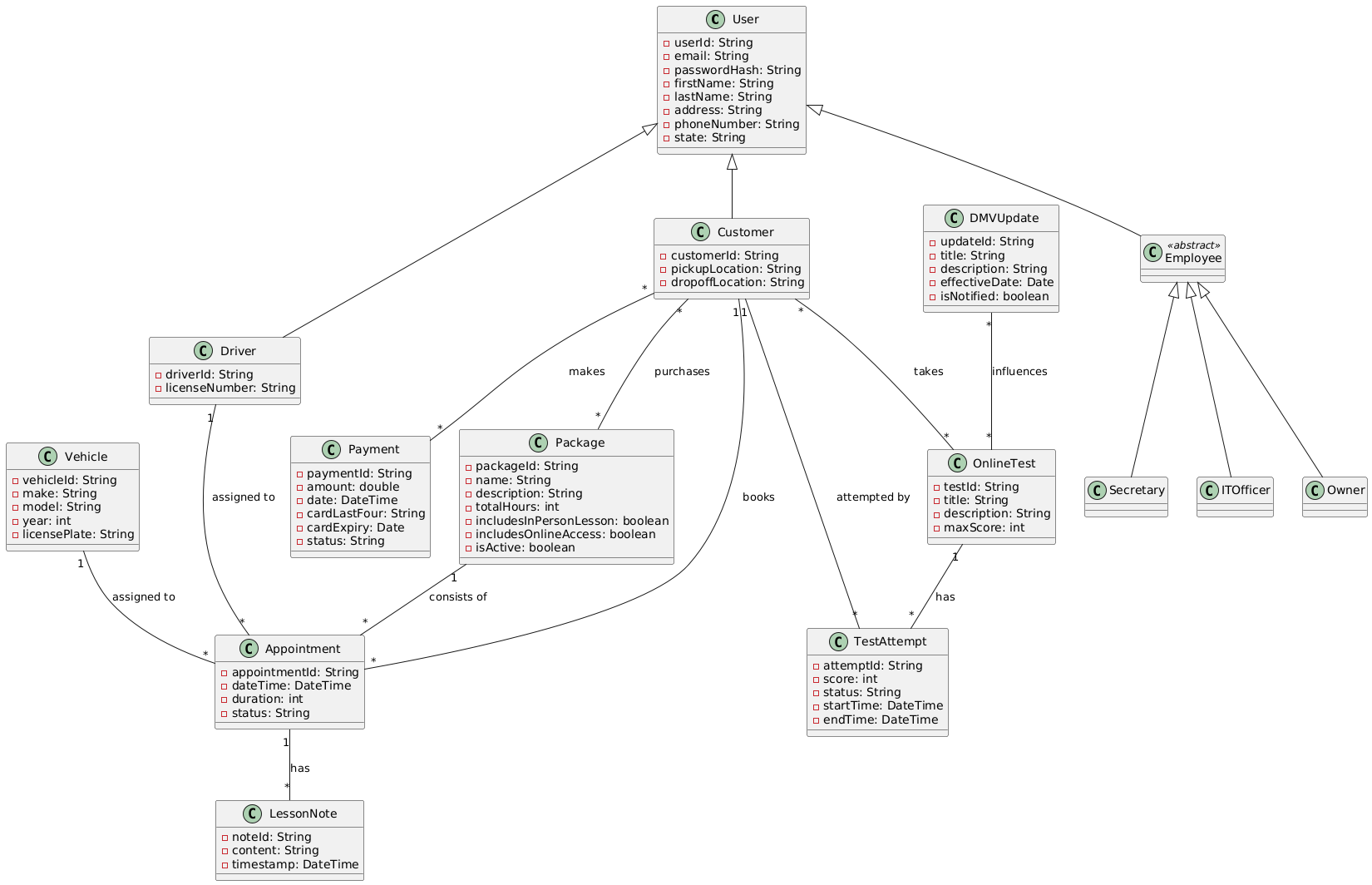


### UML Sequence Diagram

Driver Sequence Diagram

**

### UML Class Diagram

**

## Technical Requirements

**Infrastructure & Hosting**

* **Cloud Platform**:
  + Utilize a scalable cloud provider for hosting, leveraging managed services for auto-scaling, backups, and security.
  + Ensure global availability with multi-region deployment for redundancy.
* **Content Delivery Network (CDN)**:  
  Implement a CDN for static assets (e.g., practice tests, course materials) to optimize offline access and global performance.

**Hardware**

* **Server Resources**:
  + Auto-scaling compute instances to handle greater than 50 concurrent users with real-time responsiveness.
  + Load balancers to distribute traffic evenly.
* **Database Servers**:
  + Managed SQL/NoSQL database service with read replicas for audit logs, user data, and appointment scheduling.
  + Storage-optimized instances for downloadable reports.

**Software & Tools**

* **Back End**:
  + **Frameworks**: Node.js/Python/Java for real-time API handling (WebSockets for notifications).
  + **Database**: Relational DB for structured data (users, appointments); NoSQL for audit logs.
  + **APIs**: RESTful services for DMV integration, payment processing (e.g., Stripe), and email/SMS alerts (e.g., Twilio).
* **Front End**:
  + Responsive web app using React/Angular with PWA support for offline data viewing.
  + Cross-browser compatibility (Chrome, Firefox, Safari, Edge).
* **Security Tools**:
  + Encryption: TLS 1.3 for data in transit; AES-256 for data at rest.
  + WAF (Web Application Firewall) and rate-limiting (e.g., Cloudflare) to block brute-force attacks.
* **DevOps**:
  + Containerization (Docker), packaging and deploying applications within isolated, portable environments called containers, and orchestration (Kubernetes), involves the coordination and automation of various tasks required to deploy, scale, manage, and maintain applications composed of multiple containers, for zero-downtime deployments.
  + Monitoring: logs and performance metrics.

**Integration & APIs**

* **DMV API Integration**:
  + Real-time synchronization via webhooks/event-driven architecture.
  + Caching mechanism to handle API throttling.
* **Third-Party Services**:
  + Payment gateways with PCI-DSS compliance.
  + Multi-factor authentication.

**Data Management**

* **Database Schema**:
  + Tables for users (ID, email, role, package tier), appointments, drivers, vehicles, audit logs.
  + RBAC-driven access controls at the database level.
* **Audit Logs**:
  + Immutable logging of user actions with downloadable CSV/PDF reports.
* **Backups**:
  + Automated daily backups (cloud-managed) with a set amount of day retention; emergency manual snapshots.

**Security**

* **Authentication**:
  + OAuth 2.0/OpenID Connect for login; Bcrypt, uses a salt and a work factor, for password hashing.
  + Session invalidation on password reset.
* **Protections**:
  + Account lockout after 5 failed attempts; IP blocking for suspicious activity.
  + Automated alerts to admins via email/SMS for security events.
* **Compliance**:
  + GDPR/CCPA for user data; SOC 2 for cloud infrastructure.

**User Interface**

* **Front-End Framework**:
  + Component-based UI with role-specific dashboards:
    - **Customers**: Calendar for lessons, progress tracking, self-service portal.
    - **Secretaries/Drivers**: Drag-and-drop scheduling (e.g., FullCalendar.js).
    - **IT/Owners**: Admin panels for user management, package toggling, and metric visualizations.
* **Offline Capability**:
  + Service workers to cache downloadable data (e.g., reports).

**Scalability & Performance**

* **Real-Time Handling**:
  + Message queue for appointment conflicts and notifications.
  + Serverless functions (e.g., AWS Lambda) for burst traffic.

**Maintenance & Operations**

* **Updates**:
  + Zero-downtime deployments via blue-green switching.
  + Emergency patching access for IT roles.
* **Disaster Recovery**:
  + Multi-AZ cloud deployment; automated failover.