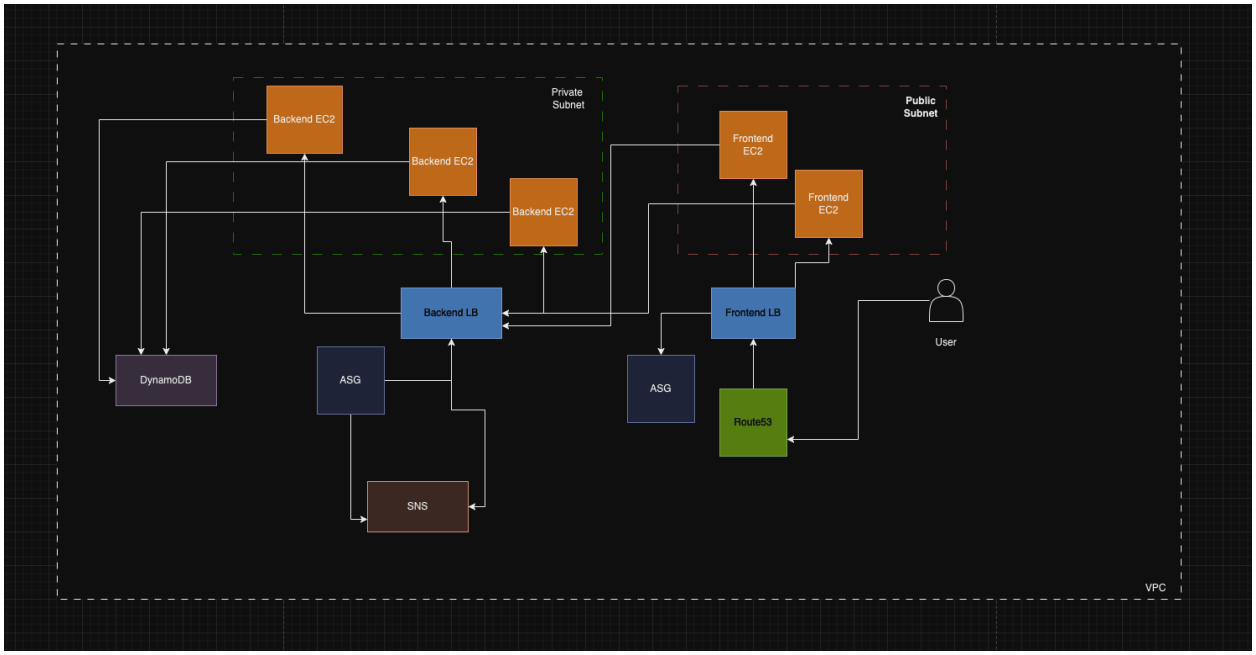
Design Document- Updated

1. A brief explanation of the function your web app will serve.

The e-commerce website is a platform where users can browse, purchase, and review products. Key features include:

1. Product Listings – Browse products by category, view product details.
2. User Accounts – Users can register, log in, and manage profiles.
3. Shopping Cart – Add, remove, and update items in a virtual cart.
4. Checkout – Secure transaction handling.
5. Order Management – Track order history, shipping details, and status.
6. Resilience Features – Automatic scaling, failure isolation, and monitoring.

2. Architecture Diagram(s) of your proposed solution to the requirements.



https://drive.google.com/file/d/1ld3gY2_fT1AGt8JqYmjY5aYxpTvH8dWv/view?usp=sharing

3. Brief bullet points on how your proposed architecture meets the resilience requirements and performance requirements.

1. I am performing a multi a-z deployment backed by a load balancer(ALB).Multi a-z deployment will provide me with redundancy and fault tolerance

2. Load balancer will be backed by auto scaling group
3. Auto scaling groups will be configured to check for CPU utilization threshold to enable automatic scaling in the event of high load.
4. In addition to this auto scaling groups will also be configured to check for ELB health whereby any instance with health check failures will be promptly replaced.
5. SNS Notifications will be sent for all activities performed by ELB and auto scaling group in addition to cloud watch monitoring for ec2 instances and ELB.

4. Your calculations for your Performance Requirements

User Data

- User Base Size:
    - Current Size: 100 daily active users.
    - Projected Monthly Growth: Assuming a 10% monthly growth rate, estimate future user counts:
        - 3 Months: ~133 users.
        - 6 Months: ~179 users.
        - 12 Months: ~313 users.
- Average Request Size:
    - Estimate each user request to be around 1 KB.
- Peak Hours:
    - Anticipate 50% of daily traffic during peak hours (e.g., 6-9 PM).
- Average Number of Requests per Day per User:
    - Estimate 20 requests per user, per day (e.g., product browsing, adding to cart, checkout).
- Seasonal Variance:
    - Account for traffic spikes during holidays or promotions, expecting traffic to double or triple.

Storage Data

- Read and Write Capacity:
    - Given the projected user count and activity:
        - Estimated Reads: 2,000 reads/day (10 reads/user, accounting for browsing and product views).
        - Estimated Writes: 400 writes/day (adding items to cart, checkout, and order updates).
- Average Read/Write Request Size:
    - Average request size estimated at 1 KB.
- Estimated Storage Size:
    - With a large product catalog, DynamoDB's storage is scalable without limits. Assuming 400 products and SKUs with each item averaging 2 KB, initial storage will be approximately 2 MB, but DynamoDB will adjust automatically as product data grows.

- Retention Policy:
    - Retain user session and order data for up to one year, while product and inventory data remain indefinitely.
- Performance Requirements for Storage:
    - Provisioned mode (specifying read and write capacity units) or on-demand mode (automatically scaling capacity to traffic) depending on cost and performance needs.
- Storage Use Cases:
    - Transactional: DynamoDB handles high-throughput product searches, cart operations, and checkout processes.
    - Archival: For archival data, consider using DynamoDB with lifecycle policies to transfer older data to Amazon S3 for cost-effective long-term storage.

Database Requirements(DynamoDB)

- Transactions Per Second (TPS): Aim for 50-100 TPS to handle peak load (during holidays or events), and configure DynamoDB auto-scaling accordingly.
- Read/Write Ratio: Estimated at 90% reads to 10% writes.
- Query Complexity: Use DynamoDB's indexes and query capabilities to optimize frequent operations, such as product searches and user order lookups.
- Indexing & Caching: Global Secondary Index (GSI): Create indexes for search fields (e.g., product category, price range).

Bandwidth Requirements

Data Ingress: Estimate 100-200 KB per second during peak hours for data coming into DynamoDB (e.g., checkout requests, adding items to cart).

- Data Egress: Anticipate up to 500 KB per second in peak periods for data retrievals (e.g., product browsing, search results).
- Latency Expectations: Sub-50 ms for non-cached data in high-traffic periods and under 40 ms for data in regular traffic.
- Network Traffic Patterns: Primary paths include user to frontend ELB. Frontend to backend ELB and backend to DynamoDB

Secure Architecture Considerations:

CNAS-1: Insecure Cloud, Container, or Orchestration Configuration

Relevance: Architecture relies on S3, DynamoDB, and Auto Scaling Groups.

Proposed Update:

- Enable S3 bucket versioning and object lock to prevent accidental deletion or overwrites.
- Use strict IAM roles to limit access to specific services like DynamoDB and S3.

- Configure Auto Scaling Groups with robust health checks and ELB-specific policies to avoid misconfigurations.

CNAS-2: Injection Flaws (App Layer, Cloud Events, Cloud Services)

Relevance: Inputs for cart operations, checkout, and product searches can be exploited.

Proposed Update:

- Sanitize and validate all user inputs at the frontend before sending them to the backend. Implement server-side validation and use parameterized queries for DynamoDB operations.

CNAS-3: Insecure or Insufficient Identity and Access Management

Relevance: IAM roles govern access to AWS services.

Proposed Update:

- Use fine-grained permissions for IAM roles, such as read-only access for frontend roles and write access only for backend services performing transactional operations.

CNAS-4: Exposed Secrets

Relevance: If environment variables are exposed, attackers could exploit them.

Proposed Update:

- Store secrets like API keys or database credentials in AWS Secrets Manager and retrieve them securely.

CNAS-5: Misconfigured or Insecure Data Storage

Relevance: Sensitive user data like order details and session information is stored.

Proposed Update:

- Encrypt data at rest in DynamoDB and in transit using HTTPS. Use DynamoDB's native encryption features for sensitive data.

CNAS-6: Broken Authentication

Relevance: User login functionality is part of the application.

Proposed Update:

- Enforce strong password policies, enable multi-factor authentication (MFA), and use secure session tokens with expiration.

CNAS-7: Lack of Secure Software Development Lifecycle

Relevance: Regular updates and secure coding practices are needed.

Proposed Update:

- Integrate automated security testing into the CI/CD pipeline to catch vulnerabilities early.

CNAS-8: Insufficient Monitoring

Relevance: CloudWatch is used for monitoring EC2 and ELB activities.

Proposed Update:

- Configure CloudWatch to monitor security events and send alerts via SNS for suspicious activities, such as unauthorized API calls.

CNAS-9: Weak Cryptographic Practices

Relevance: HTTPS is critical for secure communication.

Proposed Update:

- Use AWS Certificate Manager (ACM) to implement SSL/TLS for all communication between clients and services.

CNAS-10: Insecure APIs

Relevance: APIs are essential for handling frontend-backend communication.

Proposed Update:

- Implement rate limiting, input validation, and authentication for all APIs. Use API Gateway for secure API management.