An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

College of Engineering
Department of Computer Engineering

Project Part 2: Online Book Store “Bazar.com”

Students Names	Omar Maher Khatib	Osaid Raddad
Student ID	12144231	12111962
Course Name	Distributed Operating Systems	

Submission Date: December 21, 2025

❖ How does our project work?

3.1 Search Operation (with Caching)

- ✓ Client Request: GET /catalog-service/search/education
- ✓ The frontend receives the request and checks Redis for the key: search:education
- ✓ Cache HIT scenario
  - Data is found in Redis
  - The response is returned immediately (0–1 ms response time)
  - Log message: Cache HIT for “education” (1 ms)
- ✓ Cache MISS scenario
  - Data is not found in Redis or the cache has expired
  - The request is forwarded to the catalog service through the load balancer
  - The catalog service queries the SQLite database
  - The frontend caches the result in Redis with a 1-hour TTL
  - The response is returned to the client (3–6 ms response time)
  - Log message: Cache MISS for “education”, completed in 5 ms (cache updated)

3.2 Purchase Operation (with Cache Invalidation)

- ✓ Client Request: POST /order-service/purchase with body: { book_id: 9 }
- ✓ The frontend forwards the request to the order service
- ✓ The order service requests the catalog service to update inventory

- ✓ Catalog service (write master) process:
  - Step 1: Send cache invalidation request to the frontend
    The frontend is notified to invalidate cached data related to the book
  - Step 2: Update the database
    The number of items for the selected book is reduced by one in the database
  - Step 3: Sync the update to the read replica
    The updated book count and timestamp are sent to the catalog replica

- ✓ Frontend clears cache entries
  - Deletes info:9
  - Deletes all search cache entries that include book ID 9
- ✓ The response is returned to the client with the updated inventory count

Consistency Guarantee: Cache invalidation occurs before the database write, ensuring that no stale data is served to users.

3.3 Load Balancing in Action

- ✓ For three consecutive search requests:
  - Request 1 is routed from the frontend to Catalog Replica 1
  - Request 2 is routed from the frontend to Catalog Replica 2
  - Request 3 is routed from the frontend back to Catalog Replica 1 after the index wraps around

This approach distributes load evenly across catalog replicas and prevents any single replica from becoming a performance bottleneck.

❖ Testing the System

Search for New Books:

```powershell
# Education books
Invoke-RestMethod -Uri "http://localhost:8083/catalog-service/search/education"

# Nature book
Invoke-RestMethod -Uri "http://localhost:8083/catalog-service/search/nature"
```

Get Book Details:

```powershell
Invoke-RestMethod -Uri "http://localhost:8083/catalog-service/info/9"
```

Test Cache Performance:

```powershell
# First call (cache MISS)
Invoke-RestMethod -Uri "http://localhost:8083/catalog-service/search/systems"

# Second call (cache HIT)
Invoke-RestMethod -Uri "http://localhost:8083/catalog-service/search/systems"
```

Run Performance Experiments:

```powershell
powershell -ExecutionPolicy Bypass -File .\run-performance-experiments.ps1
```

Outputs are available in: program-output.txt in docs