Case study round submission guidelines

- Follow this template while working on your submission
- Keep your submission under 10 slides maximum (excluding slides 1 & 2)
- Download these slides as a PDF for submission. Only PDF format will be accepted
- Save your file in this name format: Name_University_YearofGrad
- (e.g. BabarAzam_IBA_2025)
- Be creative and have fun!

Evaluation Criteria

- **Clarity & Structure:** Presenting ideas in a logical and easy-to-follow manner.
- **Understanding the Numbers**: Being able to pull valuable insights from the key metrics and KPIs we're tracking.
- **Practical Steps for Improvement:** Coming up with actionable strategies that we can realistically implement to move forward.
- **Conciseness:** Keeping things clear, simple, and to the point without unnecessary fluff.
- **Impact - Driven:** At Bazaar, Every move we make is about creating meaningful change and making a real difference.

Who Am I?

Muhammad Anas Khan – A Developer Obsessed With Clean Code & Real-World Impact 🚀

Content:

- 🧑‍💻 **Computer Science Student** (FAST NUCES, Karachi – Roll: 22K-4170)
- 💡 Backend & Database enthusiast.
- 🔧 Passionate about building meaningful solutions and learning fast.
- 🧠 Created this backend solo from scratch as part of **Bazaar's Case Study Round**

"This case study wasn't just an assignment — it was an opportunity to build something real."

GitHub Repo link:

- https://github.com/Anacex/BAZAAR

Case Study Project – Kiryana Store Inventory Management Backend

بازار
BAZAAR

Content:

📦 Designed & developed a backend system to track stock movements for kiryana stores

🎯 Started with a single store, scaled to support **500+ stores**

🔐 Implemented authentication, stock handling, and data isolation

🧰 Tech Stack: Node.js, Express, PostgreSQL, Redis, JWT, BCrypt

```
backend/
├── config/
│   ├── db.js              # PostgreSQL connection
│   ├── redisClient.js     # Redis setup
│   └── eventEmitter.js    # Event logging utility
├── database/
│   ├── schema.sql         # DB schema
│   ├── populate.sql       # Sample data
├── middleware/
│   └── authMiddleware.js  # JWT checker
├── routes/
│   ├── authRoutes.js      # Login, Register
│   ├── stockRoutes.js     # Inventory endpoints
├── server.js              # Server entry point
├── SingleStoreBackendFiles/
└── .env                   # Environment config
```

Stage 1: Single Store Inventory

بازار
BAZAAR

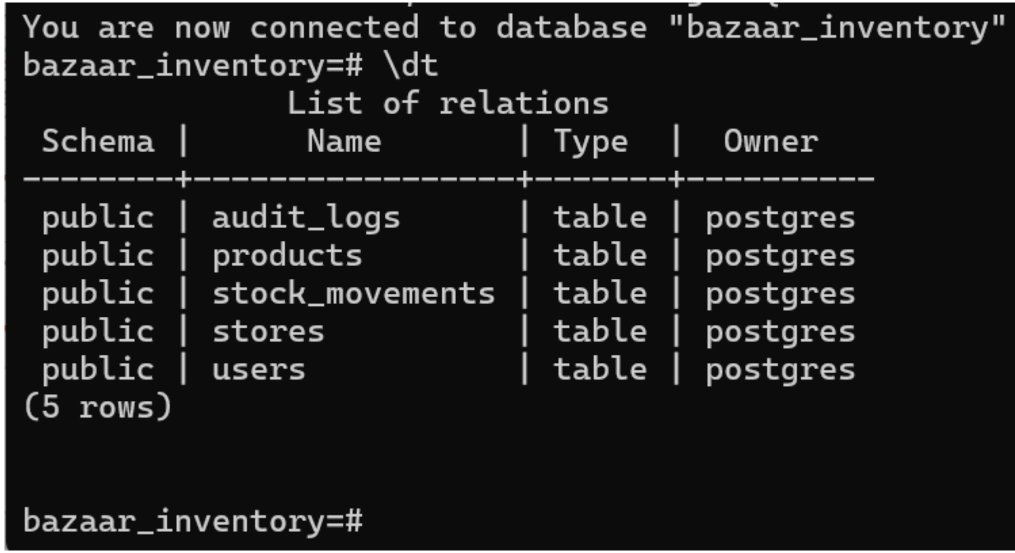
- ✅ **Built a backend using Node.js, Express.js, and MySQL** to handle inventory for a single store.
- 🗂️ **Database schema** with **3 core tables**:
  1) `products` – product catalog
  2) `stock_movements` – stock in/out tracking types (in, sale, removal)
  3) `users` – login credentials for authentication
- 🔁 Developed **Express routes** for:
  1) Adding/removing products
  2) Logging stock activity
  3) User login + basic token authentication
- ⚠️ **Challenges faced**:
  1) Schema design without flat files or ORMs like Sequelize
  2) Ensuring referential integrity with foreign keys and ENUM types
- 📁 Archived in:
  `BAZAAR/backend/SingleStoreBackendFiles/`
- 📌 See testDB.js for example queries and README.md for setup & usage

Stage 2: Thoughtful Relational Schema Re-Design

Content:

- 🔗 Relationships designed to isolate store-specific data
- ✅ Checks added for data integrity (e.g., `CHECK(type IN (...))`)
- 🔐 Secure user storage with hashed passwords via `Bcryptjs`
- Schema given in `backend/database/schema.sql`

📘 **Tables:**



```
You are now connected to database "bazaar_inventory"
bazaar_inventory=# \dt
              List of relations
 Schema |       Name        | Type  |  Owner
--------+-------------------+-------+----------
 public | audit_logs        | table | postgres
 public | products          | table | postgres
 public | stock_movements   | table | postgres
 public | stores            | table | postgres
 public | users             | table | postgres
(5 rows)


bazaar_inventory=#
```

Stage 2: Secure Authentication with JWT

Content:

- 🔐 **Passwords hashed** with `bcryptjs` before storage
- 🪪 **JWT tokens** generated on successful login
- 🧱 **Middleware** (`authMiddleware.js`) restricts access to authenticated users only
- 🏪 Token contains `store_id` to enforce **store-level access control**
- 🔄 **Tokens expire** in 1h to reduce security risks

🔧 Example:

```
const token = jwt.sign({ user_id, store_id }, JWT_SECRET, {
expiresIn: '1h' });
```

Stage 3: Redis-Powered Caching for Performance

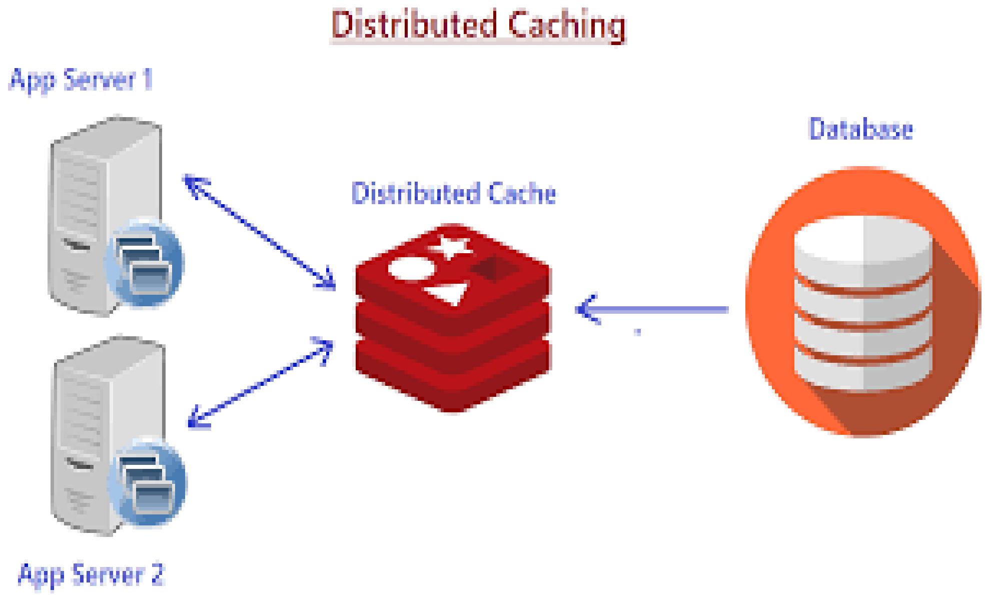
بازار
BAZAAR

Why Redis?

- Speed: In-memory store = ultra-fast access
- Reduced DB load: Common queries don’t hit PostgreSQL every time

🔧 Implementation:

- Cached results for:
  - Low stock products
  - Expensive date-range queries

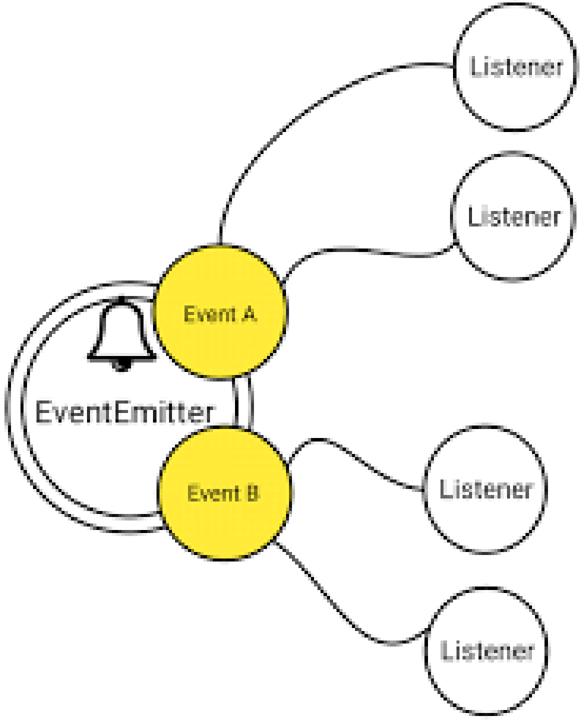


Stage 3: Centralized Event Logging with Node.js EventEmitter

BAZAAR

🛠 Why EventEmitter?

- Centralizes logging logic
- Keeps core logic clean
- Ready for future Webhooks / async queue handling
- 🛠️ Created eventEmitter.js to log actions like create/update/delete
- 📜 All mutations emit events with metadata
- ✅ Audit logs saved in audit_logs table
- 🔧 Designed to be extensible for real-time or async handling later



Stage 3: RESTful APIs for Stock & Store Management

بازار
BAZAAR

📌 Core Endpoints:

Method	Route	Description
POST	/api/auth/register	Register user
POST	/api/auth/login	Login + JWT
GET	/api/stock/	List all products
POST	/api/stock/movement	Log stock movement
GET	/api/stock/stores	Get all stores
GET	/movements?start=...&end=...	Filter by date range

💡 Sample curl Usage:

```
# Add stock
curl -X POST
http://localhost:5000/api/stock/
movement \
  -H "Authorization: Bearer
<token>" \
  -d '{"product_id": 1,
"quantity": 10, "type":
"stock_in"}'
```

Main Challenges

بازار
BAZAAR

Content:

- ⚙️ Migrating from MySQL to PostgreSQL mid-way
- ⏱️ Managing time, learning curve, and async patterns alone
- 🧪 No frontend – tested all routes using `curl`
- 🔄 Realized the value of modular design for scaling

```
//Register New User (CMD)

curl -X POST
http://localhost:5000/api/auth/register ^
-H "Content-Type: application/json" ^  -d
"{\"username\":\"store_manager\",\"passwo
rd\":\"1234\"}"

//Login & Get Token(CMD)

curl -X POST
http://localhost:5000/api/auth/login ^
  -H "Content-Type: application/json" ^
  -d
"{\"username\":\"store_manager\",\"passwo
rd\":\"1234\"}"
```

What would be next?

بازار
BAZAAR

Content:

- 🔐 **Role-Based Access Control (RBAC)**
Granular permissions for Admins, Managers, and Regional Leads to ensure secure, multi-tier access.
- 📊 **Store-Wide Analytics Dashboard**
Expose APIs for top-selling products, stock velocity, and category trends — helping drive smarter business decisions.
- 🐳 **Dockerization & CI/CD Integration**
Containerize the backend and configure CI/CD pipelines (e.g., GitHub Actions) for zero-downtime deployment and maintainability.
- 🚨 **Real-Time Low Stock Alerts via WebSockets + Redis Pub/Sub**
Notify store managers instantly when critical thresholds are hit.