WBD - Global Martech Engineering

Technical Assessment

Before you start

Please review the solution overview and main requirements below. This assessment aims to evaluate your experience level and technical depth in each area from L1 (Lowest) to L5 (Highest) based on what you're familiar with.

The levels are cumulative across all areas. When selecting a level in any particular area, you must ensure that you cover all lower-level requirements. Higher-level requirements cannot be fulfilled without first establishing the prerequisites from previous levels.

For other areas, level selection should be based on your understanding and vision of the complete solution—not simply following numerical order. The primary goal is to deliver a comprehensive, integrated solution across all areas and tiers.

In general, the goal isn't to reach the highest level in every area, but to demonstrate your basic understanding in areas outside your main speciality or comfort zone. This helps us assess your expertise from an engineering perspective and mindset.

You don't need to select the same level across all areas. For example, if you choose level 2 in Database, you can select different levels in other areas. Base your selections on your knowledge, but ensure consistency and integrity between areas to maintain reliable dependencies.

Recommendations and Notes

- Submit all deliverables to your public personal GitHub repository.
- Organize your repository with a clear structure and include a comprehensive README file as documentation. Consider adding a deployment guide as well.
- In your README file, specify your chosen level for each area (e.g., "Database: L3") and briefly explain your selections.
- Taking one of the challenges is Mandatory and should be part of your solution.
- Use **ONLY** the **MERN** stack (with **TypeScript**) and **AWS** for implementation. No other technology stacks will be accepted. Make sure to use **MongoDB** as your main DB for the solution.
- The required dashboard is related to system services and health status that should be connected with basic cloud metrics (**Grafana** could be a goot tool to be used). It is not related to business data.
- Describe any design patterns or architectural principles you implement in your solution.
- Adhere to standard naming conventions and coding practices.
- Feel free to include additional components, ideas, or diagrams that enhance system efficiency, scalability, or presentation (e.g., configurations, system alerts, AI integration).
- Complete and submit your work within **5-7** calendar days of confirming receipt of our email.
- After submission, we will schedule a review meeting to discuss your approach, implementation decisions, and technical choices.

Solution Overview

An online shopping company (like Amazon.com) wants to build a user-tracking platform that shows user journeys over specific time periods. The system needs to cover search functionality for individual users, with the display focusing on user behaviour through various events: session start time, number of pages/items visited, purchase count, and time spent on specific pages (you can specify more events or actions to cover, but at least select 4).

Assuming you already receive these events through an integration point of your choice, focus on designing the data model and determining the best storage approach to meet the search and display requirements. Showing some business KPIs or graphs will be a good addition to your solution.

Build your solution assuming that you get 1M events daily, with a variety of event types. Also, consider that user profiles count not less than 500K.

Solution Requirements

Database

L1: DB Schema/diagram for main system entities and data fields.

L2: DB implementation using free-tier cloud service (MongoDB).

L3: Include in your design an integration point with another system to import articles from different sources (a data flow diagram is required).

Backend/API

L1: API Documentation with all essential endpoints (Using Swagger or something similar will be good).

L2: Full API implementation. It should be working and hosted locally (The API should be available for testing via tools like Postman).

L3: API Deployment/hosting using free-tier cloud service (The API should be available for testing via tools like Postman).

L4: API implementation using free-tier cloud service with basic unit testing.

L5: Backend Job to export/transfer analytical data to another system daily at 10 UTC.

Cloud/DevOps

L1: System Diagram showing main system components(cloud services) and the data flow or workflow steps.

L2: Actual Cloud Infrastructure for the system components (fully running system on Cloud)

L3: Basic Deployment (CI/CD) pipeline using GitHub Actions to deploy code updates

L4: Infrastructure automated deployment using any approach like CloudFormation or Terraform, or infra as code

FrontEnd

L1: Basic and sample wireframes to provide the idea (pages/views) with some user stories

L2: More advanced prototype or MVP to present the idea (Pages transition and basic actions, but not connected to the Backend API)

L3: Simple Web App (hosted at least locally) with data display/visualization views/pages connected to Backend API (no data entry forms)

L4: More advanced Web App (hosted at least locally) with data display/visualization views/pages and data entry forms (connected to Backend API)

L5: Full Web App (hosted using free-tier cloud service) with data display/visualization views/pages and data entry forms (connected to the Backend API)

L6: Full Web App as in L5 with Admin Control Panel

Dashboards - System Services Monitoring and Support

L3: Simple dashboard (on Grafana free-tier or any other tool) for main system components to be used in monitoring services' health and performance. Using CloudWatch logs will be sufficient

Bonus/Nice to have

L4: Dashboard for data analytics showing some KPIs and stats for user insights, like most active users, most visited pages, or products, or any important analytics to show.

Challenges (Must pick at least one to be covered in your solution design/implementation):

- System active users increased by 10 times.
- More integration points added/requested to enrich the user data and display analytics.
- Increased number of supported event types.
- Increased number of daily events by 5 times.
- Instead of English only (interface or data), supporting different languages is needed.
- Instead of one-country support, the System should support multiple countries in different regions.
- The System should be highly available 247, with different environments.
