Value First Homework Assignment

Product Update Service Simulator

We're looking for a senior Go developer, and this assignment will help us assess your technical expertise, system design skills, and problem-solving abilities. Your task is to build a small Go service that simulates an e-commerce platform's product update flow, demonstrating idiomatic Go programming, concurrency, and API design.

Estimated time investment:

This assignment should take approximately 1-2 hours to complete with LLM assistance (e.g., for ideation or boilerplate code). The core design and implementation should reflect your expertise.

Assignment Requirements

Build a Go service with the following functionality:

1. API Endpoints

- **POST /events**: Accepts JSON payloads representing product updates (e.g., `{ "product_id": "abc123", "price": 49.99, "stock": 100 }`). Each event should update the product's state (later events override earlier ones for the same product_id). Enqueue the event into an in-memory queue and return a `202 Accepted` response immediately.

- **GET /products/{id}**: Retrieves the current state of a product (price and stock) from an in-memory store.

2. Asynchronous Processing

- Implement a worker pool with 3-5 configurable workers to process events asynchronously from the in-memory queue.
- Update a thread-safe in-memory store with the latest product data (price and stock). Ensure concurrency safety using appropriate Go primitives.

3. Testing

- Include tests for concurrency safety and API endpoints.

4. Documentation

Provide a `README.md` that includes:

- **Setup Instructions**: How to build and run the service (e.g., `go run main.go`).
- **Design Choices**: Explanation of your architectural decisions and implementation approach.
- **Production Considerations**: Suggestions for real-world improvements, addressing:
  - How you'd integrate production tools like RabbitMQ (for queuing), PostgreSQL or Redis (for persistence).
  - Strategies for handling large-scale data and high throughput.
  - Error handling and retry mechanisms for failed updates.
- **Troubleshooting Strategies**: How you would approach debugging common issues:
  - Data consistency problems
  - A specific scenario: products aren't updating despite events being received

---

Submission Instructions:

- Please use this form to submit your solution.
  - *or email cab@valuefirstconsulting.com if you have any issues with the form above.*
- Submit your solution as a GitHub repository or ZIP file containing all code, tests, and the README.
- You may use LLMs for boilerplate or ideation, but the core logic, concurrency patterns, and README analysis should reflect your expertise.
- Focus on clean, idiomatic Go code that demonstrates concurrency safety, error handling, and API design.
- **Timeline:** All candidates are expected to submit their assignments no later than 72 hours post instructions are sent.

What We're Looking For:

We will evaluate your submission on:

- **Code Quality**: Idiomatic Go, proper use of concurrency primitives, clean structure
- **Concurrency Safety**: Thread-safe operations, proper synchronization
- **System Design Thinking**: README insights on production considerations, scalability, and troubleshooting
- **API Design**: RESTful principles, appropriate status codes, error responses

Example Payloads

POST /events:

```
{
  "product_id": "abc123",
  "price": 49.99,
  "stock": 100
}
```

GET /products/abc123 (response):

```
{
  "product_id": "abc123",
  "price": 49.99,
  "stock": 100
}
```

Any questions?

Please use this form.