Implementation Plan for Inventory Management REST API Service in Golang

# 1. Architecture Overview

High-Level Architecture:  
Client → API Gateway → Load Balancer → ECS/EKS (Dockerized Go App) → RDS/DynamoDB  
 ↘︎ CloudWatch (Monitoring & Logging)

# 2. Tools & Technologies

|  |  |
| --- | --- |
| **Category** | **Tools/Technologies** |
| Language | Golang |
| Containerization | Docker (multi-stage builds) |
| Orchestration | AWS ECS (Fargate) |
| Database | Amazon RDS (PostgreSQL) |
| API Management | AWS API Gateway |
| Infrastructure as Code | AWS CloudFormation |
| Monitoring & Logging | AWS CloudWatch, structured logging (Zap/Logrus) |
| CI/CD | GitHub Actions / AWS CodePipeline |

# 3. System Design & Patterns

Design Patterns:  
- Repository Pattern: Abstract DB operations.  
- Factory Pattern: For creating service instances.  
- Middleware Pattern: For logging, authentication, error handling.  
- DTOs (Data Transfer Objects): For request/response validation.  
  
Concurrency Handling:  
- Use Goroutines for concurrent request handling.  
- Use Channels or sync.Mutex/RWMutex for safe concurrent access to shared resources.

# 4. API Design

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Description** |
| POST | /products | Add a new product |
| PUT | /products/{id} | Update product details or stock |
| GET | /products/{id} | Get product info |
| DELETE | /products/{id} | Delete product |

Use JSON for request/response. Validate inputs using libraries like go-playground/validator.

# 5. Data Persistence

Amazon RDS  
- Use PostgreSQL  
- Ensure ACID compliance.  
- Use GORM or sqlx for ORM.

# 6. Deployment Strategy

Containerization:  
- Use multi-stage Docker builds for optimized images.  
  
Orchestration:  
- ECS (Fargate): Easier setup, serverless containers.  
  
Infrastructure as Code:  
- Use AWS CloudFormation to provision:  
 - VPC, Subnets  
 - ECS cluster  
 - RDS PostgreSQL  
 - API Gateway  
 - IAM Roles

# 7. Handling API Traffic Load

Scalability:  
- Use API Gateway + Load Balancer.  
- Enable Auto Scaling on ECS.  
  
Performance Optimization:  
- Optimize DB queries.  
- Use connection pooling.  
- Cache frequent reads with Redis.  
  
Rate Limiting & Throttling:  
- Configure API Gateway for rate limits.

# 8. Security Considerations

- Use HTTPS via API Gateway.  
- Implement IAM roles for service access.  
- Validate all inputs to prevent injection attacks.

# 9. Monitoring & Logging

- Use structured logging with Zap or Logrus.  
- Send logs to CloudWatch Logs.  
- Set up CloudWatch Alarms for error rates, latency, etc.

# 10. CI/CD Pipeline

- Use GitHub Actions to lint, test, build Docker image.  
- Push image to Amazon ECR.  
- Deploy to ECS using CloudFormation.