Golang Assignment

Objective

Build a **scalable RESTful Task Management Service** in Go demonstrating **API design, persistence, concurrency, authentication, and clean architecture**.

Problem Statement

Design and implement a **Task Management REST API**.

Core Requirements

REST APIs

Implement the following endpoints:

- POST /tasks – Create a task
- GET /tasks – List tasks
- GET /tasks/{id} – Get task by ID
- DELETE /tasks/{id} – Delete task

Task Model

```
{
  "id": "string / uuid",
  "title": "string",
  "description": "string",
  "status": "pending | in_progress | completed",
 "created_at": "timestamp",
"updated_at": "timestamp"
}
```

Persistence

- Use **SQL (MySQL / PostgreSQL)** or **NoSQL (MongoDB)**
- Implement a clean repository/data access layer

Authentication & Authorization

- Implement **authentication** using **JWT**
- Add a simple **User model** (id, email/username, role)
- Protected APIs must require a valid JWT
- Implement **authorization**:
  - Users can access **only their own tasks**
  - Admin users can access **all tasks**
- Token expiry and proper error handling expected

Concurrency

Background Worker – Auto-Complete Tasks

Implement a **background worker** that automatically marks a task as completed after **X minutes** if it is still in pending or in_progress.

Requirements:

- Auto-completion delay (**X minutes**) configurable via environment variable
- Tasks manually completed or deleted before X minutes must **not** be auto-completed
- Updated task state must be persisted to the database

Concurrency Expectations:

- Use **goroutines** for background processing
- Use **channels / worker queue** to send task IDs
- Ensure thread-safe access to shared resources
- Background processing must **not block API requests**

Error Handling & Validation

- Proper HTTP status codes
- Input validation
- Consistent JSON error responses

Code Quality & Design

- Idiomatic Go code
- Clear folder structure
- Separation of concerns (handlers, services, repositories)
- Configuration via environment variables

Bonus (Optional)

- Pagination & filtering
- Unit tests
- Dockerfile
- Swagger / OpenAPI
- Graceful shutdown using context.Context
- Logging & metrics
- Rate limiting or middleware

Submission Guidelines

- Push code to **GitHub**
- Include a **README.md** with implementation details