

Golang Intern Assignment Document

Title: HRMS System (Attendance & Student Management) - API Implementation Assignment

Objective

Evaluate the candidate's proficiency in **Golang**, **Gin framework**, **MVVC architecture**, **API design**, **testing**, and **cron job development** by building a structured backend system.

Project Overview

Develop a minimal **HRMS System API** that includes: 1. **Student Management** – Add, Update, List, Delete students 2. **Attendance Management** – Mark and View attendance 3. **Cron Jobs** – Send weekly and monthly attendance reports to students

The system must follow the **MVVC (Model-View-ViewModel-Controller)** pattern.

1. Technology Stack

- **Language:** Golang
- **Framework:** Gin (mandatory)
- **Database:** MySQL (mandatory)
- **ORM (Optional):** GORM / SQLC / PGX
- **Cron Scheduler:** robfig/cron (v3)
Data Schema : Mysql / Postgres
- **Testing:** Go's built-in testing framework

2. Functional Requirements

A. Student Module

API Endpoints

- **POST /students** – Create student
- **GET /students** – List all students

- **GET /students/:id** – Get student by ID
- **PUT /students/:id** – Update student
- **DELETE /students/:id** – Delete student
- **Fields**
- **id**

1

```
    . . . . ent
    name   created_
    email   at
    departm
```

B. Attendance Module

API Endpoints

- **POST /attendance/mark** – Mark attendance
- **GET /attendance/:student_id** – View attendance

Fields

```
    . . . . status
    id          (Present/Abse
    student_id nt)
    date
```

3. Cron Job Requirements

Implement automated scheduled tasks to notify students of attendance summary.

Cron Functionalities

- Generate **weekly attendance report** per student
- Generate **monthly attendance report** per student
- n methods:
- Notificatio
- Console output (minimum requirement)

```
..          notification  
Email      (bonus)  
           Save generated summary into database (optional)
```

Cron Expectations

- Use **robfig/cron v3**
- Must follow clean architecture: cron → service → repository
- Should be isolated into a separate package/module
- Should be testable (mock DB/repositories)

5. Validation Requirements**

- Validate all input fields
 - Email format validation
 - Date validation
 - Reject duplicate emails
 - Return proper HTTP error codes

2

6. Testing Requirements

Candidate must include unit tests for: - Controllers - Services - Repositories (mocked) - Validation logic

Minimum Requirements

- **60%+ test coverage**
- Mocks or interfaces for DB interactions

7. Deliverables

Candidate must submit: 1. GitHub repository containing full source code 2. README with setup instructions 3. Postman collection or Swagger API documentation 4. Tests included in repository

8. Evaluation Criteria

.

- Code quality & readability
 - Proper MVVC implementation
 - API correctness
 - Test quality & coverage
 - Clean architecture usage
 - Cron implementation cleanliness
 - Execution quality
- Documentation

9. Optimization Notes

To help candidates focus on performance and code quality, include the following optimization considerations:

API Optimization

- Use context timeouts for all DB operations.
- Implement pagination for all list endpoints.
- Avoid N+1 queries by using joins where helpful.

Database Optimization

- Add indexes on frequently queried columns (e.g., email, student_id, date).
- Use prepared statements or connection pooling.
- Normalize tables properly but avoid over-normalization.

Application Optimization

- Reuse Gin router and DB connections (singleton pattern).
- Use lightweight DTOs in ViewModel for faster serialization.
- Cache repetitive queries (optional / bonus). 3

Cron Optimization

- Batch process attendance summaries.
- Avoid generating duplicate reports by tracking last execution timestamps.
- Use efficient queries with grouping and indexing.

10. Bonus Points (Optional)

- Docker support
- Swagger auto-generation
- CI/CD for tests
- Database migration tool

Submission Deadline

Candidate must complete and submit within **48–72 hours** of receiving the assignment.

End of Document