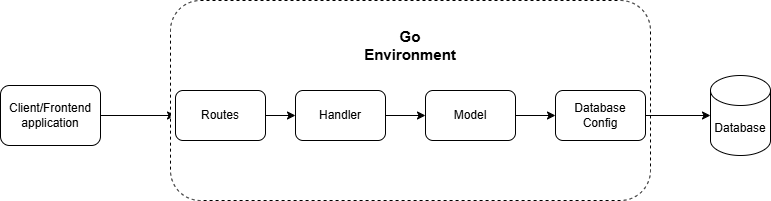
Employee Payroll System - Technical Guide



1. **Overview**

This document provides a comprehensive guide to the **Employee Payroll System**, a backend application developed in Go (Golang) designed to automate salary calculations based on attendance, overtime, and reimbursement submissions

1. **Architecture Design**



Definition :

1. Client is frontend application who consumed the endpoint of API
2. Routes is endpoint API for handle request from client. The request exposed as a JSON and then it will forward to handler for business logic
3. Business logic (handler) will mapping the request to match with the data model before execute to database
4. If data match with the data model of database, API will communicate (execute sql query) to database using database config.
5. If no errors with the request, data will be create on database and response will appears.
6. Client can retrieve data as a request
7. **Key Features**
8. Prorate salary calculation based in attendance records
9. Overtime compensation (2x prorate rate)
10. Reimbursement request
11. Salary slip generated
12. Payroll report summary
13. Fake data generate : 100 employees + 1 admin
14. Attendance period management
15. **System Roles**
16. Admin

* Create new attendance periods
* View and approve reimbursements
* Generate and view payroll reports

1. Employee

* Submit daily attendance and overtime
* Submit reimbursement requests
* Generate salary slip

1. **Data Model**
2. Attendance

type Attendance struct {

    gorm.Model

    EmployeeId string

    PeriodId   string

    Date       time.Time `gorm:"index"`

}

1. Attendance period

type AttendancePeriod struct {

    gorm.Model

    StartDate   time.Time

    EndDate     time.Time

    IsProcessed bool

    PeriodId    string

}

1. Employee

type Employee struct {

    gorm.Model

    ID         uuid.UUID

    EmployeeId string

    Username   string

    Password   string

    Role       string

    IsAdmin    bool

    Token      string

    Salary     float64

}

1. Overtime

type Overtime struct {

    gorm.Model

    EmployeeId string

    Date       time.Time `gorm:"index"`

    Hours      int

    PeriodId   uint

}

1. Payslip

type Payslip struct {

    gorm.Model

    PeriodID           string

    EmployeeID         string

    AttendanceDays     int

    OvertimeHours      float64

    Reimbursement      float64

    BaseSalary         float64

    TotalOvertime      int

    TotalReimbursement float64

    TotalPay           float64

    GeneratedAt        time.Time

}

1. Reimbursement

type Reimbursement struct {

    gorm.Model

    EmployeeId  string

    Amount      float64

    Date        time.Time

    Description string

    PeriodId    uuid.UUID

}

1. **API Endpoint**

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoint** | **Sample Request** |
| POST | {host:port}/admin/attendance-period | {      "period\_id" : "payroll-2025-05",      "start\_date" : "2025-05-01",      "end\_date" : "2025-05-31"  } |
| POST | {host:port}/admin/payroll/:period-id | Parameter :  period-id : payroll-[yyyy]-[mm] |
| GET | {host:port}/admin/payroll/summary/:period-id | Parameter :  period-id : payroll-[yyyy]-[mm] |
| POST | {host:port}/employee/attendance | {      "employee\_id" : "employee\_1",      "date" : "2025-05-30"  } |
| POST | {host:port}/employee/overtime | {      "employee\_id" : "employee\_1",      "date" : "2025-05-30",      "hours" : 3  } |
| POST | {host:port}/employee/reimbursement | {      "employee\_id" : "employee\_1",      "amount" : 200000,      "description" : "Data migration to production enviroment"  } |
| GET | {host:port}/employee/payslip/:period-id/:employee\_id | Parameter :  period-id : payroll-[yyyy]-[mm]  Employee\_id : employee\_[number] |

1. **Dummy Data Generator**

Here is the code to generate dummy data of 100 employees and 1 admin :

func InputData(db \*gorm.DB) {

    db.Exec("DELETE FROM employees")

    for i := 1; i <= 100; i++ {

        password := "password\_" + strconv.Itoa(i)

        hashed, \_ := bcrypt.GenerateFromPassword([]byte(password), bcrypt.DefaultCost)

        db.Create(&model.Employee{

            ID:         uuid.New(),

            Username:   "employee\_" + strconv.Itoa(i),

            EmployeeId: "employee\_" + strconv.Itoa(i),

            Password:   string(hashed),

            Salary:     float64(rand.Intn(5\_000\_000) + 3\_000\_000),

            IsAdmin:    false,

        })

    }

    hashedAdmin, \_ := bcrypt.GenerateFromPassword([]byte("adminPsswd"), bcrypt.DefaultCost)

    db.Create((&model.Employee{

        ID:         uuid.New(),

        Username:   "admin",

        Password:   string(hashedAdmin),

        EmployeeId: "admin",

        Salary:     0,

        IsAdmin:    true,

    }))

}