

# Payroll-api Documentation

1. Open payroll-api github link:  
<https://github.com/Sabd98/payroll-api>
2. Git clone it.
3. Then install dependencies with bun i or npm i
4. Set your env config like this:  
PORT = 3000  
  
DB\_USER=postgres  
  
DB\_PASSWORD=your\_password  
  
DB\_NAME=payroll\_management  
  
DB\_HOST=localhost  
  
JWT\_SECRET=your\_secret  
  
RUN\_SEEDERS=false
5. Create database
6. Then Run migrate with bunx/npx sequelize-cli db:migrate
7. Seed available dummy data with set RUN\_SEEDERS to true
8. Run server with node index.js/nodemon index.js
9. If run, you'll found database synced at end of log
10. Wait for many seconds, and dummy data has been seeded automatically.
11. You can try my api-documentation json called 'payroll-employee-api.postman\_collection.json' with import it to postman/thunderclient.
12. Okay finish. Now let's try many used api's.

## A. Login Admin. Post is so it'll generate token for admin. Expired in 1 hour.

POST http://localhost:3000/api/auth/login

Params Authorization Headers (9) Body Pre-request Script

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary

```
1 {
2   "username": "admin",
3   "password": "admin123"
4 }
```

Body Cookies Headers (22) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwicm9sZSI6ImFkbWluIiwiaWF0IjpmYXVpMgeE",
3   "user": {
4     "id": 1,
5     "username": "admin",
6     "role": "admin"
7   }
8 }
```

## B. Same as employee with different generate token

POST http://localhost:3000/api/auth/login

Params Authorization Headers (9) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "username": "employee1",
3   "password": "pass1"
4 }
```

Body Cookies Headers (22) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwicm9sZSI6ImVtcGxveWVlIiwiaWF0IjpmYXVpMgeE",
3   "user": {
4     "id": 1,
5     "username": "employee1",
6     "role": "employee",
7     "monthly_salary": "6683.00"
8   }
9 }
```

C. Use admin token in Authorization. Then bearer token. Input it. Next is post period like this with output respond if 201 status.

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:3000/api/admin/payroll-periods`
- Method:** POST
- Body:** A JSON object with the following structure:

```
1 {
2   "start_date": "2025-05-01",
3   "end_date": "2025-05-31"
4 }
```
- Response:** A JSON object with the following structure:

```
1 {
2   "id": 1,
3   "start_date": "2025-05-01",
4   "end_date": "2025-05-31",
5   "is_processed": false,
6   "created_by": 1,
7   "ip_address": "::1",
8   "updated_at": "2025-06-29T14:10:51.897Z",
9   "created_at": "2025-06-29T14:10:51.896Z",
10  "updated_by": null
11 }
```

D. Use user token. Next is post attendance like this.

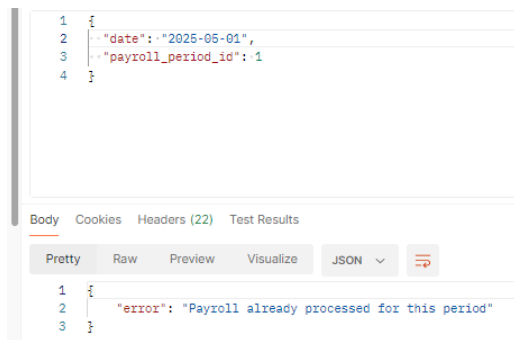
The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:3000/api/employee/attendances`
- Method:** POST
- Body:** A JSON object with the following structure:

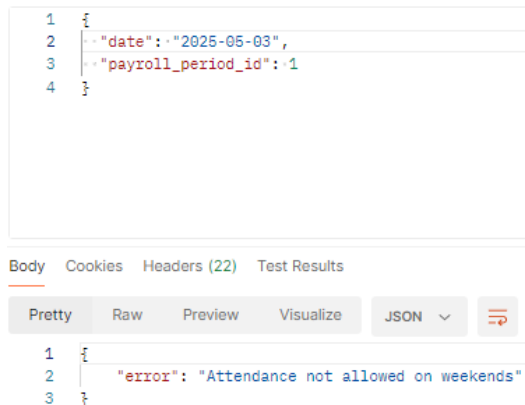
```
1 {
2   "date": "2025-05-01",
3   "payroll_period_id": 1
4 }
```
- Response:** A JSON object with the following structure:

```
1 {
2   "id": 1,
3   "employee_id": 1,
4   "date": "2025-05-01",
5   "payroll_period_id": 1,
6   "created_by": 1,
7   "ip_address": "::1",
8   "updated_at": "2025-06-29T10:31:28.179Z",
9   "created_at": "2025-06-29T10:31:28.179Z",
10  "updated_by": null
11 }
```
- Status:** 201 Created
- Time:** 385 ms
- Size:** 1.17 KB

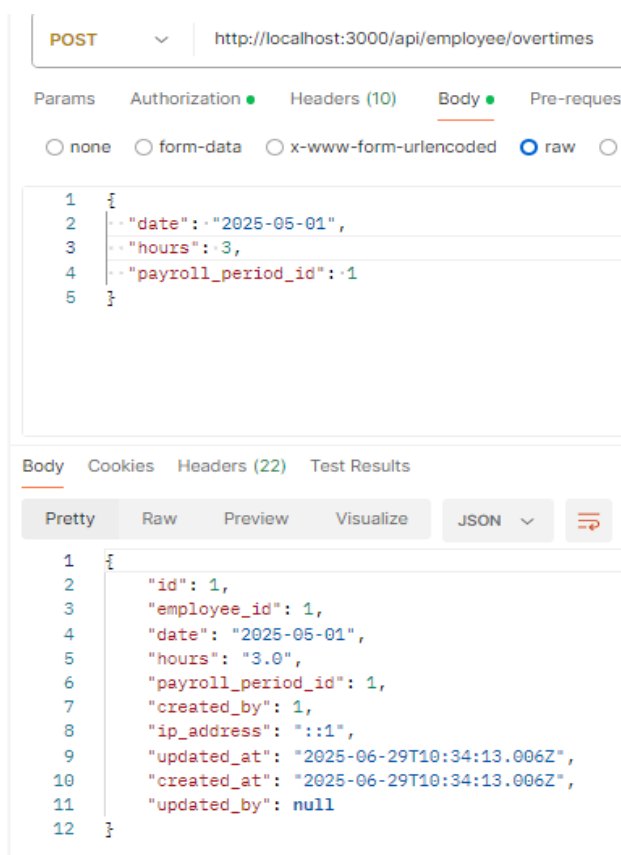
Validation when have check in on this date:



Validation when input on weekend (Saturday or Sunday):



E. Use user token. Next is post overtimes like this. And it works like this pic:



Validation when hourse More than 3 hours:

The screenshot displays a REST client interface with two main sections. The top section shows the request body as a JSON object: `{ "date": "2025-05-01", "hours": 4, "payroll_period_id": 1 }`. The bottom section, labeled 'Body', shows the response in 'Pretty' JSON format: `{ "error": "Overtime must be between 0.1 and 3 hours" }`. The interface includes tabs for 'Body', 'Cookies', 'Headers (22)', and 'Test Results', and a toolbar with options for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'.

```
1 {
2   "date": "2025-05-01",
3   "hours": 4,
4   "payroll_period_id": 1
5 }
```

Body Cookies Headers (22) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": "Overtime must be between 0.1 and 3 hours"
3 }
```

F. Use user token. Then input your reimbursement like this:

POST ▼ http://localhost:3000/api/employee/reimbursements

Params Authorization Headers (10) **Body** Pre-request Scr

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary

```
1 {
2   "amount": 80.00,
3   "description": "Work Allowance",
4   "payroll_period_id": 1
5 }
```

Body Cookies Headers (22) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "created_at": "2025-06-29T10:35:16.461Z",
3   "updated_at": "2025-06-29T10:35:16.461Z",
4   "submission_date": "2025-06-29",
5   "id": 1,
6   "employee_id": 1,
7   "amount": "80.00",
8   "description": "Work Allowance",
9   "payroll_period_id": 1,
10  "created_by": 1,
11  "ip_address": "::1",
12  "payslip_id": null,
13  "updated_by": null
14 }
```

G. Then use admin token. Run payroll with this request body. Choose specific period:

POST ▼ http://localhost:3000/api/admin/run-payroll

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼

```
1 {
2   "periodId": 1
3 }
```

Body Cookies Headers (22) Test Results Status: 200 OK Time: 1824 ms Size: 1.01 KB

Pretty Raw Preview Visualize JSON ▼

```
1 {
2   "message": "Payroll processed successfully"
3 }
```

If payroll processed succesfull, then process from this periodId and relations can't executed, like this example:

Left Request: POST `http://localhost:3000/api/employee/attendances`

Body (raw):

```
1 {
2   "date": "2025-05-06",
3   "payroll_period_id": 1
4 }
```

Left Response: Body (Pretty)

```
1 {
2   "error": "Payroll already processed for this period"
3 }
```

Right Request: POST `http://localhost:3000/api/admin/run-payroll`

Body (raw):

```
1 {
2   "periodId": 1
3 }
```

Right Response: Body (Pretty)

```
1 {
2   "error": "Payroll already processed for this period"
3 }
```

Request: POST `http://localhost:3000/api/admin/run-payroll`

Body (raw):

```
1 {
2   "periodId": 1
3 }
```

Response: Body (Pretty)

```
1 {
2   "error": "Payroll already processed for this period"
3 }
```

H. Use employee token. Then fetch a payslip based on chosen period :

Request: GET `http://localhost:3000/api/employee/payslips/1`

Response: Body (Pretty)

```
1 {
2   "period_id": "1",
3   "base_salary": "411.27",
4   "overtime_pay": "171.40",
5   "total_reimbursements": "80.00",
6   "reimbursements": [
7     {
8       "id": 1,
9       "amount": "80.00",
10      "description": "Work Allowance",
11      "date": "2025-06-29T10:35:16.461Z"
12    }
13  ],
14   "total_pay": "662.67"
15 }
```

I. Use employee token. Then fetch payroll periods list:

The screenshot shows a Postman interface for a GET request to `http://localhost:3000/api/employee/payroll-periods`. The 'Authorization' tab is selected, showing 'Bearer Token' as the type. The 'Body' tab is also visible, displaying a JSON response in 'Pretty' format:

```
1 {
2   {
3     "id": 1,
4     "start_date": "2025-05-01",
5     "end_date": "2025-05-31"
6   }
7 }
```

J. Have to login based on auth role. Have validation like these examples:

The image displays two side-by-side Postman screenshots. The left screenshot shows a POST request to `http://localhost:3000/api/admin/run-payroll` with a 'Bearer Token' authorization. The response body shows an error: `"error": "Admin access required"`. The right screenshot shows a GET request to `http://localhost:3000/api/employee/payroll-periods` with a 'Bearer Token' authorization. The response body shows an error: `"error": "Employee access required"`. Both screenshots show the 'Authorization' tab selected and the 'Body' tab displaying the JSON error response.



k. Last is Payslip Summary. Use admin token to fetch it:

GET http://localhost:3000/api/admin/payslip-summary/1

Params Authorization Headers (8) Body Pre-request Script Tests Se

Type Bearer Token

Heads up! These parameter variables. Learn more about

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token

Body Cookies Headers (22) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "period_id": "1",
3   "payslips": [
4     {
5       "id": 1,
6       "base_salary": "516.15",
7       "overtime_pay": "215.10",
8       "total_reimbursements": "80.00",
9       "total_pay": "811.25",
10      "Employee": {
11        "id": 1,
12        "username": "employee1"
13      },
14    },
15    {
16      "id": 2,
```

L. Testing:

i. Unit Testing

-Run npm/bun run test:unit. And I have test that and passes all:

```
Test Suites: 3 passed, 3 total
Tests: 11 passed, 11 total
Snapshots: 0 total
Time: 2.461 s
Ran all test suites matching tests/unit.
✓ Database connection closed
```

- Payroll service Test:

```
PASS tests/unit/services/payrollService.test.js
  • Console

  console.log
    [dotenv@17.0.0] injecting env (7) from .env - 🔒 encrypt with dotenvx: https://dotenvx.com
    at _log (node_modules/dotenv/lib/main.js:102:11)
```

- Date Utils

```

PASS tests/unit/utis/payroll.test.js
  ● Console

Determining test suites to run...[dotenv@17.0
✓ Database initialized for tests
PASS tests/unit/utis/dateUtils.test.js
  ● Console
```

ii. Integration Testing. Run npm/bun run test:integration.

```

at console.error (node_modules/@jest/console/build/index.js:124:10)
PASS tests/integration/controllers/authController.test.js
  ● Console

console.log
  [dotenv@17.0.0] injecting env (7) from .env - 🔒 encrypt with dotenvx: https://dotenvx.com
    at _log (node_modules/dotenv/lib/main.js:102:11)

console.log
  ❖ Starting tests in test environment
    at Object.log (tests/jest.setup.js:8:9)

console.log
  [dotenv@17.0.0] injecting env (7) from .env - 🔒 encrypt with dotenvx: https://dotenvx.com
    at _log (node_modules/dotenv/lib/main.js:102:11)
```

