Test Backend – Postman

**Setup**

* Environment var: baseUrl = http://localhost:5001
* Headers for all JSON requests: Content-Type: application/json
* No auth middleware is visible in your snippets, so all routes appear public.
* All API routes are mounted under /api except /health.

**Health**

**GET** {{baseUrl}}/health

**Expect 200**

{ "status": "OK", "timestamp": "2025-08-10T20:21:00.000Z" }

**Auth**

* **Login :**
  + **POST** {{baseUrl}}/api/login
  + Body: { "username": "admin", "password": "secret123" }
  + **Expect 200:**

{

  "success": true,

  "token": "JWT\_STRING",

  "user": { "id": "mongoId", "username": "admin", "isAdmin": true, "department": "QA" }

}

* + **Negative**:
    - Missing fields → 400 { "message": "Username and password are required" }
    - Bad creds → 401 { "message": "Invalid username or password" }
* **Register (optional):**
  + **POST** {{baseUrl}}/api/register
  + Body: { "username": "newuser", "password": "secret123", "isAdmin": false, "department": "Assembly" }
  + **Expect 201** with created user (without password).
  + **Negative**:
    - Existing username → 400 "Username already exists"
    - Short password (<6) → 400
* **Forgot Password**
  + **POST** {{baseUrl}}/api/forgot-password
  + Body: { "username": "admin" }
  + **Expect 200 :** { "message": "Password reset instructions sent to your email" }
  + **Negative**: unknown user → 404
* **Reset Password**
  + **POST** {{baseUrl}}/api/reset-password
  + Body:{ "username": "admin", "newPassword": "newsecret123" }
  + **Expect 200** { "message": "Password reset successfully" }
  + **Negative**: missing fields → 400; user not found → 404; weak pw → 400

**Employees**

* **Get all employees:**
  + **GET** {{baseUrl}}/api/employees
  + **Expect 200** [] or list of persons:

[{

"\_id": "mongoId",

"person\_id": "E001",

"first\_name": "John",

"last\_name": "Doe",

"department": "Assembly",

"status": "פעיל",

"role": "Employee"

}]

* **Create employee**
  + **POST** {{baseUrl}}/api/employees
  + Body:

{

"person\_id": "E001",

"first\_name": "John",

"last\_name": "Doe",

"department": "Assembly",

"status": "פעיל",

"role": "Employee"

}

* + **Expect 201** with saved doc.
  + **Negative**: missing required (implicit) → 500 with message from Mongoose if schema enforces; otherwise will create as-is.
* **Update employee (by person\_id)**
  + **PUT** {{baseUrl}}/api/employees/E001
  + Body (partial): { "department": "QA", "status": "לא פעיל" }
  + **Expect 200** updated employee
  + **Negative**: not found → 404

**Stations, Products, Workstations**

* **Get stations**
  + **GET** {{baseUrl}}/api/stations
  + **Expect 200 :**

[{ "\_id": "mongoId", "station\_id": "S001", "station\_name": "Press 1", "department": "Assembly", "product\_name": "Valve A" }]

* **Get products**
  + **GET** {{baseUrl}}/api/products
  + **Expect 200** (name-only selection)

[{ "\_id": "mongoId", "product\_name": "Valve A" }]

* **Get workstations for a station**
  + **GET** {{baseUrl}}/api/workstations/Press%201
  + **Expect 200:**

[{ "\_id": "mongoId", "station\_name": "Press 1", "workingStation\_name": "Press1-WS-01" }]

* + **Negative**: stationName not found → 404 "Station not found"

**Qualifications**

* **Get all**
  + **GET** {{baseUrl}}/api/qualifications
  + **Expect 200:**

[{ "\_id": "mongoId", "person\_id": "E001", "station\_name": "Press 1", "avg": 92 }]

* **Create**
  + **POST** {{baseUrl}}/api/qualifications
  + Body:

{ "person\_id": "E001", "station\_name": "Press 1", "avg": 88 }

* + **Expect 201** created doc
* **Upsert (update or create)**
  + **PUT** {{baseUrl}}/api/qualifications
  + Body: { "person\_id": "E001", "station\_name": "Press 1", "avg": 95 }
  + **Expect 200** updated doc
  + **Negative**: unknown person\_id (no matching Person) → 404 "Person not found"
* **Get by employee (by person\_id)**
  + **GET** {{baseUrl}}/api/qualifications/E001
  + **Expect 200** list of that employee’s qualifications

**Matching & Selection (Genetic / Ranking)**

These depend on Persons, Stations, Qualifications to be present.

* **Top employees for a station**
  + **GET** {{baseUrl}}/api/top-employees/Press%201/3
  + **Expect 200** array of top N:

**[ { "person\_id": "E001", "score": 0.91, "first\_name": "John", "last\_name": "Doe" }]**

(**shape depends on your getTopEmployeesForStation return; verify actual keys**)

* + **Negative**: station not found (if Station.findOne returns null) → likely 500 from downstream unless you add a 404; good to test behavior.
* **Sorted employees for a station**
  + **GET** {{baseUrl}}/api/sorted-employees/Press%201
  + **Expect 200** array sorted by fitness/avg

[

{ "person\_id": "E001", "score": 0.91 },

{ "person\_id": "E002", "score": 0.84 }

]

* + **Negative**: station not found → 404 "Station not found"
* **Employees with qualifications for a station**
  + **GET** {{baseUrl}}/api/employees-with-qualifications/Press%201
  + **Expect 200**

[

{

"\_id": "mongoPersonId",

"person\_id": "E001",

"first\_name": "John",

"last\_name": "Doe",

"department": "Assembly",

"role": "Employee",

"qualification\_avg": 92

}

]

* **Assign employees (geneticAlgorithm)**
  + **POST** {{baseUrl}}/api/assign-employees
  + Body (note the IDs expected):

{

"selectedStations": ["<Station \_id>", "<Station \_id>"],

"selectedEmployees": ["<Person \_id>", "<Person \_id>"]

}

* + **Expect 200** mapping by employeeId:

{

"<Person \_id>": {

"stationId": "S001", // from station.station\_id

"stationName": "Press 1",

"qualificationScore": 95

}

}

* + **Negative:**
    - Missing arrays or empty arrays → likely 500; verify behavior
    - If input uses person\_id instead of \_id, it won’t match—use \_id as per code.
  + ⚠️ Note: Endpoint fetches Person and Station by **\_id**, then later accesses station.station\_id and seeks Qualification by person\_id—ensure your test data lines up (Qualification.person\_id should match the *string id* you pass in selectedEmployees if your geneticAlgorithm expects that).

**Assignments**

* **Get assignments for a date**
  + **GET** {{baseUrl}}/api/assignments?date=2025-08-10
  + **Expect 200**

[{

"\_id": "mongoId",

"assignment\_id": "string",

"date": "2025-08-10T08:00:00.000Z",

"number\_of\_hours": 8,

"workingStation\_name": "Press1-WS-01",

"person\_id": "E001"

}]

* + **Negative**: missing date → 400 "Date parameter is required"
* **Create assignment**
  + **POST** {{baseUrl}}/api/assignments
  + Body:

{

"date": "2025-08-10",

"workingStation\_name": "Press1-WS-01",

"person\_id": "E001",

"number\_of\_hours": 8

}

* + **Expect 201** saved doc
  + **Negative**: any missing field → 400 "All fields are required"
* **Delete assignment (by ordinal for given date/person)**
  + **DELETE** {{baseUrl}}/api/assignments
  + Body:

{

"date": "2025-08-10",

"person\_id": "E001",

"assignmentNumber": 1

}

* + **Expect 200** { "message": "Assignment deleted successfully" }
  + **Negative**
    - Missing fields → 400 (with required array)
    - No assignments for date/person → 404
    - assignmentNumber out of range → 404 "Specified assignment not found"

**Dashboard**

* **Dashboard data (today)**
  + **GET** {{baseUrl}}/api/dashboard-data
  + **Expect 200**

{

  "inactiveWorkers": 2,

  "activeWorkers": 8,

  "dailyDefects": 0,

  "inactiveStations": 5

}

* **Shluker results (today, from mqttMsg collection)**
  + **GET** {{baseUrl}}/api/shluker-results
  + **Expect 200**

{ "proper": 120, "improper": 7 }

**Reports**

**One entry point:  GET /api/report with  optional query params.**

* **Monthly production (default if no params)**
  + **GET** {{baseUrl}}/api/report
  + **Expect 200** array per day:

[

{ "\_id": "2025-08-01", "goodValves": 210, "invalidValves": 8 },

{ "\_id": "2025-08-02", "goodValves": 198, "invalidValves": 11 }

]

* **Monthly report for a specific employee**
  + **GET** {{baseUrl}}/api/report?employee=E001
  + **Expect 200** same structure but filtered for that employee’s messages.
* **Daily report (optionally filter by employee)**
  + **GET** {{baseUrl}}/api/report?date=2025-08-10
  + **Expect 200**

{ "goodValves": 12, "invalidValves": 1 }

* + **GET** {{baseUrl}}/api/report?date=2025-08-10&employee=E001
  + **Expect 200** same shape filtered by User ID.
  + **Negative**
    - Bad date format → likely 200 with zeros or internal try/catch logs (no explicit 400 in code).
    - Empty data set → valid 200 with zeros/empty arrays.