**Diaz – Documentation Technical Test**

Table of Contents

[Introduction 1](#_Toc208908959)

[Environment Setup 1](#_Toc208908960)

[Point 1: Backend Development 1](#_Toc208908961)

[Point 2: Automation Testing 3](#_Toc208908962)

[Point 3: Data Processing 6](#_Toc208908963)

[Attachments 8](#_Toc208908964)

## Introduction

This project is a submission for a technical test that demonstrates skills in backend development, automation scripting, and data processing. It includes three core tasks:

1. Building a simple backend server using Node.js/Express to manage data via a REST API.
2. Creating automation scripts using a cron job for periodic data collection and cleansing.
3. Writing a series of SQL queries to manipulate and retrieve data from a table.

## Environment Setup

To run the project, please ensure that the following dependencies are installed in your environment. You can use Node.js & npm, a Linux-based environment

**Requirements**

* Node.js and npm.
* Postman.
* Dependencies (npm install).
* Linux based environment.
* SQL Client (MySQL Workbench).
* Database Server : MySQL Server or other compatible database.

# Point 1: Backend Development

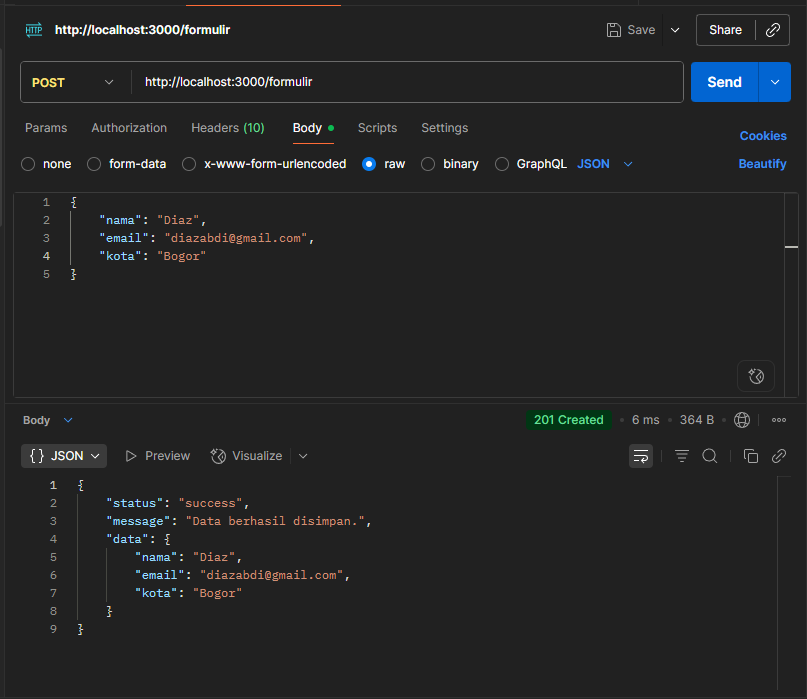
**Description**

The first task is a small backend server built with Node.js and Express. It's designed to handle form submissions. The script receives data from a form, saves it, and can send back all the data it has stored.

**How to Run**

1. Run npm install to install all the necessary libraries and packages listed in the package.json file, such as Express
2. Run the server using node server.js , if the server starts successfully, you will see a confirmation message in your terminal: Server berjalan di <http://localhost:3000>
3. Testing the API endpoint using a tool like Postman

* POST /formulir : To submit new form data.



* GET /formulir : To retrieve all submitted data.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Code Explanation**

**A screenshot of a computer program

AI-generated content may be incorrect.**

* The app.post endpoint is for receiving data from the frontend's request body and storing it in a simple array.
* The app.get endpoint retrieves all the stored data and sends it back as a response.

**Output**

A computer screen shot of a computer code

AI-generated content may be incorrect.

# Point 2: Automation Testing

**Description**

This project contains two shell scripts to automate the process of periodic data collection and the automatic cleanup of old data using cron in a Linus environment.

* **collect\_data.sh**: A script to collect data and save it to a new .csv file in /home/cron. The filename is dynamically generated based on the execution date and time.

A computer screen shot of a code

AI-generated content may be incorrect.

* **cleanup\_data.sh**: A script to check the /home/cron directory and automatically delete .csv files older than 30 days.

A screen shot of a computer code

AI-generated content may be incorrect.

**Configuration**

1. Copy the scripts to your home directory(~/)
2. Create the directory for storing the CSV data files and set the correct permissions

sudo mkdir -p /home/cron

sudo chown $(whoami):$(whoami) /home/cron

1. Make the scripts executable

chmod +x ~/collect\_data.sh

chmod +x ~/cleanup\_data.sh

1. Set up the cron schedule

* Open crontab editor: crontab -e
* Add the following lines at the end of the file, and replace YOUR\_USERNAME with your actual username.

# Run the data collection script 3 times a day (08:00, 12:00, 15:00)

0 8 \* \* \* /home/YOUR\_USERNAME/collect\_data.sh

0 12 \* \* \* /home/YOUR\_USERNAME/collect\_data.sh

0 15 \* \* \* /home/YOUR\_USERNAME/collect\_data.sh

# Run the cleanup script every day at 03:05 AM

5 3 \* \* \* /home/YOUR\_USERNAME/cleanup\_data.sh

* Save it and close the editor

**How to Run**

No manual run command. Cron will automatically run the scripts according to schedule

* **Collect\_data.sh** will run at 8 AM, 12 PM, and 3 PM every day.
* **Cleanup\_data.sh** will run at 03:05 AM everyday (if the computer is on).

**Verification**

To verify that the system is working correctly:

* Check the contents of the /home/cron directory after an execution time. A new CSV file should appear.

ls -lh /home/cron

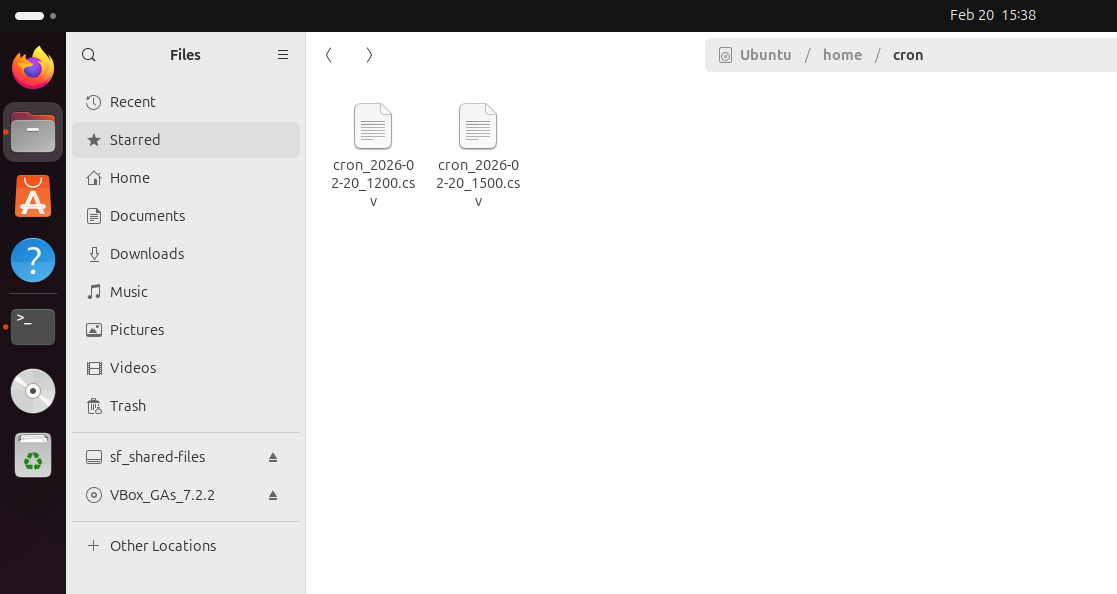
* View the system log to confirm that cron has executed your commands

sudo grep CRON /var/log/syslog

**Output**

**A screen shot of a computer screen

AI-generated content may be incorrect.**

****

# Point 3: Data Processing

**Description**

This project contains a set of SQL Scripts to create, manipulate, and query a simple employee data table. It serves CREATE, INSERT, UPDATE, SELECT operations.

**Configuration**

The SQL\_Script.sql is designed to handle most of the configuration automatically. The only manual setup step is to ensure your database server is running.

The script will automatically perform the following actions:

1. Create a new database
2. Switch to use the database
3. Create the employee table with the correct schema

**How to Run**

1. Open your SQL Client: Launch MySQL Workbench or preferred SQL Client.
2. Open the Script File: Open the SQL\_Script.sql file in your client’s query editor.
3. Execute the Entire Script

**CODE**

A computer screen shot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**OUTPUT**

* 1 and 2

A screenshot of a computer

AI-generated content may be incorrect.

* Task 3

**A close up of a sign

AI-generated content may be incorrect.**

* Task 4

**A screenshot of a computer

AI-generated content may be incorrect.**

* Task 5

**A screenshot of a computer

AI-generated content may be incorrect.**

## Attachments

* Server.js : It is the main script that Node.js executes to create, configure, and run your entire web server.
* Collect\_data.sh & cleanup\_data.sh : The script for automation testing
* Cron\_yyyy-mm-dd\_hhmm.csv : automation testing output file
* SQL\_Scripts.sql : The script for Data Processing