RANCANG BANGUN CHATBOT UNTUK AKUISISI DATA PADA GREENHOUSE DENGAN MENGGUNAKAN TELEGRAM DAN GOOGLE DRIVE API DI PT PLN NUSANTARA POWER UP REMBANG

Naufal Yafi Susanto[1)], Budi Setiyono, S.T., M.T.
Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro
Jl. Prof. Soedharto, S.H., Kampus UNDIP Tembalang, Semarang 50275, Indonesia
[1)]Email : naufalyafisusanto@students.undip.ac.id

Abstrak

Sebagai salah satu perusahaan yang peduli terhadap lingkungan, PT PLN Nusantara Power UP Rembang selalu berupaya untuk menggunakan energi baru terbarukan. Dalam penggunaan energi ini, perusahaan perlu melakukan akuisisi data secara berkala agar dapat dikaji lebih lanjut. Akan tetapi, proses akuisisi data tersebut masih dilakukan secara manual yaitu dengan menggunakan aplikasi kendali jarak jauh (TeamViewer). Cara ini tidak efektif mengingat berkas hasil akuisisi data tidak tersusun dengan rapi. Selain itu, koneksi internet yang tidak baik juga akan memperlambat dalam proses kendali jarak jauh tersebut. Dengan demikian, dibuatlah sebuah *chatbot* Telegram yang terintegrasi dengan Google Drive API untuk memudahkan proses akuisisi data tersebut. *Chatbot* akan mengirimkan berkas hasil akuisisi data berdasarkan rentang tanggal yang diinginkan. Selain itu, *chatbot* juga dibekali dengan *command* untuk menampilkan data yang terakhir disimpan secara *realtime*. Hal tersebut bertujuan untuk memudahkan dalam pemantauan akuisisi data tersebut. *Chatbot* tersedia di dalam aplikasi Telegram pada *smartphone* sehingga dapat diakses dari mana pun dan kapan pun tanpa harus menggunakan.

Kata kunci : akuisisi data, *chatbot*, Telegram, Google Drive API

Abstract

As a company that cares about the environment, PT PLN Nusantara Power UP Rembang always strives to use new renewable energy. In using this energy, the company needs to do data acquisition periodically so that it can be studied further. However, the data acquisition process is still done manually using a remote control application (TeamViewer). This method is not effective considering that the files of data acquisition results are not neatly arranged. In addition, a poor internet connection will also slow down the remote control process. Thus, a Telegram chatbot integrated with Google Drive API was created to facilitate the data acquisition process. Chatbot will send files of data acquisition results based on the desired date range. In addition, chatbot is also equipped with a command to display the last saved data in realtime. This aims to facilitate the monitoring of data acquisition. This chatbot is available in the Telegram application on smartphones so that it can be accessed from anywhere and anytime.

***Keywords** : data acquisition, chatbot,* Telegram, Google Drive API

1. Pendahuluan

Kemajuan teknologi di Indonesia menyebabkan kebutuhan akan energi listrik terus meningkat. Oleh karena itu, perlu adanya pembangkit listrik untuk memenuhi kebutuhan tersebut. PLTU merupakan jenis pembangkit yang paling banyak di Indonesia, salah satunya yaitu PLTU Rembang. PLTU berbahan bakar batu bara ini dikelola dan dimiliki oleh PT PLN Nusantara Power dengan daya sebesar 2×315 MW [1].

Pembangkit listrik tersebut tentunya tidak lepas dari berbagai macam polusi yang dihasilkan. Oleh karena itu, PT PLN Nusantara Power UP Rembang berkomitmen untuk menjaga lingkungan dengan menggunakan Energi Baru Terbarukan (EBT). Komitmen yang dilakukan yaitu dengan menggunakan energi dari PV (*Photovoltaic*) dan VAWT (*Vertical Axis Wind Turbine*) pada sebuah *greenhouse* yang dimiliki oleh perusahaan ini. Energi yang dihasilkan nantinya akan diakuisisi datanya untuk dikaji lebih lanjut.

Proses akuisisi data tersebut masih dilakukan dengan cara manual yaitu dengan menggunakan aplikasi kendali jarak jauh. Cara ini tidak efektif karena pengguna harus mencari berkas data yang tidak tersusun rapi dengan koneksi internet yang tidak stabil. Oleh karena itu, dibuatlah sebuah *chatbot* Telegram yang terintegrasi dengan Google Drive API untuk memudahkan proses akuisisi data. *Chatbot* ini dapat diakses menggunakan aplikasi Telegram pada *smartphone* sehingga pengguna dapat mengakses dari mana pun dan kapan pun.

2. Dasar Teori

2.1 Akuisisi Data

Akuisisi data merupakan suatu sistem yang digunakan untuk mengambil, mengumpulkan dan menyiapkan data, kemudian data tersebut diolah lebih lanjut untuk keperluan tertentu [2]. Akuisisi data terdiri dari beberapa bagian, yaitu unit pemrosesan sinyal, sensor, perangkat keras, serta unit komputer. Data hasil pembacaan sensor tersebut nantinya akan dikirim ke komputer secara seri atau paralel agar dapat disimpan dan dikaji lebih lanjut.

2.2 Chatbot

Chatbot adalah sebuah program yang dijalankan pada sebuah komputer tertentu dengan maksud untuk menangani kegiatan percakapan kepada manusia secara otomatis. *Chatbot* hanya bertindak sesuai dengan algoritma yang dijalankan, sehingga *chatbot* tidak memiliki emosional seperti halnya manusia. Hal tersebut membuat *chatbot* hanya mengambil tindakan atau keputusan dalam sebuah percakapan dengan manusia tanpa diikuti dengan perasaan. [3].

2.3 Telegram

Telegram merupakan salah satu aplikasi layanan berkirim pesan yang berbasis *cloud* dan tentunya tidak berbayar. Pengguna dapat mengirimkan pesan, foto, video, suara, atau berkas apa saja dengan menggunakan Telegram [4]. Selain itu, Telegram juga menyediakan bot API yang memungkinkan sebuah program aplikasi untuk menggunakan pesan Telegram sebagai *interface*.

2.4 Google Drive API

Google Drive API merupakan salah satu antarmuka pemrograman aplikasi yang disediakan oleh Google untuk memudahkan web atau aplikasi dalam mengakses penyimpanan *cloud* Google Drive [5]. Arsitektur yang digunakan pada antarmuka pemrograman aplikasi ini yaitu REST API. Tidak semua akses diberikan langsung oleh Google Drive API. Akan tetapi, akses tersebut diberikan berdasarkan *scopes* yang ada pada *access token* yang menggunakan protokol OAuth 2.0.

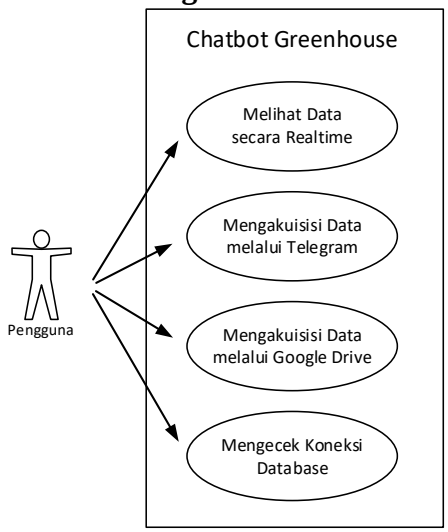
3. Rancang Bangun Chatbot

3.1 Analisis Kebutuhan

Data yang akan diakuisisi ini tersimpan di dalam *database* lokal. Struktur *database* ini terdiri dari kolom *timestamp* yang diikuti dengan kolom-kolom lainnya. Rentang data yang ingin diakuisisi yaitu berdasarkan tanggal awal dan akhir yang nantinya akan dikirimkan oleh pengguna *chatbot*. Selain itu, pengguna juga membutuhkan layanan untuk memantau koneksi *database* serta data yang terakhir disimpan secara *realtime* untuk memastikan bahwa proses akuisisi data berjalan dengan lancar.

3.2 Perancangan

3.2.1 Use Case Diagram



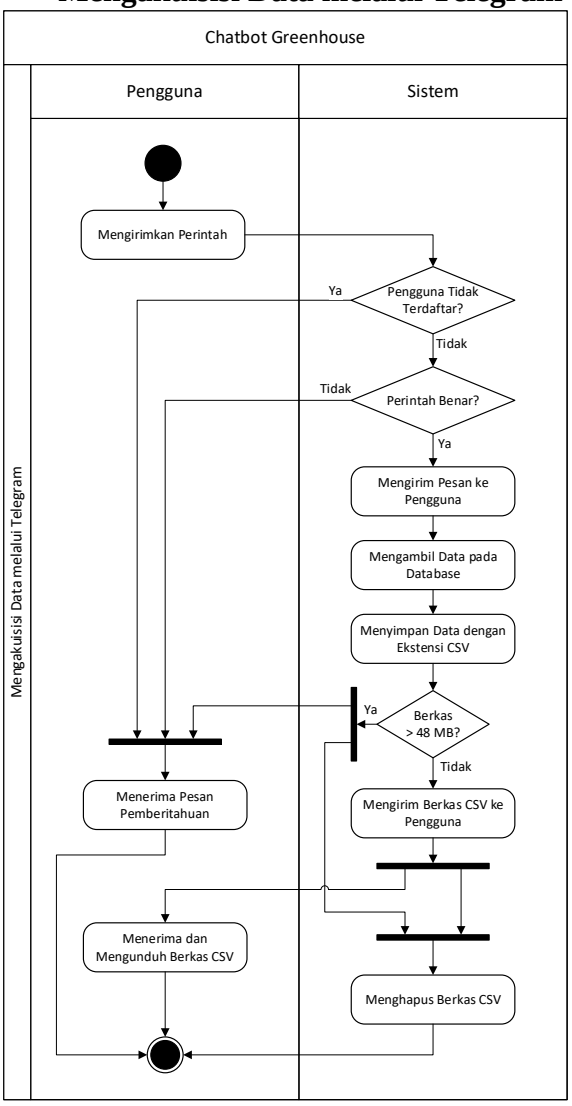
Gambar 1 *Use Case Diagram Chatbot Greenhouse*

Terlihat pada Gambar 4.1, hanya terdapat satu aktor yaitu pengguna. Pengguna dapat melihat data pembacaan sensor secara *realtime*, mengakuisisi data melalui Telegram secara langsung, mengakuisisi data melalui tautan Google Drive, serta mengecek koneksi *database* yang ada.

3.2.2 Activity Diagram

Pada tahap ini dilakukan pembuatan *activity diagram* untuk menggambarkan alur saat pengguna menggunakan *chatbot* yang ada pada *use case diagram* di atas sebagai berikut.

- **Mengakuisisi Data melalui Telegram**



Gambar 2 *Activity Diagram* Mengakuisisi Data melalui Telegram

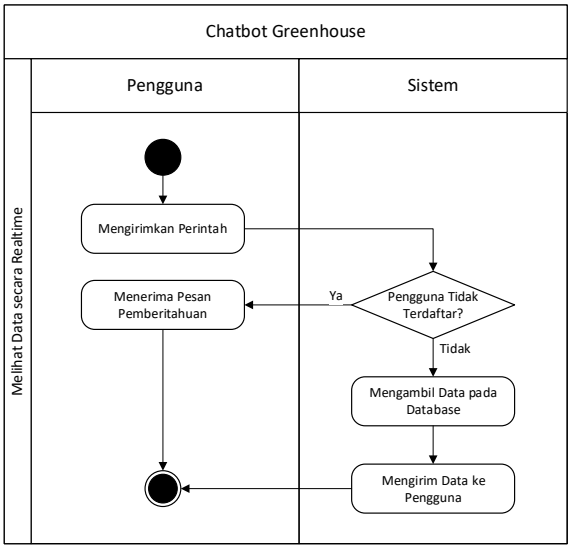
Terdapat dua *swimlane* yaitu pengguna dan sistem. Pertama, pengguna mengirimkan *command* `/get_csv YYYY-MM-DD YYYY-MM-DD` ke *chatbot* dengan YYYY-MM-DD merupakan format tanggal. Sistem akan melakukan verifikasi terlebih dahulu pada *userid* pengguna. Jika tidak terdaftar, maka pengguna akan mendapatkan pesan pemberitahuan bahwa *userid* pengguna tidak terdaftar. Jika pengguna terdaftar, maka sistem akan melakukan pengecekan format tanggal.

Jika format tanggal salah, maka pengguna akan mendapatkan pesan pemberitahuan. Jika

tidak, maka sistem akan mengirimkan pesan untuk mengonfirmasi *command* yang telah dikirim. Setelah itu, sistem akan mengambil data pada *database* sesuai rentang tanggal yang dimasukkan lalu menyimpannya ke dalam ekstensi CSV di penyimpanan lokal sistem.

Jika ukuran berkas CSV melebihi 48 MB, maka pengguna akan disarankan untuk menggunakan tautan Google Drive. Jika tidak, maka sistem akan mengirimkan berkas tersebut langsung melalui Telegram. Pengguna akan menerima berkas CSV yang dapat diunduh. Sistem akan menghapus berkas tersebut dari penyimpanan lokal sistem untuk menghemat ruang penyimpanan.

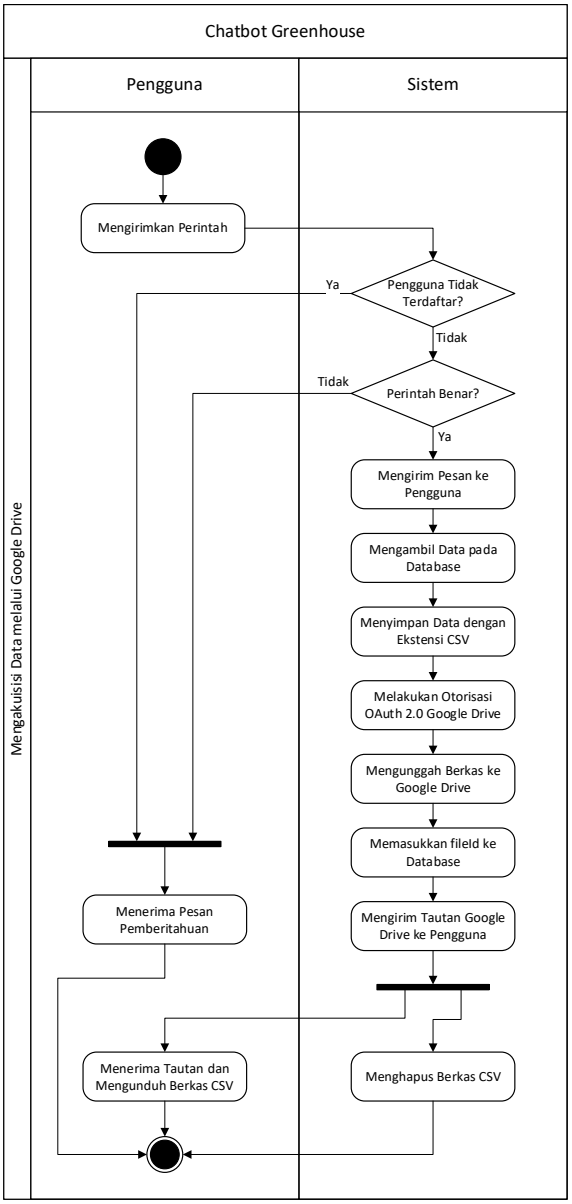
- **Melihat Data secara *Realtime***



Gambar 3 *Activity Diagram* Melihat Data secara *Realtime*

Terdapat dua *swimlane* yaitu pengguna dan sistem. Pertama, pengguna mengirimkan *command* `/show_data` ke *chatbot*. Sistem akan melakukan verifikasi terlebih dahulu pada *userid* pengguna. Jika tidak terdaftar, maka pengguna akan mendapatkan pesan pemberitahuan bahwa *userid* pengguna tidak terdaftar. Jika pengguna terdaftar, maka sistem akan mengambil data pada *database* lalu mengirimkannya ke pengguna.

- **Mengakuisisi Data melalui Google Drive**



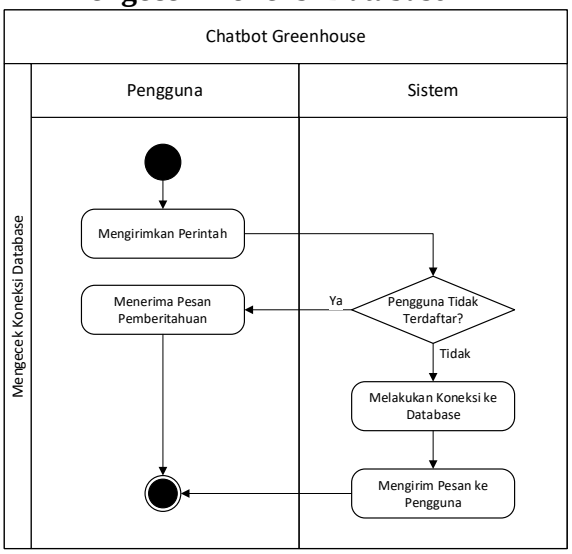
Gambar 4 *Activity Diagram* Mengakuisisi Data melalui Google Drive

Terdapat dua *swimlane* yaitu pengguna dan sistem. Pertama, pengguna mengirimkan *command* `/get_drive YYYY-MM-DD YYYY-MM-DD` ke *chatbot*. Sistem akan melakukan verifikasi pengguna. Jika tidak terdaftar, maka pengguna akan mendapatkan pesan pemberitahuan. Jika pengguna terdaftar, maka sistem akan melakukan pengecekan format tanggal.

Jika format tanggal salah, maka pengguna akan mendapatkan pesan pemberitahuan bahwa tanggal yang dimasukkan tidak benar. Jika tidak, maka sistem akan mengirimkan pesan untuk mengonfirmasi *command* yang telah dikirimkan. Setelah itu, sistem akan mengambil data pada *database* sesuai rentang tanggal yang dimasukkan lalu menyimpannya ke dalam ekstensi CSV di penyimpanan lokal sistem.

Sistem akan melakukan otorisasi OAuth 2.0 untuk dapat mengakses API. Selanjutnya sistem mengunggah berkas CSV ke Google Drive. Setelah proses pengunggahan selesai, sistem akan menyimpan fileId ke *database* serta mengirim tautan Google Drive ke pengguna. Pengguna akan menerima tautan yang dapat diunduh dengan masa aktif satu minggu sebelum berkas dihapus oleh sistem dari Google Drive. Selanjutnya, sistem akan menghapus berkas tersebut dari penyimpanan lokal sistem untuk menghemat penyimpanan.

- **Mengecek Koneksi *Database***

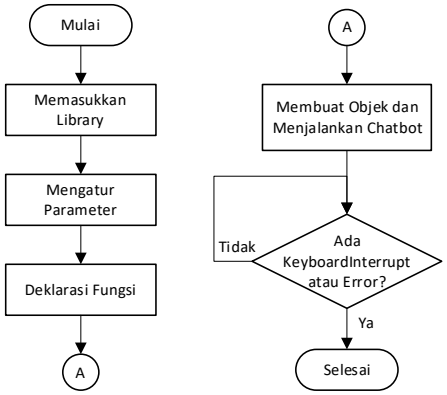


Gambar 5 *Activity Diagram* Mengecek Koneksi *Database*

Terdapat dua *swimlane* yaitu pengguna dan sistem. Pertama, pengguna mengirimkan *command* `/check_db` ke *chatbot*. Sistem akan melakukan verifikasi *userid* pengguna. Jika tidak terdaftar, maka pengguna akan mendapatkan pesan pemberitahuan. Jika pengguna terdaftar, maka sistem akan melakukan koneksi ke *database* lalu mengirimkan status koneksi ke pengguna.

3.3 Pengkodean

3.3.1 Flowchart Program



Gambar 6 *Flowchart* Program

Alur program dimulai dengan memasukkan beberapa *library* yang dibutuhkan serta mengatur beberapa parameter yang digunakan. Selanjutnya, program melakukan deklarasi fungsi yang bertujuan untuk merespons terhadap setiap *command* yang dikirimkan. Terdapat fungsi lainnya yang dideklarasikan untuk menunjang fungsi utama. Setelah itu, program akan membuat objek dari sebuah *class* yang akan menjalankan *chatbot* secara terus menerus hingga terdapat KeyboardInterrupt atau *error* yang muncul.

3.3.2 Memasukkan Library

```
import mysql.connector as sql, os, csv, json, traceback, logging
from telegram.ext import *
from datetime import datetime, timedelta
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload
from google.oauth2.credentials import Credentials
```

Gambar 7 Program untuk Memasukkan *Library*

Library mysql.connector untuk mengoneksikan ke *database*. Selain itu terdapat *library* os yang digunakan untuk mengelola berkas sistem, csv untuk menangani berkas CSV, serta json untuk menangani berkas JSON. Terdapat juga *library* traceback dan logging yang berfungsi untuk mendeteksi *error* serta menyimpannya ke dalam suatu berkas. *Library*

datetime digunakan untuk mengakses dan memanipulasi tanggal dan waktu. *Library* telegram.ext digunakan untuk menjalankan *chatbot* Telegram. Selain itu, juga terdapat beberapa *library* yang digunakan untuk mengakses Google Drive API.

3.3.3 Mengatur Parameter

```python
#Logger
logging.basicConfig(filename=r"log/bot.log",
                    format="%(asctime)s in %(funcName)s\n  %(message)s")

#Google Drive
scopes = ["https://www.googleapis.com/auth/drive"]
path_token = r"src/token.json"
folder = "[redacted]"

#Telegram
userid = ["[redacted]"]
token = "[redacted]"

#Database
host = "localhost"
user = "greenhouse"
password = "greenhouse"
database = "greenhouse"
```

Gambar 8 Program untuk Mengatur Parameter

Parameter yang perlu diatur yaitu folder, nama, serta format penulisan berkas *logger* yang digunakan. Selain itu, terdapat beberapa parameter Google Drive API yaitu *scopes*, *path_token*, serta folder yang digunakan untuk mengunggah berkas. Selanjutnya, parameter utama yaitu untuk *chatbot* Telegram terdiri dari *userid* serta *token*. Parameter terakhir yaitu parameter yang diperlukan untuk melakukan koneksi ke *database* yaitu *host*, *user*, *password*, dan *database* yang digunakan.

3.3.4 Deklarasi Fungsi

- **verify**

Proses verifikasi dilakukan dengan menggunakan *userid* yang dimiliki pengguna. Berkas `userid.json` dimuat ke dalam parameter `userid`. Jika pengguna terdaftar, maka fungsi akan mengembalikan nilai `False`. Jika tidak maka `True`.

```python
#Verifikasi User
def verify(user):
    global userid
    #Memuat Data User dari JSON
    with open(r"src/userid.json", "r") as file:
        try:
            userid = json.load(file)
        except:
            #Error
            logging.warning(traceback.format_exc().replace("\n", "\n  "))
    if str(user) in userid:
        return False
    else:
        return True
```

Gambar 9 Fungsi verify

- **show_data**

Sistem akan melakukan verifikasi pada *userid* pengguna. Jika tidak terdaftar, maka pengguna akan mendapatkan pemberitahuan. Jika terdaftar, maka sistem akan mengambil data terakhir pada *database* lalu menyimpannya ke parameter `sensor` dalam bentuk *tuple*.

```python
#Melihat Data Secara Realtime
async def show_data(update, context):
    #Verifikasi User
    if verify(update.message.from_user.id):
        await update.message.reply_text(
            "Mohon maaf, User ID %s milik Anda tidak terdaftar dalam sistem kami."\
            % update.message.from_user.id)
        return None
    try:
        #Koneksi ke Database
        db = sql.connect(
            host = host,
            user = user,
            password = password,
            database = database)
        cursor = db.cursor()
        #Mengeksekusi Query
        cursor.execute("SELECT * FROM monitoring_daya ORDER BY timestamp DESC LIMIT 1")
        sensor = tuple(cursor.fetchall()[0])[1:]
        db.close()
```

Gambar 10 Fungsi show_data Bagian 1

Selanjutnya, sistem akan mengirimkan pesan yang berisi hasil akuisisi data dalam bentuk teks. Apabila terdapat *error* saat menjalankan fungsi ini, pengguna akan mendapatkan pemberitahuan.

```python
        #Mengirim Pesan ke User
        await update.message.reply_text(
            "Monitoring Daya Greenhouse\n"\
            " - Timestamp : %s\n"\
            " - Voltage PV : %s Volt\n"\
            " - Current PV : %s Ampere\n"\
            " - Power PV : %s Watt\n"\
            " - Voltage VAWT : %s Volt\n"\
            " - Current VAWT : %s Ampere\n"\
            " - Power VAWT : %s Watt\n"\
            " - Anemometer : %s m/s\n"\
            " - Voltage Battery : %s Volt"\
            % sensor)
    except Exception as error:
        #Error
        logging.warning(traceback.format_exc().replace("\n", "\n  "))
        await update.message.reply_text(
            "Mohon maaf, terjadi kesalahan sistem saat sedang memproses data. "\
            "Silahkan coba beberapa saat lagi.\nError : %s" % error)
```

Gambar 11 Fungsi show_data Bagian 2

- **get_csv**

Sistem akan melakukan verifikasi pada *userid* pengguna. Jika tidak terdaftar, maka pengguna akan mendapatkan pemberitahuan bahwa *userid* tidak terdaftar. Jika pengguna terdaftar, maka sistem memeriksa format tanggal yang dikirimkan. Jika salah, maka pengguna akan mendapatkan pemberitahuan bahwa format tanggal yang dimasukkan salah atau tidak benar.

```
#Mengakuisisi Data CSV Melalui Telegram
async def get_csv(update, context):
    #Verifikasi User
    if verify(update.message.from_user.id):
        await update.message.reply_text(
            "Mohon maaf, User ID %s milik Anda tidak terdaftar dalam sistem kami."\
            % update.message.from_user.id)
        return None
    try:
        #Memeriksa Format Perintah
        date = str(update.message.text).split("/get_csv ")[1]
        date_start = date.split(" ")[0]
        date_end = date.split(" ")[1]
        datetime.strptime(date_start, "%Y-%m-%d")
        datetime.strptime(date_end, "%Y-%m-%d")
    except:
        #Format Tidak Benar
        await update.message.reply_text(
            "Perintah yang dimasukkan tidak benar\nFormat :\n/get_csv YYYY-MM-DD YYYY-MM-DD\n"\
            "Contoh :\n/get_csv 2023-01-01 2023-01-20")
        return None
```

Gambar 12 Fungsi get_csv Bagian 1

Jika format tanggal benar, maka sistem akan mengirimkan pesan konfirmasi lalu mengambil data pada *database* berdasarkan rentang tanggal yang dimasukkan. Data tersebut lalu disimpan di dalam parameter `data`.

```
    try:
        await update.message.reply_text("Mohon menunggu, sistem sedang memproses data.")
        #Koneksi ke Database
        db = sql.connect(
            host = host,
            user = user,
            password = password,
            database = database)
        cursor = db.cursor()
        #Mengeksekusi Query
        cursor.execute(
            "SELECT timestamp, v_pv, i_pv, p_pv, v_vawt, i_vawt, p_vawt, anemo, v_bat "\
            "FROM monitoring_daya WHERE timestamp BETWEEN '%s 00:00:00' AND '%s 00:00:00' "\
            "ORDER BY id ASC" % (date_start, date_end))
        data = list(cursor.fetchall())
        db.close()
```

Gambar 13 Fungsi get_csv Bagian 2

Setelah itu, data akan disimpan ke dalam CSV. Jika ukuran berkas tersebut melebihi 48 MB, maka pengguna disarankan untuk menggunakan *command* `/get_drive`. Jika tidak, maka sistem akan mengirimkan berkas tersebut. Apabila terdapat *error*, pengguna akan mendapatkan pemberitahuan. Setelah semuanya selesai, berkas akan dihapus dari penyimpanan lokal untuk menghemat ruang.

```
        #Menyimpan Data ke CSV
        column = ["Timestamp", "PV Voltage (Volt)", "PV Current (Ampere)", "PV Power (Watt)",\
                  "VAWT Voltage (Volt)", "VAWT Current (Ampere)", "VAWT Power (Watt)",\
                  "Anemometer (m/s)", "Battery Voltage (Volt)"]
        with open(r"cache_csv/Data Greenhouse %s to %s.csv" % (date_start, date_end),
                  mode="w", newline="") as file:
            writer = csv.writer(file)
            writer.writerow(column)
            writer.writerows(data)
            path = os.path.abspath(file.name)
        #Memeriksa Ukuran Berkas
        if os.path.getsize(path) >= 48000000:
            await update.message.reply_text("Mohon maaf, berkas csv melebihi 50 MB. "\
                                            "Silahkan menggunakan perintah /get_drive.")
        else:
            with open(path, "rb") as csv_file:
                await update.message.reply_document(
                    csv_file, read_timeout=120, write_timeout=120, connect_timeout=120)
    #Error
    except Exception as error:
        logging.warning(traceback.format_exc().replace("\n", "\n  "))
        await update.message.reply_text(
            "Mohon maaf, terjadi kesalahan sistem saat sedang memproses data. "\
            "Silahkan coba beberapa saat lagi.\nError : %s" % error)
    #Menghapus Berkas Cache
    try: os.remove(path)
    except: pass
```

Gambar 14 Fungsi get_csv Bagian 3

- **get_drive**

Sistem akan melakukan verifikasi pada *userid* pengguna. Jika tidak terdaftar, maka pengguna akan mendapatkan pemberitahuan bahwa *userid* tidak terdaftar. Jika pengguna terdaftar, maka sistem memeriksa format tanggal yang dikirimkan. Jika salah, maka pengguna akan mendapatkan pemberitahuan bahwa format tanggal yang dimasukkan salah atau tidak benar. Jika format tanggal benar, maka sistem akan mengirimkan pesan konfirmasi lalu mengambil data pada *database* berdasarkan rentang tanggal yang dimasukkan. Data tersebut lalu disimpan di dalam parameter `data`. Setelah itu, sistem akan menyimpan data tersebut ke dalam ekstensi CSV di penyimpanan lokal.

```
#Mengakuisisi Data CSV Melalui Google Drive
async def get_drive(update, context):
    #Verifikasi User
    if verify(update.message.from_user.id):
        await update.message.reply_text(
            "Mohon maaf, User ID %s milik Anda tidak terdaftar dalam sistem kami."\
            % update.message.from_user.id)
        return None
    try:
        #Memeriksa Format Perintah
        date = str(update.message.text).split("/get_drive ")[1]
        date_start = date.split(" ")[0]
        date_end = date.split(" ")[1]
        datetime.strptime(date_start, "%Y-%m-%d")
        datetime.strptime(date_end, "%Y-%m-%d")
    except:
        #Format Tidak Benar
        await update.message.reply_text(
            "Perintah yang dimasukkan tidak benar\nFormat :\n/get_drive YYYY-MM-DD YYYY-MM-DD\n"\
            "Contoh :\n/get_drive 2023-01-01 2023-01-20")
        return None
    try:
        await update.message.reply_text("Mohon menunggu, sistem sedang memproses data.")
        #Koneksi ke Database
        db = sql.connect(
            host = host,
            user = user,
            password = password,
            database = database)
        cursor = db.cursor()
        #Mengeksekusi Query
        cursor.execute(
            "SELECT timestamp, v_pv, i_pv, p_pv, v_vawt, i_vawt, p_vawt, anemo, v_bat "\
            "FROM monitoring_daya WHERE timestamp BETWEEN '%s 00:00:00' AND '%s 00:00:00' "\
            "ORDER BY id ASC" % (date_start, date_end))
        data = list(cursor.fetchall())
        #Menyimpan Data ke CSV
        column = ["Timestamp", "PV Voltage (Volt)", "PV Current (Ampere)", "PV Power (Watt)",\
                  "VAWT Voltage (Volt)", "VAWT Current (Ampere)", "VAWT Power (Watt)",\
                  "Anemometer (m/s)", "Battery Voltage (Volt)"]
        with open(r"cache_drive/Data Greenhouse %s to %s.csv" % (date_start, date_end),
                  mode="w", newline="") as file:
            writer = csv.writer(file)
            writer.writerow(column)
            writer.writerows(data)
            path = os.path.abspath(file.name)
```

Gambar 15 Fungsi get_drive Bagian 1

Sistem melakukan protokol otorisasi OAuth 2.0 dengan menggunakan parameter `path_token` yang mengarah ke `token.json` agar dapat mengakses Google Drive API. Setelah mendapatkan otorisasi, sistem akan mengunggah berkas tersebut dalam ekstensi CSV lalu mengubah *permission*-nya menjadi *reader* dan *anyone*. Selanjutnya, sistem akan memasukkan `fileId` ke dalam

database serta mengirimkan tautan Google Drive ke pengguna yang disertai dengan batas masa berlaku. Apabila terdapat *error* saat menjalankan fungsi ini, pengguna akan mendapatkan pemberitahuan. Setelah semuanya selesai, sistem akan menghapus berkas yang disimpan di dalam penyimpanan lokal untuk menghemat ruang penyimpanan.

```
        #Otorisasi Google Drive
        creds = Credentials.from_authorized_user_file(path_token, scopes)
        service = build("drive", "v3", credentials=creds)
        #Menggunggah Berkas
        file_metadata = {
            "name": "Data Greenhouse %s to %s.csv" % (date_start, date_end),
            "parents": [folder]}
        media = MediaFileUpload(
            path,
            mimetype="text/csv",
            resumable=True)
        file = service.files().create(
            body=file_metadata,
            media_body=media,
            fields="id, webViewLink").execute()
        media.__del__()
        #Mengatur Akses Berkas
        permission = {
            "role": "reader",
            "type": "anyone"}
        service.permissions().create(
            fileId=file.get("id"),
            body=permission).execute()
        #Memasukkan fileId ke Database
        timestamp = datetime.now()
        cursor.execute("INSERT INTO google_drive (timestamp, file_id) VALUES ('%s', '%s')"\
            % (timestamp.strftime("%Y-%m-%d %H:%M:%S"), file.get("id")))
        db.commit()
        #Mengirim Tautan ke Telegram
        await update.message.reply_text("Berlaku sampai %s\n%s" %
            ((timestamp + timedelta(days=7)).strftime("%Y-%m-%d %H:%M:%S"),
                file.get("webViewLink")))
    #Error
    except Exception as error:
        logging.warning(traceback.format_exc().replace("\n", "\n  "))
        await update.message.reply_text(
            "Mohon maaf, terjadi kesalahan sistem saat sedang memproses data. "\
            "Silahkan coba beberapa saat lagi.\nError : %s" % error)
    #Menghapus Berkas Cache
    try: os.remove(path)
    except: pass
```

Gambar 16 Fungsi get_drive Bagian 2

- **check_db**

Sistem akan melakukan verifikasi pada *userid* pengguna. Jika tidak terdaftar, maka pengguna akan mendapatkan pemberitahuan bahwa *userid* tidak terdaftar. Jika pengguna terdaftar, maka sistem akan melakukan koneksi ke *database* lokal. Hasil dari koneksi tersebut nantinya akan dikirim ke pengguna.

```
#Mengecek Koneksi Database
async def check_db(update, context):
    #Verifikasi User
    if verify(update.message.from_user.id):
        await update.message.reply_text(
            "Mohon maaf, User ID %s milik Anda tidak terdaftar dalam sistem kami."\
            % update.message.from_user.id)
        return None
    try:
        #Koneksi ke Database
        db = sql.connect(
            host = host,
            user = user,
            password = password,
            database = database)
        db.close()
        await update.message.reply_text("Local database berhasil terhubung")
    except Exception as error:
        #Error
        logging.warning(traceback.format_exc().replace("\n", "\n  "))
        await update.message.reply_text("Local database tidak terhubung\nError : %s" % error)
```

Gambar 17 Fungsi check_db

- **delete_drive**

Fungsi ini bertugas untuk menghapus berkas yang ada di Google Drive apabila sudah melewati 7 hari sejak berkas tersebut diunggah. Sistem akan mengambil data `fileId` beserta *timestamp* yang telah disimpan dalam *database*. Selanjutnya, sistem akan melakukan iterasi sebanyak `fileId` yang ada. Jika *timestamp* berkas sudah lewat 7 hari, maka berkas tersebut akan dihapus dari Google Drive. Selain itu, sistem juga akan menghapus `fileId` tersebut dari *database*.

```
#Menghapus Berkas yang Lebih dari Seminggu
async def delete_drive(context):
    try:
        #Koneksi ke Database
        db = sql.connect(
            host = host,
            user = user,
            password = password,
            database = database
            )
        cursor = db.cursor()
        cursor.execute("SELECT * FROM google_drive")
        data = cursor.fetchall()
        #Iterasi fileId
        for datum in data:
            now = datetime.now()
            past = datum[1]
            #Mengecek Selisih Waktu Berkas
            if (now - past).days >= 7:
                try:
                    #Menghapus Berkas di Google Drive
                    creds = Credentials.from_authorized_user_file(path_token, scopes)
                    service = build("drive", "v3", credentials=creds)
                    service.files().delete(fileId=str(datum[2])).execute()

                    #Menghapus Baris di Database
                    cursor.execute("DELETE FROM google_drive WHERE id = %s" % datum[0])
                    db.commit()
                except:
                    #Error
                    logging.warning(traceback.format_exc().replace("\n", "\n  "))
        db.close()
    #Error
    except:
        logging.warning(traceback.format_exc().replace("\n", "\n  "))
```

Gambar 18 Fungsi delete_drive

- **error_handler**

Fungsi ini bertugas untuk menyimpan *traceback error* apabila suatu saat terjadi *error* saat menjalankan program utama.

```
#Handler untuk Error
async def error_handler(update, context):
    #Memasukkan Traceback Error ke Log
    traceback_string = "".join(traceback.format_exception(None,\
        context.error, context.error.__traceback__))
    logging.warning(traceback_string.replace("\n", "\n  "))
```

Gambar 19 Fungsi error_handler

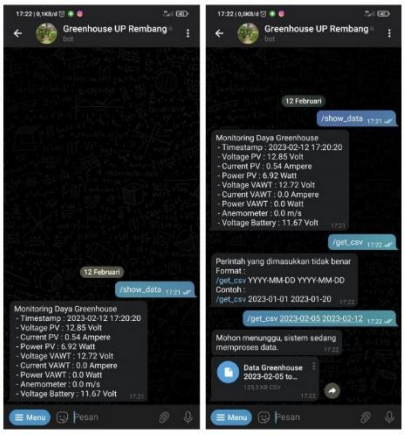
3.3.5 Membuat Objek dan Menjalankan Chatbot

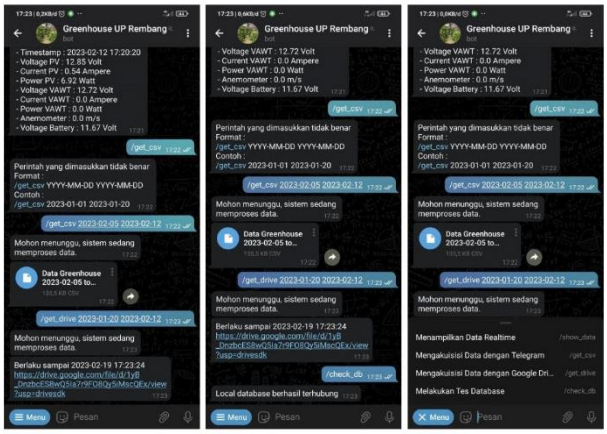
```python
if __name__ == "__main__":
    #Membuat Objek dari class Application
    application = Application.builder().token(token).build()
    #Membuat Handler untuk Perintah
    application.add_handler(CommandHandler("check_db", check_db))
    application.add_handler(CommandHandler("show_data", show_data))
    application.add_handler(CommandHandler("get_csv", get_csv))
    application.add_handler(CommandHandler("get_drive", get_drive))
    #Membuat Handler untuk Error
    application.add_error_handler(error_handler)
    #Membuat Job Queue untuk Menghapus Berkas
    application.job_queue.run_repeating(delete_drive, interval=3600)
    #Menjalankan Bot
    application.run_polling()
```

Gambar 20 Program untuk Membuat Objek dan Menjalankan *Chatbot*

Program akan membuat objek dengan menggunakan parameter *token* dari Telegram dengan nama application. Objek ini nantinya akan mengatur jalannya fungsi check_db, show_data, get_csv, serta get_drive saat *chatbot* mendapatkan *trigger command*. Objek application juga akan menjalankan fungsi error_handler jika terjadi *error* atau kegagalan sistem. Selain itu, objek ini juga menjalankan job_queue pada fungsi delete_drive dengan interval tiap perulangannya sebesar 3600 detik. Yang terakhir, objek akan menjalankan *method* run_polling untuk menjalankan *chatbot* secara terus menerus hingga terdapat *exception* KeyboardInterrupt atau *error*.

3.4 Hasil Chatbot





Gambar 21 Hasil *Chatbot*

4. PENUTUP

4.1 Kesimpulan

1. Telegram bot dapat digunakan sebagai *interface* yang menyediakan kecepatan dan kemudahan dalam proses akuisisi data.
2. Pada program *chatbot* digunakan beberapa *library* seperti telegram.ext, mysql.connector, datetime, googleapiclient, google.oauth2, os, csv, json, traceback, serta logging.
3. Terdapat empat *command* yang dapat dikirim ke *chatbot*, yaitu `/show_data`, `/get_csv`, `/get_drive`, dan `/check_db`. Setiap *command* memiliki fungsi yang berbeda-beda.
4. Ukuran berkas yang dapat dikirim melalui Telegram secara langsung maksimal 48 MB karena batasan dari API bot sebesar 50 MB.
5. Semakin jauh rentang tanggal yang diinginkan, semakin lama juga waktu yang dibutuhkan untuk memproses data tersebut.
6. Pengguna dengan *userid* yang tidak terdaftar dalam sistem tidak akan bisa menggunakan *command* pada *chatbot*.
7. Google Drive API dapat digunakan untuk mengunggah, menyunting, maupun menghapus berkas secara otomatis.

4.2 Saran

Penulis masih menemui beberapa kekurangan dalam *chatbot* Telegram yang dibangun. *Chatbot* sesekali tidak merespons saat pengguna mengirimkan *command*. Hal tersebut disebabkan karena dua hal, yaitu koneksi internet tidak stabil serta mini *computer* mati karena kekurangan daya. Kekurangan ini dapat ditangani dengan mengganti penyedia layanan internet yang memiliki koneksi stabil serta menyediakan kebutuhan daya yang cukup. Selain itu, pengiriman berkas juga mengalami kendala apabila ukuran berkas terlalu besar. Hal tersebut dapat terjadi karena proses pengunggahan membutuhkan waktu yang lama, sedangkan koneksi internet tidak mendukung sehingga sering terjadi *timeout* atau kehabisan waktu. Kekurangan ini dapat disiasati dengan memilih rentang waktu yang tidak terlalu jauh agar ukuran berkas tidak terlalu besar.

DAFTAR PUSTAKA

[1] PT PLN Nusantara Power, "Operation and Maintenance," 2023. https://www.plnnusantarapower.co.id/operation-and-maintenance/ (accessed Mar. 01, 2023).

[2] Rachmad Setiawan, *Teknik Akuisisi Data*. Yogyakarta: Graha Ilmu, 2008.

[3] P. Dewonoto, L. Santoso, I. Riski, N. Kholik, and M. R. Akbar, "Penerapan Artificial Intelligence dalam Aplikasi Chatbot sebagai Media Informasi dan Pembelajaran mengenai Kebudayaan Bangsa," vol. 6, no. 3, pp. 579–589, 2021.

[4] Telegram, "Telegram FAQ," 2023. https://telegram.org/faq?setln=en (accessed Feb. 05, 2023).

[5] Google Developers, "Introduction to Google Drive API," 2022. https://developers.google.com/drive/api/guides/about-sdk (accessed Feb. 04, 2023).

BIODATA PENULIS

Naufal Yafi Susanto (21060120120011). Lahir di Kabupaten Batang pada tanggal 28 Mei 2002. Saat ini penulis sedang menempuh pendidikan sarjana S1 Teknik Elektro di Fakultas Teknik Universitas Diponegoro Semarang dengan konsentrasi Teknologi Informasi angkatan 2020.

Saya menyatakan bahwa segala informasi yang tersedia di makalah ini adalah benar, merupakan hasil karya sendiri, bebas dari plagiat, dan semua karya orang lain telah di kutip dengan benar.

Naufal Yafi Susanto
NIM. 21060120120011

Pengesahan
Telah disetujui untuk diajukan pada Seminar Kerja Praktek.

Menyetujui,
Dosen Pembimbing

Budi Setiyono, S.T., M.T.
NIP. 1970052120001210