INSANtek – Jurnal Inovasi dan Sains Teknik Elektro
Volume 4 No. 1 Mei 2023
E-ISSN: 2722-547X

Analisis Chatbot Otomatisasi Tugas Administratif dan Manajemen Dalam Lingkungan Digital Dengan Menggunakan Python

Sigit Wibawa

Universitas Bina sarana Informatika
e-mail: sigit.stb@bsi.ac.id

Abstrak - Dalam era digital saat ini, otomatisasi tugas administratif dan manajemen telah menjadi suatu kebutuhan yang mendesak dalam upaya meningkatkan efisiensi dan produktivitas. Dalam penelitian ini, kami menyajikan analisis otomatisasi tugas administratif dan manajemen menggunakan platform chatbot yang dikembangkan dengan Python. Dalam analisis ini, kami mengidentifikasi beberapa tugas administratif yang sering terjadi dalam lingkungan digital, termasuk pengelolaan User atau user management, pengelolaan tugas, dan penyediaan informasi dasar kepada pengguna. Penelitian ini mencakup analisis kebutuhan dan identifikasi tugas administratif dan manajemen yang paling relevan dalam konteks lingkungan digital. pengembangan dan konfigurasi Chatbot menggunakan Python dan pustaka terkait untuk mengimplementasikan logika dan proses yang diperlukan untuk tugas-tugas administratif, pengujian dan evaluasi kinerja Chatbot dalam menangani tugas-tugas tersebut, termasuk kecepatan respons, akurasi, dan kehandalan. Hasil analisis menunjukkan bahwa Chatbot mampu secara efektif dan efisien menangani tugas-tugas administratif dan manajemen dalam lingkungan digital. Chatbot dapat memberikan respons yang cepat dan akurat kepada pengguna, mengurangi waktu dan usaha yang diperlukan untuk tugas-tugas rutin, dan membebaskan sumber daya manusia untuk fokus pada tugas yang lebih strategis. Penelitian ini memberikan kontribusi pada pengembangan chatbot berbasis Python untuk otomatisasi tugas administratif dan manajemen dalam lingkungan digital. Penggunaan Chatbot dapat membantu organisasi meningkatkan efisiensi operasional, mengoptimalkan pengelolaan waktu, dan memberikan layanan yang lebih responsif kepada pelanggan.

Kata Kunci: Chatbot, GoogleColab, Python.

Abstract - In today's digital era, automation of administration and management tasks is an urgent need in efforts to increase efficiency and productivity. In this study, we present an analysis of the automation of administration and management tasks using a chatbot platform developed in Python. In this analysis, we identify some of the administrative tasks that occur frequently in digital environments, including user management, task management, and providing basic information to users. This research includes a needs analysis and identification of administrative and management tasks that are most relevant in the context of the digital environment. Chatbot development and configuration uses Python and related libraries to implement the logic and processes needed for administrative tasks, and test and evaluate the Chatbot's performance in handling these tasks, including response speed, accuracy, and reliability. The results of the analysis show that Chatbot is able to handle administration and management tasks effectively and efficiently in a digital environment. Chatbots can provide users with fast and accurate responses, reduce the time and effort required for routine tasks, and free up human resources to focus on more strategic tasks. This research contributes to the development of a Python-based chatbot to automate administrative and management tasks in a digital environment. Using Chatbots can help organizations improve operational efficiency, optimize time management, and provide a more responsive service to customers.

Keywords: Chatbot, Googlecolab, Python

PENDAHULUAN

Dalam era digital yang semakin maju, tugas administratif dan manajemen dapat menjadi rumit dan memakan waktu. Untuk mengatasi tantangan ini, ChatBot dan Bot Admin telah muncul sebagai solusi yang menjanjikan (Ramasubbu, D., Baskaran, K., & Yann, G. (2018). ChatBot adalah program komputer yang dirancang untuk melakukan tugas-tugas administratif secara otomatis, menghemat waktu dan upaya manusia dalam prosesnya. Pengembangan ChatBot telah mendapatkan perhatian yang signifikan dalam komunitas pengembang karena potensi untuk meningkatkan efisiensi dan produktivitas. Python, sebagai salah satu bahasa pemrograman yang populer dan fleksibel, telah digunakan secara luas dalam pengembangan ChatBot (K., H. K., Palakurthi, A. K., Putnala, V., & Kumar K., A. (2020)).

Python menawarkan berbagai pustaka dan kerangka kerja yang dapat mempercepat dan menyederhanakan proses pembuatan ChatBot (Kohli, B., Choudhury, T., Sharma, S., & Kumar, P. (2018)).

Penelitian sebelumnya telah mengidentifikasi beberapa pustaka dan kerangka kerja Python yang dapat digunakan dalam pengembangan ChatBot seperti Python-telegram-bot, Tweepy, dan ChatterBot. Namun, penelitian yang lebih mendalam diperlukan untuk menganalisis fitur-fitur yang dapat diimplementasikan dalam ChatBot dan Bot Admin menggunakan Python dan untuk mengevaluasi kemampuan dan kinerja pustaka dan kerangka kerja tersebut (Gunawan, T. S., Falelmula Babiker, A. B., Ismail, N., & Effendi, M. R. (2021)).

Dalam konteks ini, penelitian ini bertujuan untuk mempelajari dan mendalami penggunaan Python dalam pengembangan ChatBot dengan menganalisis berbagai pustaka dan kerangka kerja yang tersedia. Dengan mengeksplorasi fitur-fitur yang dapat diimplementasikan, penelitian ini dapat memberikan pemahaman yang lebih baik tentang potensi dan batasan dalam pengembangan ChatBot menggunakan Python.

Melalui penelitian ini, diharapkan pengembang dapat mendapatkan wawasan yang lebih baik tentang pilihan pustaka dan kerangka kerja yang tersedia dan bagaimana menggunakannya secara efektif dalam mengembangkan ChatBot dan Bot Admin. Hal ini akan membantu dalam meningkatkan efisiensi dan produktivitas dalam tugas-tugas administratif dan manajemen, serta memberikan kontribusi bagi pengembangan teknologi otomasi yang lebih canggih.

No.	Pustaka / Kerangka Kerja	Bahasa Pemrograman	Platform	Fitur Utama
1	Python-telegram-bot	Python	Telegram	Interaksi dengan pengguna, manajemen grup, pengaturan keyboard inline
2	Tweepy	Python	Twitter	Interaksi dengan API Twitter, pengelolaan tweet, pemantauan aliran (stream)
3	ChatterBot	Python	Python,Django,Flask,Node.JS	Chatbot dengan kemampuan belajar, generasi respons, pengelolaan korpus percakapan
4	Selenium	Python	Web dan berbagai Aplikasi E2E	Otomatisasi tugas di web, interaksi dengan halaman web, scraping data
5	Botpress	JavaScript	Open Source	Pembangunan bot dengan alur percakapan, integrasi NLP, manajemen respons
6	Microsoft Bot Framework	C#, JavaScript	Berbagai macam: Teams, Slack,Skype,FB	Pembangunan bot multichannel, pengenalan suara, pemantauan analitik

Tabel 1. Perbandingan Framework ChatBot

Program ChatBot harus memiliki berbagai fitur tergantung pada kebutuhan spesifik. Berikut ini adalah beberapa contoh fitur yang wajib ada dalam program ChatBot:

1. Pengelolaan Pengguna: ChatBot dapat memiliki kemampuan untuk mendaftarkan, menghapus, atau memodifikasi akun pengguna. Ini melibatkan fitur seperti pendaftaran pengguna baru, login, pengaturan profil, dan manajemen hak akses.

2. Manajemen Tugas: ChatBot dapat membantu dalam mengatur dan melacak tugas-tugas administratif. Fitur ini dapat mencakup penjadwalan tugas, pengingat, delegasi tugas, dan pemantauan status tugas.

3. Pengarsipan dan Pencarian Data: ChatBot dapat memiliki kemampuan untuk menyimpan dan mengatur data penting, seperti dokumen, catatan, atau riwayat percakapan. Fitur ini dapat mencakup pengarsipan data, pencarian berdasarkan kata kunci, dan pengelompokan data.

4. Notifikasi dan Pengumuman: ChatBot dapat mengirimkan notifikasi atau pengumuman kepada pengguna terkait informasi penting atau perubahan yang terjadi. Fitur ini dapat mencakup pemberitahuan melalui pesan teks, email, atau pemberitahuan dalam aplikasi.

5. Integrasi dengan Platform Lain: ChatBot dapat terintegrasi dengan platform lain yang digunakan dalam konteks administratif atau manajerial, seperti platform obrolan, sistem manajemen proyek, atau platform manajemen pelanggan. Ini memungkinkan ChatBot untuk berinteraksi dengan platform lain dan mengambil data atau melakukan tugas secara otomatis.

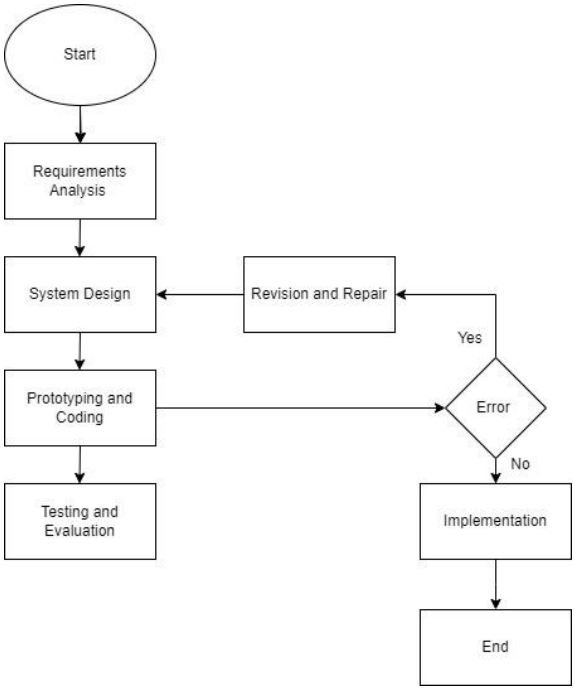
6. Analisis dan Pelaporan: ChatBot membantu dalam menganalisis data dan menghasilkan laporan yang berguna untuk pengambilan keputusan. Fitur ini dapat mencakup pembuatan laporan otomatis, visualisasi data, atau analisis statistik.

7. Interaksi dengan Pengguna: ChatBot dapat merespons pertanyaan atau permintaan pengguna secara otomatis dan dapat memberikan informasi, memecahkan masalah umum, atau mengarahkan pengguna ke sumber daya yang tepat.

8. Keamanan dan Otorisasi: ChatBot harus memiliki fitur keamanan yang memadai untuk melindungi data sensitif dan mengelola akses pengguna. Ini mencakup fitur seperti autentikasi pengguna, pengendalian akses, dan enkripsi data. Setiap program ChatBot akan memiliki implementasi dan fitur yang unik sesuai dengan kebutuhan spesifik. Dalam pengembangan program ChatBot, penting untuk merencanakan dan merancang fitur-fitur yang sesuai dengan kebutuhan pengguna dan tujuan administratif atau manajerial yang ingin dicapai

2. Metode Penelitian

Metodologi digunakan dalam penelitian ini adalah Software Development Life Cycle dengan model Waterfall yang telah dimodifikasi, dimana proses perbaikan hanya akan dilakukan setelah melalui tahap pengujian dan evaluasi (Sinha, A., & Das, P. (2021)) yang dapat dilihat dalam diagram metode penelitian pada gambar 1.



Gambar 1. Diagram Metode Penelitian

Selanjutnya Metode penelitian yang digunakan dalam jurnal ini adalah studi literatur. Studi literatur dilakukan dengan melakukan penelusuran literatur yang relevan yang berkaitan dengan pengembangan ChatBot menggunakan Python.

Berikut adalah langkah-langkah yang diambil dalam metode penelitian ini:

1. Penentuan Ruang Lingkup: Penentuan ruang lingkup penelitian adalah tahap awal yang melibatkan mengidentifikasi topik penelitian, yaitu pengembangan ChatBot menggunakan Python dan Google Colab yang merupakan platform komputasi awan (cloud computing) gratis yang disediakan oleh Google. Ini memungkinkan pengguna untuk menjalankan kode Python dan mengembangkan proyek menggunakan lingkungan Jupyter Notebook yang di-hosting di cloud (Canesche, M., Braganca, L., Neto, O. P. V., Nacif, J. A., & Ferreira, R. (2021)).

2. Identifikasi Sumber Informasi: Pada tahap ini, sumber-sumber informasi yang relevan untuk penelitian ini diidentifikasi. Sumber informasi dapat mencakup artikel jurnal, konferensi, buku, laporan teknis, dan situs web resmi dari pustaka dan kerangka kerja Python yang akan dianalisis.

3. Penelusuran Literatur: Langkah ini melibatkan melakukan penelusuran literatur menggunakan basis data dan mesin pencari akademik yang relevan. Kata kunci yang relevan, seperti "ChatBot dan Bot Admin", "Python", "pustaka Python", "kerangka kerja Python", "analisis", dan "interaksi pengguna", digunakan untuk mencari sumber-sumber yang sesuai (T, J. J., & Swaminathan, J. (2022)).

4. Seleksi dan Evaluasi Literatur: Setelah penelusuran literatur selesai, langkah selanjutnya adalah memilih literatur yang paling relevan dengan topik penelitian. Pada tahap ini, abstrak, ringkasan, atau kutipan dari literatur dievaluasi untuk memastikan relevansi dan kualitas.

5. Analisis dan Sintesis: Pada tahap ini, literatur yang dipilih dianalisis secara mendalam. Pustaka dan kerangka kerja Python yang berhubungan dengan pengembangan ChatBot dan Bot Admin dieksplorasi secara lebih rinci. Fitur-fitur yang dapat diimplementasikan dan kemampuan masing-masing pustaka dan kerangka kerja dievaluasi dan dicatat.

6. Penulisan Jurnal: Langkah terakhir adalah menulis jurnal berdasarkan temuan dan analisis dari studi literatur yang dilakukan. Jurnal mencakup latar belakang penelitian, metode penelitian, hasil temuan, dan kesimpulan yang diambil dari analisis literatur. Metode penelitian ini dirancang untuk mengumpulkan informasi yang relevan dan terkini mengenai pengembangan chatbot dan ChatBot dan Bot Admin menggunakan Python. Melalui analisis literatur yang mendalam, penelitian ini memberikan pemahaman yang komprehensif tentang pilihan pustaka dan kerangka kerja yang tersedia untuk pengembangan ChatBot dan Bot Admin serta fitur-fitur yang dapat diimplementasikan dalam konteks tersebut. Selanjutnya ditampilkan Use Case diagram untuk menunjukkan hubungan antara entitas yang menggunakan atau bergantung pada entitas lain. Dalam kasus chatbot manajemen, kita dapat memiliki beberapa entitas yang terlibat. Berikut adalah deskripsi uses diagram untuk contoh chatbot dan ChatBot dan Bot Administrasi dan manajemen:

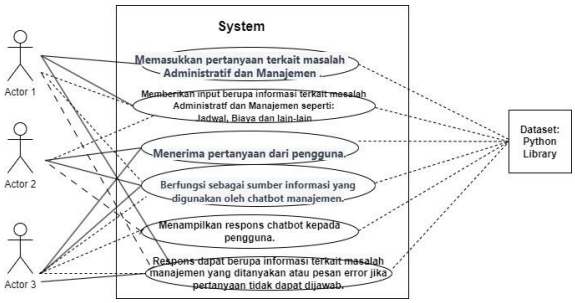
1. Entitas "Pengguna":
   - Mengirim pertanyaan tentang masalah manajemen.
   - Memberikan masukan berupa informasi terkait masalah manajemen, seperti anggaran proyek atau penjadwalan.

2. Entitas "Chatbot Manajemen":
   - Menerima pertanyaan dari pengguna.
   - Memproses pertanyaan dan menentukan jenis masalah manajemen yang diajukan.
   - Berinteraksi dengan pengguna untuk meminta informasi tambahan, jika diperlukan.
   - Menghitung atau memanipulasi data berdasarkan masukan pengguna.
   - Menghasilkan respons yang relevan berdasarkan masalah manajemen yang diajukan.

3. Entitas "Sumber Data":
- Berfungsi sebagai sumber informasi yang digunakan oleh chatbot manajemen.
- Contoh sumber data dapat mencakup database anggaran proyek, jadwal tugas, atau informasi produk.

4. Entitas "Output":
- Menampilkan respons chatbot kepada pengguna.
- Respons dapat berupa informasi terkait masalah manajemen yang ditanyakan atau pesan error jika pertanyaan tidak dapat dijawab.

Dalam uses diagram, entitas-entitas ini dapat digambarkan sebagai kotak atau lingkaran dengan panah yang menunjukkan arah aliran informasi antara entitas-entitas tersebut. Kotak atau lingkaran dapat diberi label yang menjelaskan peran entitas dalam sistem.



Gambar 2. Use Case Diagram

Dalam konteks chatbot dan ChatBot dan Bot Admin, persamaan matematika tidak sering digunakan secara langsung. Namun, beberapa konsep matematika dapat relevan dalam pengembangannya. Berikut ini adalah beberapa konsep yang dapat digunakan:

1. Fungsi: Dalam Chatbot dan ChatBot dan Bot Admin, fungsi digunakan untuk memetakan masukan pengguna ke respons yang sesuai. Misalnya, dengan menggunakan fungsi if-else atau switch-case untuk mengarahkan alur percakapan berdasarkan masukan pengguna.

2. Probabilitas: Beberapa kasus, Chatbot dan ChatBot dan Bot Admin menggunakan model probabilistik seperti model bahasa atau *model Machine Learning*. Model ini dapat digunakan untuk menghitung probabilitas respons yang paling tepat berdasarkan masukan pengguna atau konteks percakapan sebelumnya.

3. Pemrosesan Bahasa Alami atau Natural Language Processing: Dalam chatbot dan ChatBot dan Bot Admin yang lebih canggih, teknik pemrosesan bahasa alami (Natural Language Processing/NLP) digunakan. Hal ini melibatkan penggunaan metode matematika seperti pemodelan bahasa statistik, parsing, atau klasifikasi teks untuk memahami masukan pengguna dan menghasilkan respons yang tepat (Singh, D., S, S. V., & K, V. (2022)).

4. Algoritma Pencarian: Dalam beberapa kasus, ChatBot dan Bot Admin dapat menggunakan algoritma pencarian seperti pencocokan string atau pencarian berbasis aturan untuk mencocokkan masukan pengguna dengan respons yang telah ditentukan.

Namun, penting untuk dicatat bahwa pengembangan ChatBot dan Bot Admin lebih terkait dengan pemrograman, logika, dan pengolahan data daripada persamaan matematika. Konsep matematika yang relevan digunakan sebagai bagian dari algoritma dan model yang diimplementasikan.

HASIL DAN PEMBAHASAN

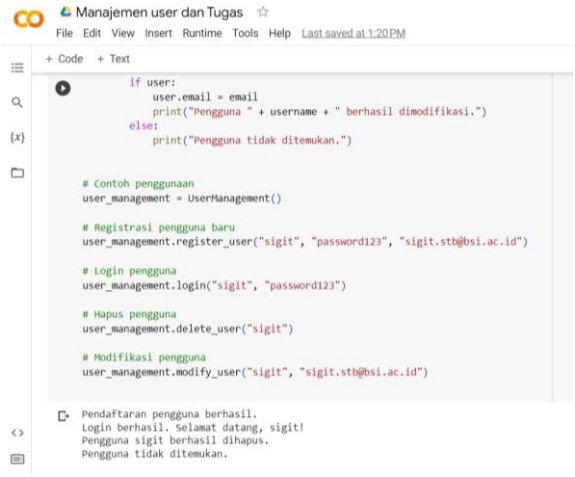
Dengan menggunakan bahasa pemograman Python kita bisa memenuhi kebutuhan Program Chatbot dan ChatBot dan Bot Admin yang memiliki berbagai fitur yang dipersyaratkan. Berikut ini adalah beberapa fitur tersebut:

1. **Pengelolaan Pengguna: ChatBot dan Bot Admin dapat memiliki kemampuan untuk mendaftarkan, menghapus, atau memodifikasi akun pengguna. Ini melibatkan fitur seperti pendaftaran pengguna baru, login, pengaturan profil, dan manajemen hak akses**.

Kode Program:

```
class User:
    def __init__(self, username, password, email, role):
        self.username = username
        self.password = password
        self.email = email
        self.role = role

class UserManagement:
    def __init__(self):
        self.users = []

    def register_user(self, username, password, email):
        # Cek apakah username sudah digunakan
        if self.get_user_by_username(username):
            print("Username sudah digunakan. Silakan pilih username lain.")
            return

        # Buat objek User baru
        user = User(username, password, email, "user")
```

```
        # Tambahkan user ke daftar pengguna
        self.users.append(user)

        print("Pendaftaran pengguna berhasil.")

    def login(self, username, password):
        # Cari pengguna berdasarkan username
        user = self.get_user_by_username(username)

        if user and user.password == password:
            print("Login berhasil. Selamat datang, " + user.username + "!")
        else:
            print("Login gagal. Periksa kembali username dan password Anda.")

    def get_user_by_username(self, username):
        # Cari pengguna berdasarkan username
        for user in self.users:
            if user.username == username:
                return user

        return None

    def delete_user(self, username):
        # Cari pengguna berdasarkan username
        user = self.get_user_by_username(username)

        if user:
            self.users.remove(user)
            print("Pengguna " + username + " berhasil dihapus.")
        else:
            print("Pengguna tidak ditemukan.")

    def modify_user(self, username, email):
        # Cari pengguna berdasarkan username
        user = self.get_user_by_username(username)

        if user:
            user.email = email
            print("Pengguna " + username + " berhasil dimodifikasi.")
        else:
            print("Pengguna tidak ditemukan.")
```



program di atas terdiri dari dua kelas utama: User dan UserManagement. Kelas User digunakan untuk mewakili objek pengguna dengan atribut seperti username, password, email, dan role. Kelas UserManagement adalah kelas yang bertanggung jawab untuk mengelola pengguna dengan fitur seperti pendaftaran pengguna baru, login, penghapusan pengguna, dan modifikasi pengguna.

2. **Fitur Manajemen Tugas: Chatbot dan ChatBot dan Bot Admin dapat membantu dalam mengatur dan melacak tugas-tugas administratif. Fitur ini dapat mencakup penjadwalan tugas, pengingat, delegasi tugas, dan pemantauan status tugas.**

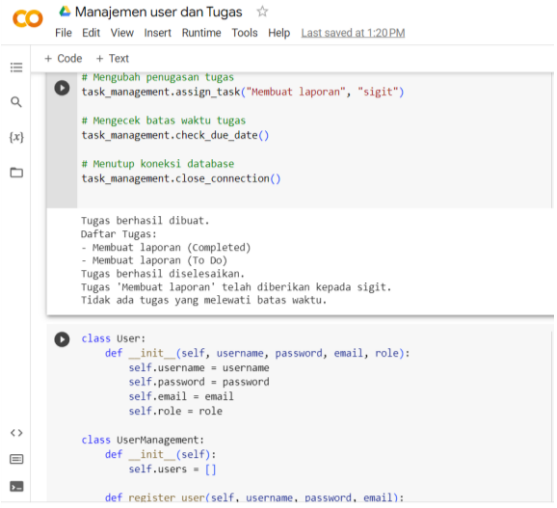
Kode Program:
Program Python yang menggunakan library `datetime` dan `sqlite3` untuk mengimplementasikan Fitur Manajemen Tugas yang mencakup penjadwalan tugas, pengingat, delegasi tugas, dan pemantauan status tugas:

```
import datetime
import sqlite3

# Inisialisasi database
conn = sqlite3.connect('task_management.db')
c = conn.cursor()

# Membuat tabel tugas
c.execute('''CREATE TABLE IF NOT EXISTS tasks
             (title TEXT, description TEXT, assignee TEXT, due_date DATE, status TEXT)''')

class TaskManagement:
    def __init__(self):
        self.conn = sqlite3.connect('task_management.db')
        self.c = self.conn.cursor()

    def create_task(self, title, description, assignee, due_date):
```

```
        # Tambahkan tugas ke database
        self.c.execute("INSERT INTO tasks VALUES (?, ?, ?, ?, ?)",
                       (title, description, assignee, due_date, "To Do"))
        self.conn.commit()
        print("Tugas berhasil dibuat.")

    def view_tasks(self):
        # Ambil daftar tugas dari database
        self.c.execute("SELECT * FROM tasks")
        tasks = self.c.fetchall()

        if not tasks:
            print("Tidak ada tugas yang tersedia.")
        else:
            print("Daftar Tugas:")
            for task in tasks:
                print(f"- {task[0]} ({task[4]})")

    def complete_task(self, title):
        # Update status tugas menjadi "Completed" di database
        self.c.execute("UPDATE tasks SET status = 'Completed' WHERE title = ?", (title,))
        self.conn.commit()
        print("Tugas berhasil diselesaikan.")

    def assign_task(self, title, assignee):
        # Update penugasan tugas di database
        self.c.execute("UPDATE tasks SET assignee = ? WHERE title = ?", (assignee, title))
        self.conn.commit()
        print(f"Tugas '{title}' telah diberikan kepada {assignee}.")

    def check_due_date(self):
        today = datetime.date.today()

        # Ambil tugas-tugas yang melewati batas waktu dari database
        self.c.execute("SELECT * FROM tasks WHERE due_date <= ? AND status != 'Completed'", (today,))
        overdue_tasks = self.c.fetchall()

        if not overdue_tasks:
            print("Tidak ada tugas yang melewati batas waktu.")
        else:
            print("Tugas-tugas yang melewati batas waktu:")
            for task in overdue_tasks:
                print(f"- {task[0]}")

    def close_connection(self):
        # Tutup koneksi database
        self.conn.close()

# Contoh penggunaan
task_management = TaskManagement()

# Membuat tugas baru
task_management.create_task("Membuat laporan", "Buat laporan bulanan", "sigit", datetime.date(2023, 7, 31))

# Menampilkan daftar tugas
task_management.view_tasks()

# Menyelesaikan tugas
task_management.complete_task("Membuat laporan")

# Mengubah penugasan tugas
task_management.assign_task("Membuat laporan", "sigit")

# Mengecek batas waktu tugas
task_management.check_due_date()

# Menutup koneksi database
task_management.close_connection()
```

Python library `**datetime**` untuk memanipulasi tanggal dan waktu, serta `**sqlite3**` untuk menghubungkan dan mengelola database SQLite yang digunakan untuk menyimpan informasi tugas-tugas.

Program ini dibuat untuk mencakup kelas `TaskManagement` yang menggunakan metode-metode untuk membuat tugas baru, melihat daftar tugas, menyelesaikan tugas, mengubah penugasan tugas, dan memeriksa batas waktu tugas yang sudah melewati batasnya. Informasi tugas disimpan dan diambil dari database SQLite menggunakan perintah SQL yang dijalankan dengan bantuan kursor `sqlite3`.



https://colab.research.google.com/drive/12zTf97tNQSxDZknTE9Jf6i5uV7yPsXF5?usp=sharing

KESIMPULAN

Kesimpulan dari penelitian ini adalah bahwa Python menyediakan berbagai pilihan untuk pengembangan Chatbot dan ChatBot dan Bot Admin yang efektif. Penggunaan pustaka dan kerangka kerja yang tepat dapat memungkinkan implementasi fitur-fitur yang diinginkan dalam Chatbot dan ChatBot dan Bot Admin, termasuk interaksi dengan pengguna, pengelolaan tugas, dan integrasi dengan platform lain. Penelitian ini diharapkan memberikan kontribusi dalam memahami kemampuan Python dalam pengembangan Chatbot dan ChatBot dan Bot Admin dan memberikan panduan bagi pengembang dan peneliti yang ingin memanfaatkan Python dalam mengembangkan Chatbot dan ChatBot dan Bot Admin yang efektif. Saran Studi lebih lanjut dapat dilakukan untuk membandingkan kinerja berbagai pustaka dan kerangka kerja yang tersedia dan menerapkan metode pengujian yang lebih mendalam serta penggunaan aplikasi OpenAi seperti ChatGPT untuk memberikan kontribusi yang lebih besar dan pengembangan yang terkini.

REFERENSI

Ramasubbu, D., Baskaran, K., & Yann, G. (2018). Intrusive Plug Management System Using Chatbots in Office Environments. In 2018 Asian Conference on Energy, Power and Transportation Electrification (ACEPT). 2018 Asian Conference on Energy, Power and Transportation Electrification (ACEPT). IEEE. https://doi.org/10.1109/acept.2018.8610869

K., H. K., Palakurthi, A. K., Putnala, V., & Kumar K., A. (2020). Smart College Chatbot using ML and Python. In 2020 International Conference on System, Computation, Automation and Networking (ICSCAN). 2020 International Conference on System, Computation, Automation and Networking (ICSCAN). IEEE. https://doi.org/10.1109/icscan49426.2020.9262426

Kohli, B., Choudhury, T., Sharma, S., & Kumar, P. (2018). A Platform for Human-Chatbot Interaction Using Python. In 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT). IEEE https://doi.org/10.1109/icgciot.2018.8753031

Gunawan, T. S., Falelmula Babiker, A. B., Ismail, N., & Effendi, M. R. (2021). Development of Intelligent Telegram Chatbot Using Natural Language Processing. In 2021 7th International Conference on Wireless and Telematics (ICWT). 2021 7th International Conference on Wireless and Telematics (ICWT). IEEE. https://doi.org/10.1109/icwt52862.2021.9678471

Sinha, A., & Das, P. (2021). Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry. In 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech). 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech). IEEE. https://doi.org/10.1109/iementech53263.2021.9614779

Canesche, M., Braganca, L., Neto, O. P. V., Nacif, J. A., & Ferreira, R. (2021). Google Colab CAD4U: Hands-On Cloud Laboratories for Digital Design. In 2021 IEEE International Symposium on Circuits and Systems (ISCAS). 2021 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE. https://doi.org/10.1109/iscas51556.2021.9401151

T, J. J., & Swaminathan, J. (2022). An Experimental Study of Parallelism in Different Python Frameworks. In 2022 International Conference on Inventive Computation Technologies (ICICT). 2022 International Conference on Inventive Computation Technologies (ICICT). IEEE. https://doi.org/10.1109/icict54344.2022.9850566

Singh, D., S, S. V., & K, V. (2022). A Proposed Approach for Identifying the Connotative Relationship of English Sentences and paragraphs using the NLP package of Python. In 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES). 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES). IEEE. https://doi.org/10.1109/icses55317.2022.9914220

Potluri, T., Shi, Y., Shahriar, H., Lo, D., Parizi, R., Chi, H., & Qian, K. (2023). Secure Software Development in Google Colab. In 2023 IEEE World AI IoT Congress (AIIoT). 2023 IEEE World AI IoT Congress (AIIoT). IEEE. https://doi.org/10.1109/aiiot58121.2023.10174336