LAPORAN AKHIR PROJECT PEMROGRAMAN DASAR
"TASK REMINDER APPLICATION"

DISUSUN OLEH :

Firda Ayla Mutiahadi 25031554256

Kameliatuz Zahra 25031554263

Daanya Kirana Arifa 25031554272

2025G

DOSEN PENGAMPU :

Hassanuddin Al-Habib, S.Si., M.Si.

Dr. Heri Purnawan, S.Si., M.Si.

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

PROGRAM STUDI S1 SAINS DATA

UNIVERSITAS NEGERI SURABAYA

2025

DAFTAR ISI

PENDAHULUAN 3
A. Latar Belakang 3
B. Rumusan Masalah 4
C. Tujuan 4
ANALISIS DAN PERANCANGAN 5
D. Analisis Kebutuhan Aplikasi 5
1. Identitas Aplikasi 5
2. Kebutuhan Fungsional 5
3. Kebutuhan Non-Fungsional 6
4. Teknis Implementasi 6
5. Limitasi Sistem 7
6. Use Cases 7
7. Kelebihan dan Kekurangan 7
E. Diagram Alur 9
1. Alur utama 9
2. Alur Registrasi (Sign Up) 9
3. Alur Login 10
4. Alur Tambah Tugas 10
5. Alur Reminder System 11
6. Alur Edit Tugas 11
7. Alur Kelola Mahasiswa (Admin) 12
8. Alur Backup Sistem 12
9. Alur Export to Excel 13
10. Alur Log Out 13
11. Alur Cek Deadline 14
12. Alur Shutdown 14
F. Sketsa Desain Antarmuka 15
IMPLEMENTASI 16
LAMPIRAN 17
DAFTAR PUSTAKA 31

PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi informasi yang semakin pesat telah memberikan kemudahan dalam berbagai aspek kehidupan, termasuk dalam pengelolaan tugas dan aktivitas sehari-hari. Dalam dunia akademik maupun profesional, individu sering dihadapkan pada banyak tugas dengan batas waktu yang berbeda-beda. Tanpa sistem pencatatan dan pengingat yang baik, tugas-tugas tersebut berpotensi terlupakan atau tidak diselesaikan tepat waktu, yang dapat berdampak pada penurunan kinerja dan produktivitas.

Permasalahan tersebut menunjukkan perlunya sebuah aplikasi yang mampu membantu pengguna dalam mengatur, menyimpan, dan memantau tugas secara terstruktur. Aplikasi pengingat tugas hadir sebagai solusi untuk mencatat daftar tugas, mengelompokkan berdasarkan waktu dan prioritas, serta memberikan pengingat kepada pengguna sebelum tenggat waktu tiba. Dengan adanya fitur ini, pengguna diharapkan dapat lebih disiplin dalam mengelola waktu dan memiliki kontrol yang lebih baik terhadap aktivitas yang harus diselesaikan.

Berdasarkan latar belakang tersebut, kami mengembangkan aplikasi pengingat tugas sebagai media pendukung manajemen waktu yang praktis dan mudah digunakan. Aplikasi ini dirancang untuk membantu pengguna mengurangi risiko lupa terhadap tugas, meningkatkan efektivitas penyelesaian pekerjaan, serta mendukung kebiasaan kerja yang lebih terorganisir. Diharapkan aplikasi ini dapat memberikan manfaat nyata bagi pengguna dalam meningkatkan produktivitas dan tanggung jawab terhadap setiap tugas yang dimiliki.

B. Rumusan Masalah

1. Bagaimana merancang dan membangun aplikasi pengingat tugas yang mampu membantu pengguna mencatat dan mengelola tugas secara terstruktur?
2. Bagaimana aplikasi dapat memberikan pengingat yang efektif agar pengguna tidak melewatkan tenggat waktu tugas?
3. Bagaimana merancang antarmuka aplikasi yang sederhana dan mudah digunakan oleh pengguna?

C. Tujuan

1. Merancang dan mengembangkan aplikasi pengingat tugas yang dapat membantu pengguna dalam mencatat dan mengelola tugas secara terstruktur.
2. Menyediakan fitur pengingat agar pengguna dapat menyelesaikan tugas tepat waktu sesuai dengan tenggat yang ditentukan.
3. Menghasilkan aplikasi dengan antarmuka yang sederhana, menarik, dan mudah digunakan.
4. Membantu meningkatkan manajemen waktu, kedisiplinan, dan produktivitas pengguna.
5. Menerapkan konsep pemrograman dan teknologi yang telah dipelajari dalam pembuatan aplikasi pengingat tugas.

ANALISIS DAN PERANCANGAN

D. Analisis Kebutuhan Aplikasi

1. Identitas Aplikasi

- Nama : Task Reminder
- Tipe : Aplikasi Desktop
- Teknologi : (Python)Tkinter, Excel, JSON)
- Target User : Mahasiswa dan Admin

2. Kebutuhan Fungsional

2.1. Manajemen User

- Registrasi dengan validasi (username 3+ karakter, password 6+ karakter)
- Login multi role (mahasiswa, admin)
- Password SHA-256
- Session Management

2.2. Manajemen Tugas

Mahasiswa :

- CRUD tugas pribadi
- Update status (Pending/Completed)
- Lihat tugas sendiri

Admin :

- Kelola data mahasiswa
- Lihat semua tugas

2.3. Sistem Pengingat

- Notifikasi otomatis (H-1, H-0, terlambat)
- Background thread (check 30 detik)
- Manual deadline checker

2.4. Manajemen Data

- Backup otomatis ke Excel
- Export/Import data
- Storage JSON + Excel

3. Kebutuhan Non-Fungsional

3.1. Usability

- GUI intuitif dengan Tkinter
- Warna konsisten
- Navigasi mudah
- Feedback jelas

3.2. Reability

- Validasi input
- Backup otomatis
- Error handling

3.3. Perfoma

- Respon cepat
- Resource minimal
- Background processing

3.4. Security

- Password hashing’
- Role based access
- Input sanitization

4. Teknis Implementasi

Struktur Data :

```
// Users
{"username": "string", "password": "hashed", "role": "string"}

// Tasks
{
  "id_tugas": int,
  "judul": "string",
  "mata_kuliah": "string",
  "deadline": "date",
  "username_pemilik": "string",
  "status": "string",
  "dibuat_oleh": "string"
}
```

Arsitektur :

- Presentation Layer : Tkinter GUI
- Business Logic : TaskReminderApp class
- Data Access : JSON operations
- Storage : JSON/Excel files

5. Limitasi Sistem

Kapasitas :

- Optimal : ≤1000 user, ≤10.000 tugas
- File size : ≤10MB JSON
- Offline only
- Single user focus

Despensasi :

- Python 3.6+
- Pandas (Excel ops)
- Plyer (notifikasi)

6. Use Cases

Mahasiswa :

- Registrasi → Login → Tambah tugas
- Lihat tugas → Edit status → Terima notif

Admin :

- Login → Kelola mahasiswa
- Backup data → Export Excel → Monitoring

7. Kelebihan dan Kekurangan

Kelebihan :

- GUI user friendly
- Multi role access
- Auto reminder system

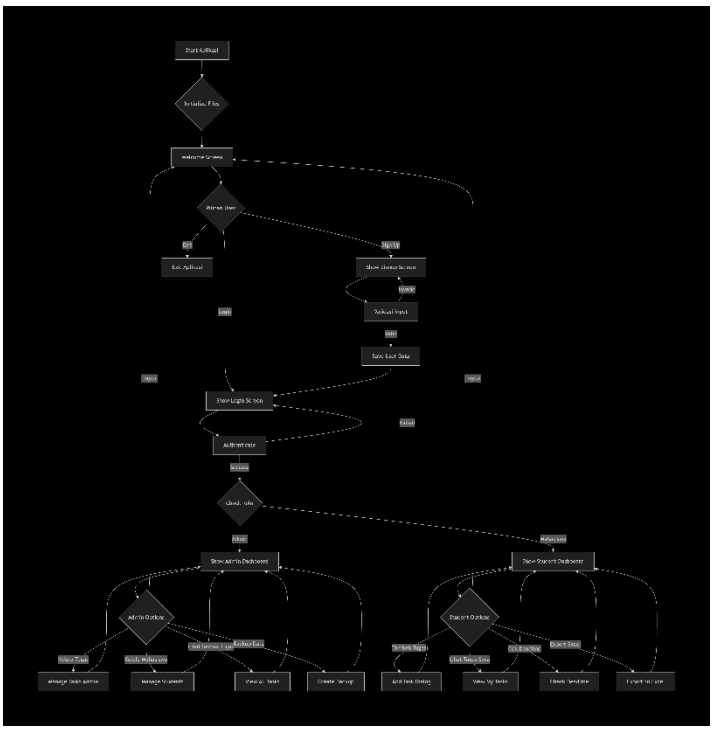
- Cross platform
- Backup otomatis

Kekurangan :

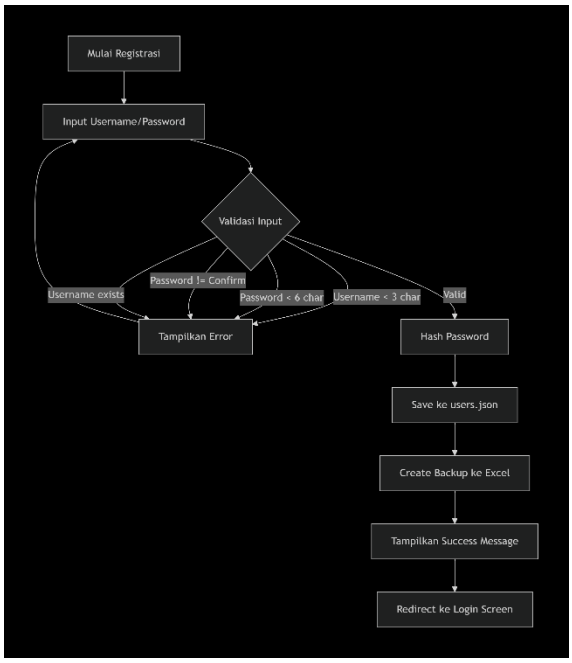
- Offline only
- Fitur dasar saja
- NO mobile version

E. Diagram Alur

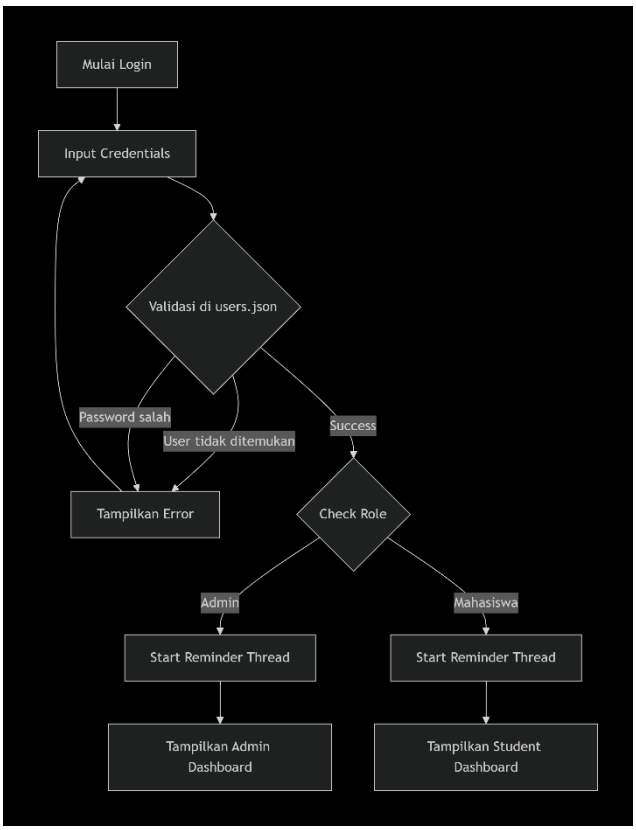
1. Alur utama



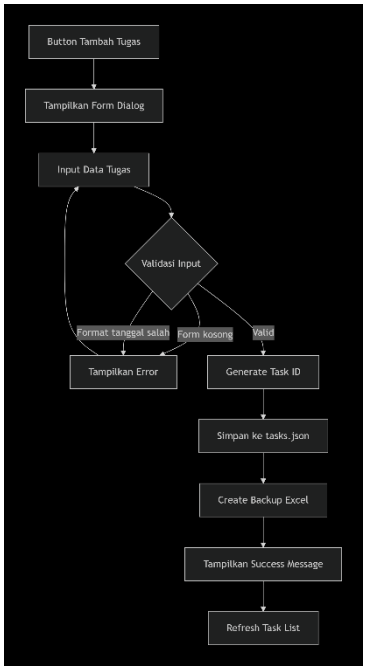
2. Alur Registrasi (Sign Up)



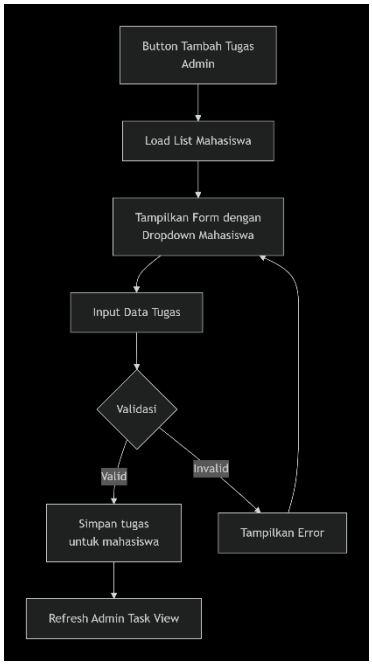
3. Alur Login



4. Alur Tambah Tugas

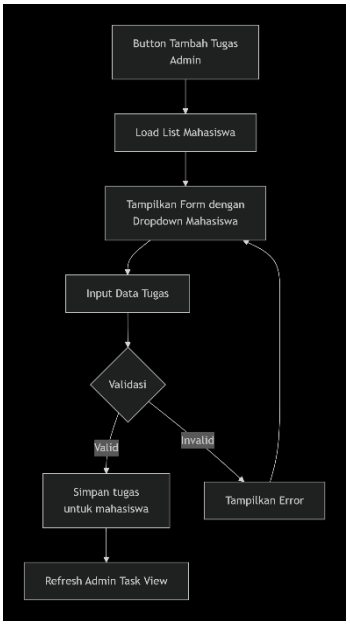


4.1 Mahasiswa

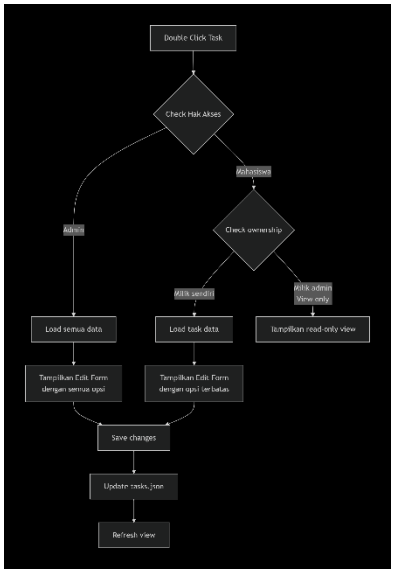


4.2. Admin

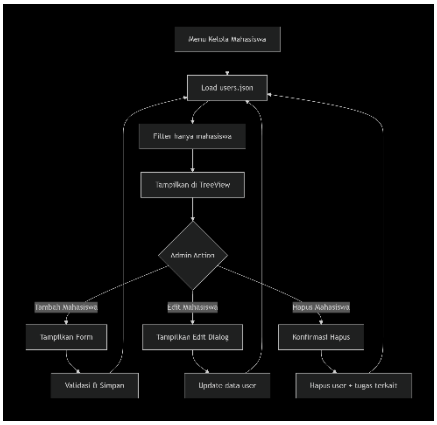
5. Alur Reminder System



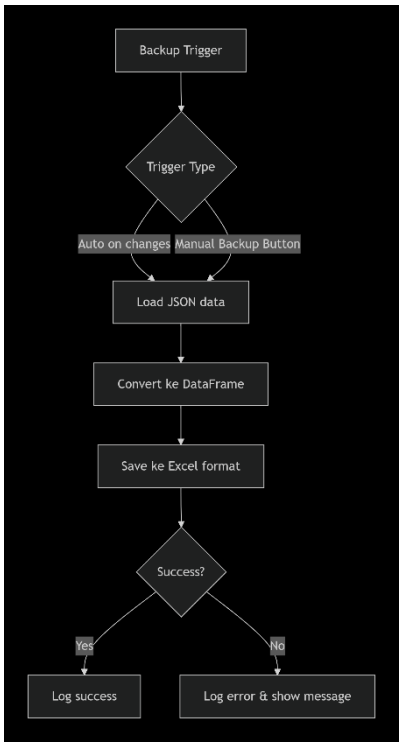
6. Alur Edit Tugas



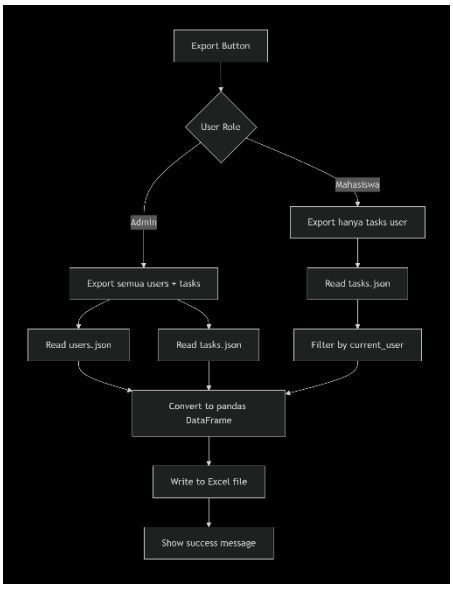
7. Alur Kelola Mahasiswa (Admin)



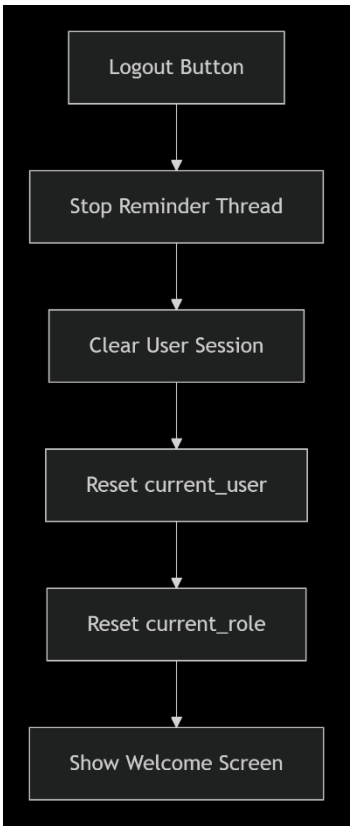
8. Alur Backup Sistem



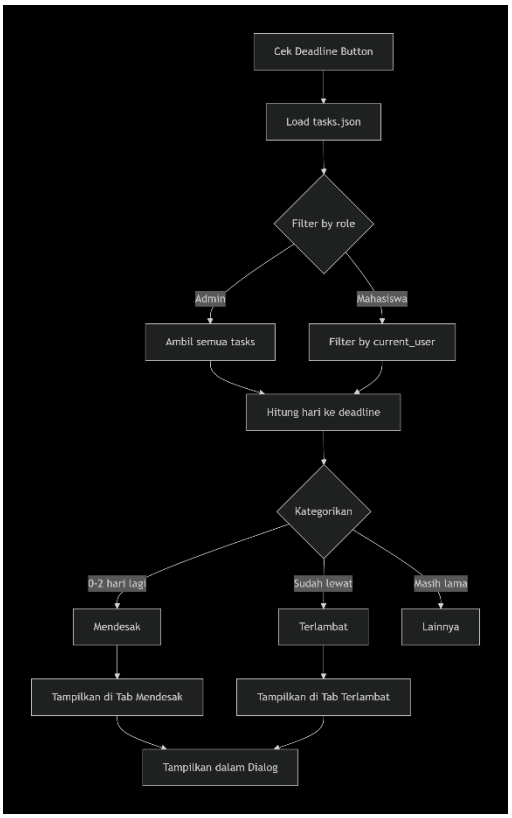
9. Alur Export to Excel



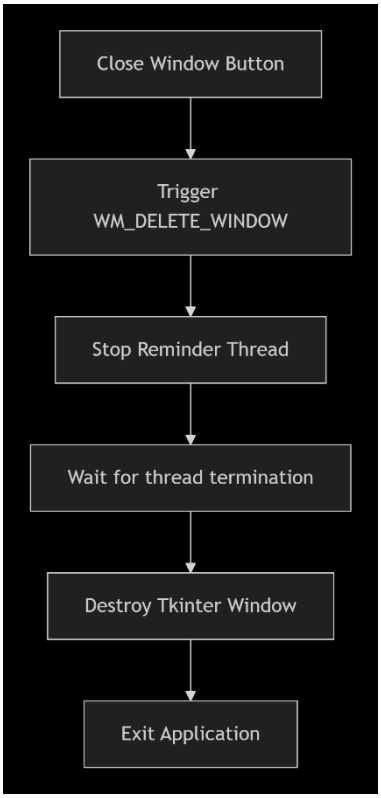
10. Alur Log Out



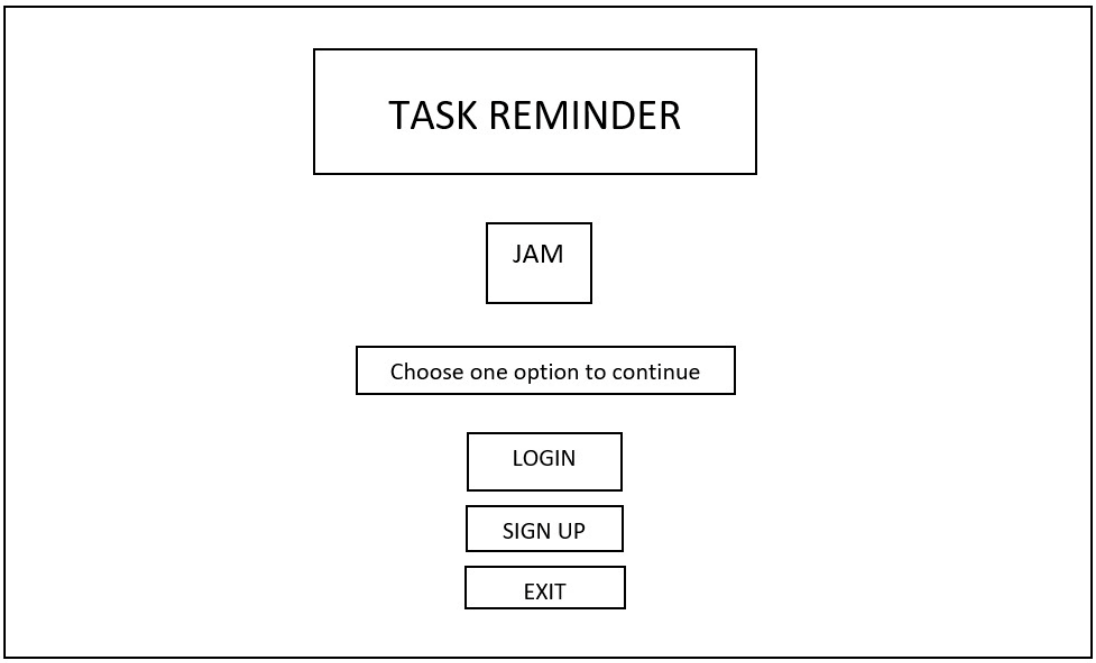
11. Alur Cek Deadline



12. Alur Shutdown



F. Sketsa Desain Antarmuka



IMPLEMENTASI

```python
def show_welcome_screen(self):
    self.clear_window()

    welcome_label = tk.Label(self.root, text="TASK REMINDER", font=("Berlin Sans FB Demi", 50, "bold"), bg=self.bg_color, fg="#2c5c92")
    welcome_label.pack(pady=50)

    welcome_label = tk.Label(self.root, text="", font=("Berlin Sans FB Demi", 60, "bold"), bg=self.bg_color, fg="#0B8AC0")
    welcome_label.pack(pady=10)

    subtitle_label = tk.Label(self.root, text="Choose one option to continue :", font=("Berlin Sans FB Demi", 16, "bold"), bg=self.bg_color, fg="#2c5c92")
    subtitle_label.pack(pady=20)

    button_frame = tk.Frame(self.root, bg=self.bg_color)
    button_frame.pack(pady=50)

    login_button = tk.Button(button_frame, text="LOGIN", font=("Times New Roman", 14, "bold"), bg=self.button_color, fg="white", width=20, height=2, cursor="hand2", command=self.show_login_screen)
    login_button.pack(pady=5)

    signup_button = tk.Button(button_frame, text="SIGN UP", font=("Times New Roman", 14, "bold"), bg="#2c5c92", fg="white", width=20, height=2, cursor="hand2", command=self.show_signup_screen)
    signup_button.pack(pady=5)

    exit_button = tk.Button(button_frame, text="EXIT", font=("Times New Roman", 14), bg="#f44336", fg="white", width=22, height=2, cursor="hand2", command=self.root.quit)
    exit_button.pack(pady=5)
```

- Menampilkan halaman awal Task Reminder
- Menampilkan judul, ikon, dan petunjuk
- Menyediakan tombol LOGIN, SIGN UP, dan EXIT

```python
def show_admin_dashboard(self):
    self.clear_window()

    header_frame = tk.Frame(self.root, bg=self.admin_color, height=80)
    header_frame.pack(fill=tk.X)
    header_frame.pack_propagate(False)

    tk.Label(header_frame, text=f"ADMIN DASHBOARD\nWelcome, {self.current_u

    logout_button = tk.Button(header_frame, text="Logout", font=("Times New
    logout_button.place(x=10, y=20)

    self.show_admin_stats()

    menu_frame = tk.Frame(self.root, bg=self.bg_color)
```

- Menampilkan Dashboard Admin
- Menyediakan menu manajemen aplikasi
- Mengatur tombol secara rapi dan otomatis
- Memudahkan admin mengelola Task Reminder

```python
def check_automatic_reminders(self):
    try:
        tasks = self.load_json(self.tasks_file)
        today = datetime.now().date()

        for task in tasks:
            if self.current_role == 'mahasiswa' and task['username_pemilik'] != self.current_user:
                continue

            if task.get('status', 'Pending') == 'Completed':
                continue

            try:
                deadline_date = datetime.strptime(str(task['deadline']), '%Y-%m-%d').date()
                days_left = (deadline_date - today).days

                if days_left == 0 and not task.get('notified_today', False):
                    self.send_notification("⏰ Deadline Hari Ini!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")
                    task['notified_today'] = True
                    self.update_task_in_json(task)
                elif days_left == 1 and not task.get('notified_tomorrow', False):
                    self.send_notification("⏰ Deadline Besok!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")
                    task['notified_tomorrow'] = True
                    self.update_task_in_json(task)
                elif days_left < 0 and not task.get('notified_late', False):
                    self.send_notification("⚠️ Tugas Terlambat!", f"Tugas: {task['judul']}\nTerlambat: {abs(days_left)} hari")
                    task['notified_late'] = True
                    self.update_task_in_json(task)
            except:
                continue
    except Exception as e:
        print(f"Error checking reminders: {e}")

def send_notification(self, title, message):
    try:
        notification.notify(title=title, message=message, app_name="Task Reminder", timeout=10)
```

- Mengecek deadline otomatis
- Menyesuaikan dengan role user
- Menghindari notifikasi ganda
- Memberi peringatan hari ini, besok, dan terlambat
- Menggunakan JSON + notifikasi sistem

LAMPIRAN

TaskReminderApp1.py

C:\Users\bagum\Downloads\REMINDER TUGAS\TaskReminderApp1.py (preview)

```python
import tkinter as tk
from tkinter import ttk, messagebox, simpledialog
import json
import os
import hashlib
from datetime import datetime
import threading
import time
import pandas as pd
from plyer import notification

class TaskReminderApp:
    def __init__(self, root):
        self.root = root
        self.root.title("TASK REMINDER")
        self.root.geometry("1000x700")

        self.users_file = 'users.json'
        self.tasks_file = 'tasks.json'
        self.backup_users_file = 'users_backup.xlsx'
        self.backup_tasks_file = 'tasks_backup.xlsx'

        self.current_user = None
        self.current_role = None

        self.bg_color = "#f5f7da"
        self.button_color = "#DB8AC0"
        self.button_hover = "#eddea1"
        self.admin_color = "#2c5c92"
        self.student_color = "#2c5c92"

        self.root.configure(bg=self.bg_color)

        self.reminder_running = False
        self.reminder_thread = None

        self.initialize_files()
        self.show_welcome_screen()
```

```python
    def show_welcome_screen(self):
        self.clear_window()

        welcome_label = tk.Label(self.root, text="TASK REMINDER", font=("Berlin Sans FB Demi", 50, "bold"), bg=self.bg_color, fg="#2c5c92")
        welcome_label.pack(pady=50)

        welcome_label = tk.Label(self.root, text="", font=("Berlin Sans FB Demi", 60, "bold"), bg=self.bg_color, fg="#DB8AC0")
        welcome_label.pack(pady=10)

        subtitle_label = tk.Label(self.root, text="Choose one option to continue :", font=("Berlin Sans FB Demi", 16, "bold"), bg=self.bg_color, fg="#2c5c92")
        subtitle_label.pack(pady=20)

        button_frame = tk.Frame(self.root, bg=self.bg_color)
        button_frame.pack(pady=50)

        login_button = tk.Button(button_frame, text="LOGIN", font=("Times New Roman", 14, "bold"), bg=self.button_color, fg="white", width=20, height=2, cursor="hand2", command=self.show_login_screen)
        login_button.pack(pady=5)

        signup_button = tk.Button(button_frame, text="SIGN UP", font=("Times New Roman", 14, "bold"), bg="#2c5c92", fg="white", width=20, height=2, cursor="hand2", command=self.show_signup_screen)
        signup_button.pack(pady=5)

        exit_button = tk.Button(button_frame, text="EXIT", font=("Times New Roman", 14), bg="#f44336", fg="white", width=22, height=2, cursor="hand2", command=self.root.quit)
        exit_button.pack(pady=5)
```

```python
    def show_login_screen(self):
        self.clear_window()

        header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
        header_frame.pack(fill=tk.X)
        header_frame.pack_propagate(False)

        tk.Label(header_frame, text="LOGIN", font=("Times New Roman", 18, "bold"), bg=self.admin_color, fg="white").pack(pady=15)

        back_button = tk.Button(header_frame, text="← Kembali", font=("Times New Roman", 10), bg="white", fg=self.admin_color, bd=0, command=self.show_welcome_screen, cursor="hand2")
        back_button.place(x=10, y=15)

        form_frame = tk.Frame(self.root, bg=self.bg_color)
        form_frame.pack(pady=50)

        tk.Label(form_frame, text="Username:", font=("Times New Roman", 15), bg=self.bg_color).grid(row=0, column=0, padx=10, pady=10, sticky="w")
        self.username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50)
        self.username_entry.grid(row=0, column=1, padx=10, pady=35)

        tk.Label(form_frame, text="Password:", font=("Times New Roman", 15), bg=self.bg_color).grid(row=1, column=0, padx=10, pady=10, sticky="w")
        self.password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50, show="*")
        self.password_entry.grid(row=1, column=1, padx=10, pady=35)

        tk.Label(form_frame, text="Role:", font=("Times New Roman", 15), bg=self.bg_color).grid(row=2, column=0, padx=10, pady=10, sticky="w")
        self.role_var = tk.StringVar(value="mahasiswa")
        role_combo = ttk.Combobox(form_frame, textvariable=self.role_var, values=["mahasiswa", "admin"], width=48, font=("Times New Roman", 12), state="readonly")
        role_combo.grid(row=2, column=1, padx=10, pady=35)

        login_btn_frame = tk.Frame(self.root, bg=self.bg_color)
        login_btn_frame.pack(pady=30)

        login_button = tk.Button(login_btn_frame, text="LOGIN", font=("Times New Roman", 15, "bold"), bg=self.button_color, fg="white", width=20, cursor="hand2", command=self.login)
        login_button.pack()
```

```
    def show_signup_screen(self):
        self.clear_window()

        header_frame = tk.Frame(self.root, bg=self.student_color, height=60)
        header_frame.pack(fill=tk.X)
        header_frame.pack_propagate(False)

        tk.Label(header_frame, text="SIGN UP", font=("Times New Roman", 18, "bold"), bg=self.student_color, fg="white").pack(pady=15)
        back_button = tk.Button(header_frame, text="← Kembali", font=("Times New Roman", 10), bg="white", fg=self.student_color, bd=0, command=self.show_welcome_screen, cursor="hand2")
        back_button.place(x=10, y=15)

        form_frame = tk.Frame(self.root, bg=self.bg_color)
        form_frame.pack(pady=30)

        tk.Label(form_frame, text="Username:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=0, column=0, padx=10, pady=10, sticky="w")
        self.signup_username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50)
        self.signup_username_entry.grid(row=0, column=1, padx=10, pady=35)

        tk.Label(form_frame, text="Password:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=1, column=0, padx=10, pady=10, sticky="w")
        self.signup_password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50, show="*")
        self.signup_password_entry.grid(row=1, column=1, padx=10, pady=35)

        tk.Label(form_frame, text="Confirm Password:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=2, column=0, padx=10, pady=10, sticky="w")
        self.signup_confirm_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50, show="*")
        self.signup_confirm_entry.grid(row=2, column=1, padx=10, pady=35)

        signup_btn_frame = tk.Frame(self.root, bg=self.bg_color)
        signup_btn_frame.pack(pady=30)

        signup_button = tk.Button(signup_btn_frame, text="SIGN UP", font=("Times New Roman", 15, "bold"), bg=self.student_color, fg="white", width=20, cursor="hand2", command=self.signup)
        signup_button.pack()
```

```
    def show_admin_dashboard(self):
        self.clear_window()

        header_frame = tk.Frame(self.root, bg=self.admin_color, height=80)
        header_frame.pack(fill=tk.X)
        header_frame.pack_propagate(False)

        tk.Label(header_frame, text=f"ADMIN DASHBOARD\nWelcome, {self.current_user}!", font=("Times New Roman", 18, "bold"), bg=self.admin_color, fg="white").pack(pady=10)

        logout_button = tk.Button(header_frame, text="Logout", font=("Times New Roman", 10), bg="white", fg=self.admin_color, bd=0, command=self.logout, cursor="hand2")
        logout_button.place(x=10, y=20)

        self.show_admin_stats()

        menu_frame = tk.Frame(self.root, bg=self.bg_color)
        menu_frame.pack(pady=85)

        buttons = [
            ("📝 Kelola Tugas", self.manage_tasks_admin),
            ("🎓 Kelola Mahasiswa", self.manage_students),
            ("📋 Lihat Semua Tugas", self.view_all_tasks_gui),
            ("📊 Export to Excel", self.export_to_excel),
            ("⏰ Cek Deadline", self.check_deadline_gui),
            ("💾 Backup Data", self.create_backup_files)
        ]

        for i, (text, command) in enumerate(buttons):
            row, col = i // 3, i % 3
            btn = tk.Button(menu_frame, text=text, font=("Times New Roman", 12, "bold"), bg=self.button_color, fg="white", width=20, height=3, cursor="hand2", command=command)
            btn.grid(row=row, column=col, padx=20, pady=20)
```

```
    def show_student_dashboard(self):
        self.clear_window()

        header_frame = tk.Frame(self.root, bg=self.student_color, height=80)
        header_frame.pack(fill=tk.X)
        header_frame.pack_propagate(False)

        tk.Label(header_frame, text=f"STUDENT DASHBOARD\nWelcome, {self.current_user}!", font=("Times New Roman", 18, "bold"), bg=self.student_color, fg="white").pack(pady=10)

        logout_button = tk.Button(header_frame, text="Logout", font=("Times New Roman", 10), bg="white", fg=self.student_color, bd=0, command=self.logout, cursor="hand2")
        logout_button.place(x=10, y=20)

        self.show_student_stats()

        menu_frame = tk.Frame(self.root, bg=self.bg_color)
        menu_frame.pack(pady=85)

        buttons = [
            ("➕ Tambah Tugas", self.add_task_dialog),
            ("📝 Lihat Tugas Saya", self.view_my_tasks_gui),
            ("⏰ Cek Deadline", self.check_deadline_gui),
            ("📊 Export Tugas Saya", self.export_to_excel),
            ("🔄 Refresh", lambda: self.show_student_dashboard())
        ]

        for i, (text, command) in enumerate(buttons):
            row, col = i // 3, i % 3
            btn = tk.Button(menu_frame, text=text, font=("Times New Roman", 12, "bold"), bg=self.button_color, fg="white", width=20, height=3, cursor="hand2", command=command)
            btn.grid(row=row, column=col, padx=20, pady=20)
```

```
    def initialize_files(self):
        if not os.path.exists(self.users_file):
            users_data = [
                {'username': 'admin', 'password': self.hash_password('admin123'), 'role': 'admin'},
                {'username': 'mahasiswa1', 'password': self.hash_password('mhs123'), 'role': 'mahasiswa'},
                {'username': 'mahasiswa2', 'password': self.hash_password('mhs123'), 'role': 'mahasiswa'}
            ]
            self.save_json(self.users_file, users_data)

        if not os.path.exists(self.tasks_file):
            self.save_json(self.tasks_file, [])

        self.create_backup_files()

    def save_json(self, filename, data):
        try:
            with open(filename, 'w', encoding='utf-8') as f:
                json.dump(data, f, indent=4, ensure_ascii=False)
        except Exception as e:
            print(f"Error saving {filename}: {e}")

    def load_json(self, filename):
        try:
            if os.path.exists(filename):
                with open(filename, 'r', encoding='utf-8') as f:
                    return json.load(f)
            return []
        except Exception as e:
            print(f"Error loading {filename}: {e}")
            return []

    def create_backup_files(self):
        try:
            users = self.load_json(self.users_file)
            if users:
                pd.DataFrame(users).to_excel(self.backup_users_file, index=False)

            tasks = self.load_json(self.tasks_file)
            if tasks:
                pd.DataFrame(tasks).to_excel(self.backup_tasks_file, index=False)
        except Exception as e:
            print(f"Error creating backup: {e}")

    def hash_password(self, password):
        return hashlib.sha256(password.encode()).hexdigest()

    def clear_window(self):
        for widget in self.root.winfo_children():
            widget.destroy()
```

```
    def start_reminder_system(self):
        if self.reminder_running:
            return

        self.reminder_running = True
        self.reminder_thread = threading.Thread(target=self.reminder_check_loop, daemon=True)
        self.reminder_thread.start()

    def stop_reminder_system(self):
        self.reminder_running = False
        if self.reminder_thread:
            self.reminder_thread.join(timeout=1)

    def reminder_check_loop(self):
        while self.reminder_running:
            try:
                self.check_automatic_reminders()
                time.sleep(30)
            except Exception as e:
                print(f"Error in reminder loop: {e}")
                time.sleep(30)

    def check_automatic_reminders(self):
        try:
            tasks = self.load_json(self.tasks_file)
            today = datetime.now().date()
```

```
            for task in tasks:
                if self.current_role == 'mahasiswa' and task['username_pemilik'] != self.current_user:
                    continue

                if task.get('status', 'Pending') == 'Completed':
                    continue

                try:
                    deadline_date = datetime.strptime(str(task['deadline']), '%Y-%m-%d').date()
                    days_left = (deadline_date - today).days

                    if days_left == 0 and not task.get('notified_today', False):
                        self.send_notification("⏰ Deadline Hari Ini!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")
                        task['notified_today'] = True
                        self.update_task_in_json(task)
                    elif days_left == 1 and not task.get('notified_tomorrow', False):
                        self.send_notification("📅 Deadline Besok!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")
                        task['notified_tomorrow'] = True
                        self.update_task_in_json(task)
                    elif days_left < 0 and not task.get('notified_late', False):
                        self.send_notification("⚠️ Tugas Terlambat!", f"Tugas: {task['judul']}\nTerlambat: {abs(days_left)} hari")
                        task['notified_late'] = True
                        self.update_task_in_json(task)
                except:
                    continue
        except Exception as e:
            print(f"Error checking reminders: {e}")
```

```
    def send_notification(self, title, message):
        try:
            notification.notify(title=title, message=message, app_name="Task Reminder", timeout=10)
        except:
            if self.root and tk.Toplevel:
                self.root.after(0, lambda: messagebox.showwarning(title, message))

    def update_task_in_json(self, updated_task):
        try:
            tasks = self.load_json(self.tasks_file)
            for i, task in enumerate(tasks):
                if task['id_tugas'] == updated_task['id_tugas']:
                    tasks[i] = updated_task
                    break
            self.save_json(self.tasks_file, tasks)
        except Exception as e:
            print(f"Error updating task in JSON: {e}")
```

```
    def signup(self):
        username = self.signup_username_entry.get().strip()
        password = self.signup_password_entry.get().strip()
        confirm_password = self.signup_confirm_entry.get().strip()

        if not all([username, password, confirm_password]):
            messagebox.showerror("Sign Up Error", "Semua field harus diisi!")
            return

        if len(username) < 3:
            messagebox.showerror("Sign Up Error", "Username minimal 3 karakter!")
            return

        if len(password) < 6:
            messagebox.showerror("Sign Up Error", "Password minimal 6 karakter!")
            return

        if password != confirm_password:
            messagebox.showerror("Sign Up Error", "Password tidak cocok!")
            return

        try:
            users = self.load_json(self.users_file)
            usernames = [user['username'] for user in users]

            if username in usernames:
                messagebox.showerror("Sign Up Error", "Username sudah digunakan!")
                return

            new_user = {'username': username, 'password': self.hash_password(password), 'role': 'mahasiswa'}
            users.append(new_user)
            self.save_json(self.users_file, users)

            self.create_backup_files()

            messagebox.showinfo("Sign Up Success", f"Akun {username} berhasil dibuat!\nSilakan login dengan akun Anda.")
            self.show_login_screen()

        except Exception as e:
            messagebox.showerror("Sign Up Error", f"Error saat mendaftar: {str(e)}")
```

```
    def login(self):
        username = self.username_entry.get()
        password = self.password_entry.get()
        role = self.role_var.get()

        try:
            users = self.load_json(self.users_file)
            user_found = None
            for user in users:
                if user['username'] == username and user['role'] == role:
                    user_found = user
                    break

            if user_found:
                if self.hash_password(password) == user_found['password']:
                    self.current_user = username
                    self.current_role = role
                    self.start_reminder_system()

                    if role == 'admin':
                        self.show_admin_dashboard()
                    else:
                        self.show_student_dashboard()
                else:
                    messagebox.showerror("Login Error", "Password salah!")
            else:
                messagebox.showerror("Login Error", "Username atau role tidak ditemukan!")

        except Exception as e:
            messagebox.showerror("Error", f"Error saat login: {str(e)}")

    def get_next_task_id(self):
        tasks = self.load_json(self.tasks_file)
        if not tasks:
            return 1
        max_id = max(task['id_tugas'] for task in tasks)
        return max_id + 1
```

```
    def add_task_dialog(self):
        dialog = tk.Toplevel(self.root)
        dialog.title("Tambah Tugas Baru")
        dialog.geometry("500x500")
        dialog.configure(bg=self.bg_color)
        dialog.transient(self.root)
        dialog.grab_set()

        tk.Label(dialog, text="TAMBAH TUGAS BARU", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)

        form_frame = tk.Frame(dialog, bg=self.bg_color)
        form_frame.pack(pady=10)

        row = 0

        tk.Label(form_frame, text="Judul Tugas:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        title_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Mata Kuliah:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        course_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Deadline (YYYY-MM-DD):", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        deadline_entry.insert(0, datetime.now().strftime("%Y-%m-%d"))
        deadline_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Deskripsi:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=6)
        desc_text.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        button_frame = tk.Frame(dialog, bg=self.bg_color)
        button_frame.pack(pady=20)
```

```
        def save_task():
            title = title_entry.get().strip()
            course = course_entry.get().strip()
            deadline = deadline_entry.get().strip()
            description = desc_text.get("1.0", tk.END).strip()

            if not all([title, course, deadline]):
                messagebox.showerror("Error", "Semua field harus diisi!")
                return

            try:
                datetime.strptime(deadline, '%Y-%m-%d')
            except:
                messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
                return

            try:
                tasks = self.load_json(self.tasks_file)
                new_task = {
                    'id_tugas': self.get_next_task_id(),
                    'judul': title,
                    'deskripsi': description,
                    'mata_kuliah': course,
                    'deadline': deadline,
                    'username_pemilik': self.current_user,
                    'dibuat_oleh': 'mahasiswa',
                    'status': 'Pending',
                    'created_at': datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
                    'notified_today': False,
                    'notified_tomorrow': False,
                    'notified_late': False
                }
                tasks.append(new_task)
                self.save_json(self.tasks_file, tasks)
                self.create_backup_files()
                messagebox.showinfo("Success", "Tugas berhasil ditambahkan!")
                dialog.destroy()
                self.view_my_tasks_gui()
            except Exception as e:
                messagebox.showerror("Error", f"Error: {str(e)}")

        tk.Button(button_frame, text="Simpan", font=("Times New Roman", 12, "bold"), bg=self.button_color, fg="white", width=15, command=save_task, cursor="hand2").pack(side=tk.LEFT, padx=10)
        tk.Button(button_frame, text="Batal", font=("Times New Roman", 12), bg="#f44336", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)
```

```
    def add_task_dialog_admin(self):
        dialog = tk.Toplevel(self.root)
        dialog.title("Tambah Tugas Baru")
        dialog.geometry("500x500")
        dialog.configure(bg=self.bg_color)
        dialog.transient(self.root)
        dialog.grab_set()

        tk.Label(dialog, text="TAMBAH TUGAS BARU (ADMIN)", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)

        try:
            users = self.load_json(self.users_file)
            students = [user for user in users if user['role'] == 'mahasiswa']
            student_list = [student['username'] for student in students]
        except:
            student_list = []

        form_frame = tk.Frame(dialog, bg=self.bg_color)
        form_frame.pack(pady=10)

        row = 0
```

```
        tk.Label(form_frame, text="Mahasiswa:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        student_var = tk.StringVar(value=student_list[0] if student_list else "")
        student_combo = ttk.Combobox(form_frame, textvariable=student_var, values=student_list, width=27, font=("Times New Roman", 12))
        student_combo.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Judul Tugas:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        title_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Mata Kuliah:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        course_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Deadline (YYYY-MM-DD):", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        deadline_entry.insert(0, datetime.now().strftime("%Y-%m-%d"))
        deadline_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Deskripsi:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
        desc_text.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        button_frame = tk.Frame(dialog, bg=self.bg_color)
        button_frame.pack(pady=20)
```

```
        def save_task():
            student = student_var.get()
            title = title_entry.get().strip()
            course = course_entry.get().strip()
            deadline = deadline_entry.get().strip()
            description = desc_text.get("1.0", tk.END).strip()

            if not all([student, title, course, deadline]):
                messagebox.showerror("Error", "Semua field harus diisi!")
                return

            try:
                datetime.strptime(deadline, '%Y-%m-%d')
            except:
                messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
                return

            try:
                tasks = self.load_json(self.tasks_file)
                new_task = {
                    'id_tugas': self.get_next_task_id(),
                    'judul': title,
                    'deskripsi': description,
                    'mata_kuliah': course,
                    'deadline': deadline,
                    'username_pemilik': student,
                    'dibuat_oleh': 'admin',
                    'status': 'Pending',
                    'created_at': datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
                    'notified_today': False,
                    'notified_tomorrow': False,
                    'notified_late': False
                }
                tasks.append(new_task)
                self.save_json(self.tasks_file, tasks)
                self.create_backup_files()
                messagebox.showinfo("Success", "Tugas berhasil ditambahkan!")
                dialog.destroy()
                self.manage_tasks_admin()
            except Exception as e:
                messagebox.showerror("Error", f"Error: {str(e)}")

        tk.Button(button_frame, text="Simpan", font=("Times New Roman", 12, "bold"), bg=self.button_color, fg="white", width=15, command=save_task, cursor="hand2").pack(side=tk.LEFT, padx=10)
        tk.Button(button_frame, text="Batal", font=("Times New Roman", 12), bg="#f44336", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)
```

```
573     def show_admin_stats(self):
574         try:
575             tasks = self.load_json(self.tasks_file)
576             users = self.load_json(self.users_file)
577 
578             stats_frame = tk.Frame(self.root, bg=self.bg_color)
579             stats_frame.pack(pady=20)
580 
581             total_tasks = len(tasks)
582             total_students = len([user for user in users if user['role'] == 'mahasiswa'])
583             pending_tasks = len([task for task in tasks if task.get('status', 'Pending') == 'Pending'])
584             completed_tasks = len([task for task in tasks if task.get('status', 'Pending') == 'Completed'])
585 
586             stats = [
587                 f"Total Tugas: {total_tasks}",
588                 f"Total Mahasiswa: {total_students}",
589                 f"Tugas Pending: {pending_tasks}",
590                 f"Tugas Selesai: {completed_tasks}"
591             ]
592 
593             for i, stat in enumerate(stats):
594                 tk.Label(stats_frame, text=stat, font=("Times New Roman", 12, "bold"), bg=self.bg_color).grid(row=0, column=i, padx=20)
595         except Exception as e:
596             print(f"Error loading stats: {e}")
```

```
598     def show_student_stats(self):
599         try:
600             tasks = self.load_json(self.tasks_file)
601             my_tasks = [task for task in tasks if task['username_pemilik'] == self.current_user]
602 
603             stats_frame = tk.Frame(self.root, bg=self.bg_color)
604             stats_frame.pack(pady=20)
605 
606             total_tasks = len(my_tasks)
607             pending_tasks = len([task for task in my_tasks if task.get('status', 'Pending') == 'Pending'])
608             completed_tasks = len([task for task in my_tasks if task.get('status', 'Pending') == 'Completed'])
609 
610             today = datetime.now().date()
611             urgent_tasks = 0
612             for task in my_tasks:
613                 try:
614                     if task.get('status', 'Pending') != 'Completed':
615                         deadline_date = datetime.strptime(str(task['deadline']), '%Y-%m-%d').date()
616                         days_left = (deadline_date - today).days
617                         if 0 <= days_left <= 2:
618                             urgent_tasks += 1
619                 except:
620                     continue
621 
622             stats = [
623                 f"Total Tugas: {total_tasks}",
624                 f"Tugas Pending: {pending_tasks}",
625                 f"Tugas Selesai: {completed_tasks}",
626                 f"Tugas Mendesak: {urgent_tasks}"
627             ]
628 
629             for i, stat in enumerate(stats):
630                 tk.Label(stats_frame, text=stat, font=("Times New Roman", 12, "bold"), bg=self.bg_color).grid(row=0, column=i, padx=20)
631         except Exception as e:
632             print(f"Error loading stats: {e}")
```

```
    def manage_students(self):
        self.clear_window()

        header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
        header_frame.pack(fill=tk.X)
        header_frame.pack_propagate(False)

        tk.Label(header_frame, text="KELOLA DATA MAHASISWA", font=("Times New Roman", 16, "bold"), bg=self.admin_color, fg="white").pack(pady=15)

        back_button = tk.Button(header_frame, text="← Kembali", font=("Times New Roman", 10), bg="white", fg=self.admin_color, bd=0, command=self.show_admin_dashboard, cursor="hand2")
        back_button.place(x=10, y=15)

        control_frame = tk.Frame(self.root, bg=self.bg_color)
        control_frame.pack(pady=10)

        tk.Button(control_frame, text="Tambah Mahasiswa", font=("Times New Roman", 11, "bold"), bg=self.button_color, fg="white", width=20, command=self.add_student_dialog, cursor="hand2").pack(side=tk.LEFT, padx=5)
        tk.Button(control_frame, text="Refresh", font=("Times New Roman", 11, "bold"), bg="#607D8B", fg="white", width=20, command=self.manage_students, cursor="hand2").pack(side=tk.LEFT, padx=5)

        table_frame = tk.Frame(self.root)
        table_frame.pack(pady=10, padx=20, fill=tk.BOTH, expand=True)

        columns = ('Username', 'Role', 'Actions')
        tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)

        tree.heading('Username', text='Username')
        tree.heading('Role', text='Role')
        tree.heading('Actions', text='Actions')

        tree.column('Username', width=200)
        tree.column('Role', width=100)
        tree.column('Actions', width=150)

        scrollbar = ttk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
        tree.configure(yscroll=scrollbar.set)

        tree.grid(row=0, column=0, sticky='nsew')
        scrollbar.grid(row=0, column=1, sticky='ns')

        table_frame.grid_rowconfigure(0, weight=1)
        table_frame.grid_columnconfigure(0, weight=1)

        try:
            users = self.load_json(self.users_file)
            students = [user for user in users if user['role'] == 'mahasiswa']

            for student in students:
                tree.insert('', tk.END, values=(student['username'], student['role'], 'delete atau edit'))
        except Exception as e:
            messagebox.showerror("Error", f"Error loading data: {str(e)}")

        tree.bind('<Double-1>', lambda e: self.edit_student_dialog(tree))
```

```
    def manage_tasks_admin(self):
        self.clear_window()

        header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
        header_frame.pack(fill=tk.X)
        header_frame.pack_propagate(False)

        tk.Label(header_frame, text="KELOLA TUGAS (ADMIN)", font=("Times New Roman", 16, "bold"), bg=self.admin_color, fg="white").pack(pady=15)

        back_button = tk.Button(header_frame, text="← Kembali", font=("Times New Roman", 10), bg="white", fg=self.admin_color, bd=0, command=self.show_admin_dashboard, cursor="hand2")
        back_button.place(x=10, y=15)

        control_frame = tk.Frame(self.root, bg=self.bg_color)
        control_frame.pack(pady=10)

        tk.Button(control_frame, text="Tambah Tugas", font=("Times New Roman", 11, "bold"), bg=self.button_color, fg="white", width=20, command=self.add_task_dialog_admin, cursor="hand2").pack(side=tk.LEFT, padx=5)
        tk.Button(control_frame, text="Refresh", font=("Times New Roman", 11, "bold"), bg="#607D8B", fg="white", width=20, command=self.manage_tasks_admin, cursor="hand2").pack(side=tk.LEFT, padx=5)

        table_frame = tk.Frame(self.root)
        table_frame.pack(pady=10, padx=20, fill=tk.BOTH, expand=True)

        columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Pemilik', 'Status', 'Dibuat Oleh', 'Actions')
        tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)

        tree.heading('ID', text='ID')
        tree.heading('Judul', text='Judul')
        tree.heading('Mata Kuliah', text='Mata Kuliah')
        tree.heading('Deadline', text='Deadline')
        tree.heading('Pemilik', text='Pemilik')
        tree.heading('Status', text='Status')
        tree.heading('Dibuat Oleh', text='Dibuat Oleh')
        tree.heading('Actions', text='Actions')

        tree.column('ID', width=50)
        tree.column('Judul', width=150)
        tree.column('Mata Kuliah', width=120)
        tree.column('Deadline', width=100)
        tree.column('Pemilik', width=100)
        tree.column('Status', width=80)
        tree.column('Dibuat Oleh', width=100)
        tree.column('Actions', width=100)

        scrollbar = ttk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
        tree.configure(yscroll=scrollbar.set)

        tree.grid(row=0, column=0, sticky='nsew')
        scrollbar.grid(row=0, column=1, sticky='ns')

        table_frame.grid_rowconfigure(0, weight=1)
        table_frame.grid_columnconfigure(0, weight=1)
```

```
        try:
            tasks = self.load_json(self.tasks_file)

            for task in tasks:
                status = task.get('status', 'Pending')
                tree.insert('', tk.END, values=(
                    task['id_tugas'],
                    task['judul'],
                    task['mata_kuliah'],
                    task['deadline'],
                    task['username_pemilik'],
                    status,
                    task['dibuat_oleh'],
                    'Edit'
                ))
        except Exception as e:
            messagebox.showerror("Error", f"Error loading data: {str(e)}")

        tree.bind('<Double-1>', lambda e: self.edit_task_dialog_admin(tree))
```

```
    def view_all_tasks_gui(self):
        self.clear_window()

        header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
        header_frame.pack(fill=tk.X)
        header_frame.pack_propagate(False)

        tk.Label(header_frame, text="SEMUA TUGAS", font=("Times New Roman", 16, "bold"), bg=self.admin_color, fg="white").pack(pady=15)

        back_button = tk.Button(header_frame, text="← Kembali", font=("Times New Roman", 10), bg="white", fg=self.admin_color, bd=0, command=self.show_admin_dashboard, cursor="hand2")
        back_button.place(x=10, y=15)

        table_frame = tk.Frame(self.root)
        table_frame.pack(pady=10, padx=20, fill=tk.BOTH, expand=True)

        columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Pemilik', 'Status', 'Dibuat Oleh')
        tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=20)

        tree.heading('ID', text='ID')
        tree.heading('Judul', text='Judul')
        tree.heading('Mata Kuliah', text='Mata Kuliah')
        tree.heading('Deadline', text='Deadline')
        tree.heading('Pemilik', text='Pemilik')
        tree.heading('Status', text='Status')
        tree.heading('Dibuat Oleh', text='Dibuat Oleh')

        tree.column('ID', width=50)
        tree.column('Judul', width=200)
        tree.column('Mata Kuliah', width=150)
        tree.column('Deadline', width=100)
        tree.column('Pemilik', width=100)
        tree.column('Status', width=100)
        tree.column('Dibuat Oleh', width=100)

        scrollbar = ttk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
        tree.configure(yscroll=scrollbar.set)

        tree.grid(row=0, column=0, sticky='nsew')
        scrollbar.grid(row=0, column=1, sticky='ns')

        table_frame.grid_rowconfigure(0, weight=1)
        table_frame.grid_columnconfigure(0, weight=1)
```

```
        try:
            tasks = self.load_json(self.tasks_file)

            for task in tasks:
                tree.insert('', tk.END, values=(
                    task['id_tugas'],
                    task['judul'],
                    task['mata_kuliah'],
                    task['deadline'],
                    task['username_pemilik'],
                    task.get('status', 'Pending'),
                    task['dibuat_oleh']
                ))
        except Exception as e:
            messagebox.showerror("Error", f"Error loading data: {str(e)}")
```

```
    def view_my_tasks_gui(self):
        self.clear_window()

        header_frame = tk.Frame(self.root, bg=self.student_color, height=60)
        header_frame.pack(fill=tk.X)
        header_frame.pack_propagate(False)

        tk.Label(header_frame, text="TUGAS SAYA", font=("Times New Roman", 16, "bold"), bg=self.student_color, fg="white").pack(pady=15)

        back_button = tk.Button(header_frame, text="← Kembali", font=("Times New Roman", 10), bg="white", fg=self.student_color, bd=0, command=self.show_student_dashboard, cursor="hand2")
        back_button.place(x=10, y=15)

        control_frame = tk.Frame(self.root, bg=self.bg_color)
        control_frame.pack(pady=10)

        tk.Button(control_frame, text="Refresh", font=("Times New Roman", 11, "bold"), bg="#607D8B", fg="white", width=20, command=self.view_my_tasks_gui, cursor="hand2").pack(side=tk.LEFT, padx=5)

        table_frame = tk.Frame(self.root)
        table_frame.pack(pady=10, padx=20, fill=tk.BOTH, expand=True)

        columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Status', 'Dibuat Oleh', 'Actions')
        tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)

        tree.heading('ID', text='ID')
        tree.heading('Judul', text='Judul')
        tree.heading('Mata Kuliah', text='Mata Kuliah')
        tree.heading('Deadline', text='Deadline')
        tree.heading('Status', text='Status')
        tree.heading('Dibuat Oleh', text='Dibuat Oleh')
        tree.heading('Actions', text='Actions')

        tree.column('ID', width=50)
        tree.column('Judul', width=200)
        tree.column('Mata Kuliah', width=150)
        tree.column('Deadline', width=100)
        tree.column('Status', width=100)
        tree.column('Dibuat Oleh', width=100)
        tree.column('Actions', width=100)

        scrollbar = ttk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
        tree.configure(yscroll=scrollbar.set)

        tree.grid(row=0, column=0, sticky='nsew')
        scrollbar.grid(row=0, column=1, sticky='ns')

        table_frame.grid_rowconfigure(0, weight=1)
        table_frame.grid_columnconfigure(0, weight=1)
```

```
        try:
            tasks = self.load_json(self.tasks_file)
            my_tasks = [task for task in tasks if task['username_pemilik'] == self.current_user]

            for task in my_tasks:
                actions = 'Edit | Delete' if task['dibuat_oleh'] == 'mahasiswa' else 'View Only'
                tree.insert('', tk.END, values=(
                    task['id_tugas'],
                    task['judul'],
                    task['mata_kuliah'],
                    task['deadline'],
                    task.get('status', 'Pending'),
                    task['dibuat_oleh'],
                    actions
                ))
        except Exception as e:
            messagebox.showerror("Error", f"Error loading data: {str(e)}")

        tree.bind('<Double-1>', lambda e: self.view_task_details_gui(tree))
```

```
    def check_deadline_gui(self):
        try:
            tasks = self.load_json(self.tasks_file)
            today = datetime.now().date()

            urgent_tasks = []
            late_tasks = []

            for task in tasks:
                if self.current_role == 'mahasiswa' and task['username_pemilik'] != self.current_user:
                    continue

                try:
                    deadline_date = datetime.strptime(str(task['deadline']), '%Y-%m-%d').date()
                    days_left = (deadline_date - today).days

                    if 0 <= days_left <= 2:
                        urgent_tasks.append({
                            'id': task['id_tugas'],
                            'judul': task['judul'],
                            'mata_kuliah': task['mata_kuliah'],
                            'deadline': task['deadline'],
                            'days_left': days_left,
                            'pemilik': task['username_pemilik'],
                            'status': task.get('status', 'Pending')
                        })
                    elif days_left < 0 and task.get('status', 'Pending') != 'Completed':
                        late_tasks.append({
                            'id': task['id_tugas'],
                            'judul': task['judul'],
                            'mata_kuliah': task['mata_kuliah'],
                            'deadline': task['deadline'],
                            'days_left': days_left,
                            'pemilik': task['username_pemilik'],
                            'status': task.get('status', 'Pending')
                        })
                except:
                    continue

            dialog = tk.Toplevel(self.root)
            dialog.title("Pengingat Deadline")
            dialog.geometry("600x500")
            dialog.configure(bg=self.bg_color)
            dialog.transient(self.root)

            tk.Label(dialog, text="⏰ PENGINGAT DEADLINE", font=("Times New Roman", 18, "bold"), bg=self.bg_color).pack(pady=20)

            notebook = ttk.Notebook(dialog)
            notebook.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)
```

```
            urgent_frame = tk.Frame(notebook, bg=self.bg_color)
            notebook.add(urgent_frame, text=f"Tugas Mendesak ({len(urgent_tasks)})")

            if urgent_tasks:
                columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Hari Lagi', 'Status')
                tree_urgent = ttk.Treeview(urgent_frame, columns=columns, show='headings', height=10)

                for col in columns:
                    tree_urgent.heading(col, text=col)
                    tree_urgent.column(col, width=100)

                tree_urgent.column('Judul', width=150)
                tree_urgent.column('Mata Kuliah', width=120)

                scrollbar_urgent = ttk.Scrollbar(urgent_frame, orient=tk.VERTICAL, command=tree_urgent.yview)
                tree_urgent.configure(yscroll=scrollbar_urgent.set)

                tree_urgent.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
                scrollbar_urgent.pack(side=tk.RIGHT, fill=tk.Y)

                for task in urgent_tasks:
                    days_text = "HARI INI!" if task['days_left'] == 0 else f"{task['days_left']} hari"
                    tree_urgent.insert('', tk.END, values=(
                        task['id'],
                        task['judul'],
                        task['mata_kuliah'],
                        task['deadline'],
                        days_text,
                        task['status']
                    ))
            else:
                tk.Label(urgent_frame, text="Tidak ada tugas mendesak (≤ 2 hari)", font=("Times New Roman", 14), bg=self.bg_color).pack(pady=50)

            late_frame = tk.Frame(notebook, bg=self.bg_color)
            notebook.add(late_frame, text=f"Tugas Terlambat ({len(late_tasks)})")
```

```
            if late_tasks:
                columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Terlambat', 'Status')
                tree_late = ttk.Treeview(late_frame, columns=columns, show='headings', height=10)

                for col in columns:
                    tree_late.heading(col, text=col)
                    tree_late.column(col, width=100)

                tree_late.column('Judul', width=150)
                tree_late.column('Mata Kuliah', width=120)

                scrollbar_late = ttk.Scrollbar(late_frame, orient=tk.VERTICAL, command=tree_late.yview)
                tree_late.configure(yscroll=scrollbar_late.set)

                tree_late.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
                scrollbar_late.pack(side=tk.RIGHT, fill=tk.Y)

                for task in late_tasks:
                    tree_late.insert('', tk.END, values=(
                        task['id'],
                        task['judul'],
                        task['mata_kuliah'],
                        task['deadline'],
                        f"{abs(task['days_left'])} hari",
                        task['status']
                    ))
            else:
                tk.Label(late_frame, text="Tidak ada tugas terlambat", font=("Times New Roman", 14), bg=self.bg_color).pack(pady=50)

            tk.Button(dialog, text="Tutup", font=("Times New Roman", 12), bg="#607D8B", fg="white", width=20, command=dialog.destroy, cursor="hand2").pack(pady=20)

        except Exception as e:
            messagebox.showerror("Error", f"Error: {str(e)}")
```

```
    def view_task_details_gui(self, tree):
        try:
            selected_item = tree.selection()
            if not selected_item:
                return

            item = tree.item(selected_item[0])
            task_id = item['values'][0]

            tasks = self.load_json(self.tasks_file)
            task = None
            for t in tasks:
                if t['id_tugas'] == task_id:
                    task = t
                    break

            if not task:
                messagebox.showerror("Error", "Tugas tidak ditemukan!")
                return

            dialog = tk.Toplevel(self.root)
            dialog.title(f"Detail Tugas #{task_id}")
            dialog.geometry("600x500")
            dialog.configure(bg=self.bg_color)
            dialog.transient(self.root)

            tk.Label(dialog, text=f"📋 DETAIL TUGAS #{task_id}", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)

            notebook = ttk.Notebook(dialog)
            notebook.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)

            info_frame = tk.Frame(notebook, bg=self.bg_color)
            notebook.add(info_frame, text="Informasi Dasar")

            details = [
                f"Judul: {task['judul']}",
                f"Mata Kuliah: {task['mata_kuliah']}",
                f"Deadline: {task['deadline']}",
                f"Pemilik: {task['username_pemilik']}",
                f"Dibuat Oleh: {task['dibuat_oleh']}",
                f"Status: {task.get('status', 'Pending')}",
                f"Dibuat Pada: {task.get('created_at', 'N/A')}"
            ]
```

```
            for i, detail in enumerate(details):
                tk.Label(info_frame, text=detail, font=("Times New Roman", 12), bg=self.bg_color, anchor="w").pack(pady=5, padx=20, fill=tk.X)

            desc_frame = tk.Frame(notebook, bg=self.bg_color)
            notebook.add(desc_frame, text="Deskripsi")

            desc_text = tk.Text(desc_frame, font=("Times New Roman", 11), wrap=tk.WORD, height=15)
            desc_scrollbar = tk.Scrollbar(desc_frame, command=desc_text.yview)
            desc_text.configure(yscrollcommand=desc_scrollbar.set)

            desc_text.pack(side=tk.LEFT, fill=tk.BOTH, expand=True, padx=10, pady=10)
            desc_scrollbar.pack(side=tk.RIGHT, fill=tk.Y, pady=10)

            desc_text.insert("1.0", task.get('deskripsi', 'Tidak ada deskripsi'))
            desc_text.configure(state='disabled')

            button_frame = tk.Frame(dialog, bg=self.bg_color)
            button_frame.pack(pady=20)

            user_can_edit = (
                (self.current_role == 'admin') or
                (self.current_role == 'mahasiswa' and task['dibuat_oleh'] == 'mahasiswa')
            )
```

```
            if user_can_edit:
                def edit_task():
                    dialog.destroy()
                    if self.current_role == 'admin':
                        self.edit_task_dialog_admin_by_id(task_id)
                    else:
                        self.edit_task_dialog_student(task_id)

                def delete_task():
                    if messagebox.askyesno("Konfirmasi", "Apakah Anda yakin ingin menghapus tugas ini?"):
                        success = self.delete_task(task_id)
                        if success:
                            dialog.destroy()
                            if self.current_role == 'admin':
                                self.manage_tasks_admin()
                            else:
                                self.view_my_tasks_gui()

                tk.Button(button_frame, text="✏️ Edit", font=("Times New Roman", 11), bg=self.button_color, fg="white", width=15, command=edit_task, cursor="hand2").pack(side=tk.LEFT, padx=10)
                tk.Button(button_frame, text="🗑️ Hapus", font=("Times New Roman", 11), bg="#f44336", fg="white", width=15, command=delete_task, cursor="hand2").pack(side=tk.LEFT, padx=10)

            tk.Button(button_frame, text="Tutup", font=("Times New Roman", 11), bg="#607D8B", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)

        except Exception as e:
            messagebox.showerror("Error", f"Error menampilkan detail: {str(e)}")
```

```
    def delete_task(self, task_id):
        try:
            tasks = self.load_json(self.tasks_file)
            new_tasks = [task for task in tasks if task['id_tugas'] != task_id]

            if len(new_tasks) < len(tasks):
                self.save_json(self.tasks_file, new_tasks)
                messagebox.showinfo("Success", "Tugas berhasil dihapus!")
                return True
            else:
                messagebox.showerror("Error", "Tugas tidak ditemukan!")
                return False
        except Exception as e:
            messagebox.showerror("Error", f"Error menghapus tugas: {str(e)}")
            return False
```

```python
def edit_task_dialog_admin_by_id(self, task_id):
    try:
        tasks = self.load_json(self.tasks_file)
        task = None
        for t in tasks:
            if t['id_tugas'] == task_id:
                task = t
                break

        if not task:
            messagebox.showerror("Error", "Tugas tidak ditemukan!")
            return

        users = self.load_json(self.users_file)
        students = [user for user in users if user['role'] == 'mahasiswa']
        student_list = [student['username'] for student in students]

        dialog = tk.Toplevel(self.root)
        dialog.title(f"Edit Tugas #{task_id}")
        dialog.geometry("500x500")
        dialog.configure(bg=self.bg_color)
        dialog.transient(self.root)
        dialog.grab_set()

        tk.Label(dialog, text=f"EDIT TUGAS #{task_id}", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)

        form_frame = tk.Frame(dialog, bg=self.bg_color)
        form_frame.pack(pady=10)

        row = 0

        tk.Label(form_frame, text="Mahasiswa:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        student_var = tk.StringVar(value=task['username_pemilik'])
        student_combo = ttk.Combobox(form_frame, textvariable=student_var, values=student_list, width=27, font=("Times New Roman", 12))
        student_combo.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Judul Tugas:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        title_entry.insert(0, task['judul'])
        title_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Mata Kuliah:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        course_entry.insert(0, task['mata_kuliah'])
        course_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1
```

```python
        tk.Label(form_frame, text="Deadline (YYYY-MM-DD):", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        deadline_entry.insert(0, task['deadline'])
        deadline_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Status:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        status_var = tk.StringVar(value=task.get('status', 'Pending'))
        status_combo = ttk.Combobox(form_frame, textvariable=status_var, values=['Pending', 'Completed'], width=27, font=("Times New Roman", 12))
        status_combo.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Deskripsi:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
        desc_text.insert("1.0", task.get('deskripsi', ''))
        desc_text.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        button_frame = tk.Frame(dialog, bg=self.bg_color)
        button_frame.pack(pady=20)
```

```python
        def save_changes():
            student = student_var.get()
            title = title_entry.get().strip()
            course = course_entry.get().strip()
            deadline = deadline_entry.get().strip()
            status = status_var.get()
            description = desc_text.get("1.0", tk.END).strip()

            if not all([student, title, course, deadline, status]):
                messagebox.showerror("Error", "Semua field harus diisi!")
                return

            try:
                datetime.strptime(deadline, '%Y-%m-%d')
            except:
                messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
                return

            try:
                task['username_pemilik'] = student
                task['judul'] = title
                task['mata_kuliah'] = course
                task['deadline'] = deadline
                task['status'] = status
                task['deskripsi'] = description

                self.update_task_in_json(task)
                self.create_backup_files()

                messagebox.showinfo("Success", "Tugas berhasil diperbarui!")
                dialog.destroy()
                self.manage_tasks_admin()
            except Exception as e:
                messagebox.showerror("Error", f"Error: {str(e)}")

        tk.Button(button_frame, text="Simpan", font=("Times New Roman", 12, "bold"), bg=self.button_color, fg="white", width=15, command=save_changes, cursor="hand2").pack(side=tk.LEFT, padx=10)
        tk.Button(button_frame, text="Batal", font=("Times New Roman", 12), bg="#f44336", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)

    except Exception as e:
        messagebox.showerror("Error", f"Error: {str(e)}")
```

```
    def edit_task_dialog_student(self, task_id):
        try:
            tasks = self.load_json(self.tasks_file)
            task = None
            for t in tasks:
                if t['id_tugas'] == task_id and t['username_pemilik'] == self.current_user:
                    task = t
                    break

            if not task:
                messagebox.showerror("Error", "Tugas tidak ditemukan atau tidak dapat diakses!")
                return

            dialog = tk.Toplevel(self.root)
            dialog.title(f"Edit Tugas #{task_id}")
            dialog.geometry("500x500")
            dialog.configure(bg=self.bg_color)
            dialog.transient(self.root)
            dialog.grab_set()

            tk.Label(dialog, text=f"EDIT TUGAS #{task_id}", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)

            form_frame = tk.Frame(dialog, bg=self.bg_color)
            form_frame.pack(pady=10)

            row = 0

            tk.Label(form_frame, text="Judul Tugas:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
            title_entry.insert(0, task['judul'])
            title_entry.grid(row=row, column=1, padx=10, pady=10)
            row += 1

            tk.Label(form_frame, text="Mata Kuliah:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
            course_entry.insert(0, task['mata_kuliah'])
            course_entry.grid(row=row, column=1, padx=10, pady=10)
            row += 1

            tk.Label(form_frame, text="Deadline (YYYY-MM-DD):", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
            deadline_entry.insert(0, task['deadline'])
            deadline_entry.grid(row=row, column=1, padx=10, pady=10)
            row += 1

            tk.Label(form_frame, text="Status:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            status_var = tk.StringVar(value=task.get('status', 'Pending'))
            status_combo = ttk.Combobox(form_frame, textvariable=status_var, values=['Pending', 'Completed'], width=27, font=("Times New Roman", 12))
            status_combo.grid(row=row, column=1, padx=10, pady=10)
            row += 1
```

```
            tk.Label(form_frame, text="Deskripsi:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
            desc_text.insert("1.0", task.get('deskripsi', ''))
            desc_text.grid(row=row, column=1, padx=10, pady=10)
            row += 1

            button_frame = tk.Frame(dialog, bg=self.bg_color)
            button_frame.pack(pady=20)
```

```
            def save_changes():
                title = title_entry.get().strip()
                course = course_entry.get().strip()
                deadline = deadline_entry.get().strip()
                status = status_var.get()
                description = desc_text.get("1.0", tk.END).strip()

                if not all([title, course, deadline, status]):
                    messagebox.showerror("Error", "Semua field harus diisi!")
                    return

                try:
                    datetime.strptime(deadline, '%Y-%m-%d')
                except:
                    messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
                    return

                try:
                    task['judul'] = title
                    task['mata_kuliah'] = course
                    task['deadline'] = deadline
                    task['status'] = status
                    task['deskripsi'] = description

                    self.update_task_in_json(task)
                    self.create_backup_files()

                    messagebox.showinfo("Success", "Tugas berhasil diperbarui!")
                    dialog.destroy()
                    self.view_my_tasks_gui()
                except Exception as e:
                    messagebox.showerror("Error", f"Error: {str(e)}")

            tk.Button(button_frame, text="Simpan", font=("Times New Roman", 12, "bold"), bg=self.button_color, fg="white", width=15, command=save_changes, cursor="hand2").pack(side=tk.LEFT, padx=10)
            tk.Button(button_frame, text="Batal", font=("Times New Roman", 12), bg="#f44336", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)

        except Exception as e:
            messagebox.showerror("Error", f"Error: {str(e)}")
```

```
    def export_to_excel(self):
        try:
            users = self.load_json(self.users_file)
            pd.DataFrame(users).to_excel('export_users.xlsx', index=False)

            tasks = self.load_json(self.tasks_file)
            pd.DataFrame(tasks).to_excel('export_tasks.xlsx', index=False)

            messagebox.showinfo("Export Success", "Data berhasil diekspor ke Excel!")
        except Exception as e:
            messagebox.showerror("Export Error", f"Error saat export: {str(e)}")

    def import_from_excel(self):
        try:
            file_path = simpledialog.askstring("Import", "Masukkan nama file Excel (users.xlsx):")
            if file_path and os.path.exists(file_path):
                df = pd.read_excel(file_path)
                data = df.to_dict('records')
                self.save_json(self.users_file, data)
                messagebox.showinfo("Import Success", "Data berhasil diimport dari Excel!")
        except Exception as e:
            messagebox.showerror("Import Error", f"Error saat import: {str(e)}")

    def logout(self):
        self.stop_reminder_system()
        self.current_user = None
        self.current_role = None
        self.show_welcome_screen()
```

```
    def add_student_dialog(self):
        dialog = tk.Toplevel(self.root)
        dialog.title("Tambah Mahasiswa")
        dialog.geometry("400x350")
        dialog.configure(bg=self.bg_color)
        dialog.transient(self.root)
        dialog.grab_set()

        tk.Label(dialog, text="TAMBAH MAHASISWA", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)

        form_frame = tk.Frame(dialog, bg=self.bg_color)
        form_frame.pack(pady=10)

        row = 0

        tk.Label(form_frame, text="Username:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
        username_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Password:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30, show="*")
        password_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        tk.Label(form_frame, text="Konfirmasi Password:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
        confirm_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30, show="*")
        confirm_entry.grid(row=row, column=1, padx=10, pady=10)
        row += 1

        button_frame = tk.Frame(dialog, bg=self.bg_color)
        button_frame.pack(pady=20)
```

```
        def save_student():
            username = username_entry.get().strip()
            password = password_entry.get().strip()
            confirm = confirm_entry.get().strip()

            if not all([username, password, confirm]):
                messagebox.showerror("Error", "Semua field harus diisi!")
                return

            if len(username) < 3:
                messagebox.showerror("Error", "Username minimal 3 karakter!")
                return

            if len(password) < 6:
                messagebox.showerror("Error", "Password minimal 6 karakter!")
                return

            if password != confirm:
                messagebox.showerror("Error", "Password tidak cocok!")
                return

            try:
                users = self.load_json(self.users_file)
                for user in users:
                    if user['username'] == username:
                        messagebox.showerror("Error", "Username sudah digunakan!")
                        return

                new_student = {'username': username, 'password': self.hash_password(password), 'role': 'mahasiswa'}
                users.append(new_student)
                self.save_json(self.users_file, users)
                self.create_backup_files()
                messagebox.showinfo("Success", f"Mahasiswa {username} berhasil ditambahkan!")
                dialog.destroy()
                self.manage_students()
            except Exception as e:
                messagebox.showerror("Error", f"Error: {str(e)}")

        tk.Button(button_frame, text="Simpan", font=("Times New Roman", 12, "bold"), bg=self.button_color, fg="white", width=15, command=save_student, cursor="hand2").pack(side=tk.LEFT, padx=10)
        tk.Button(button_frame, text="Batal", font=("Times New Roman", 12), bg="#f44336", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)
```

```
    def edit_student_dialog(self, tree):
        try:
            selected_item = tree.selection()
            if not selected_item:
                return

            item = tree.item(selected_item[0])
            username = item['values'][0]

            users = self.load_json(self.users_file)
            student = None
            for user in users:
                if user['username'] == username and user['role'] == 'mahasiswa':
                    student = user
                    break

            if not student:
                messagebox.showerror("Error", "Mahasiswa tidak ditemukan!")
                return

            dialog = tk.Toplevel(self.root)
            dialog.title(f"Edit Mahasiswa: {username}")
            dialog.geometry("400x400")
            dialog.configure(bg=self.bg_color)
            dialog.transient(self.root)
            dialog.grab_set()

            tk.Label(dialog, text=f"EDIT MAHASISWA: {username}", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)

            form_frame = tk.Frame(dialog, bg=self.bg_color)
            form_frame.pack(pady=10)

            row = 0

            tk.Label(form_frame, text="Username saat ini:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            tk.Label(form_frame, text=username, font=("Times New Roman", 12, "bold"), bg=self.bg_color).grid(row=row, column=1, padx=10, pady=10, sticky="w")
            row += 1

            tk.Label(form_frame, text="Username baru (kosongkan jika tidak diubah):", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            new_username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
            new_username_entry.grid(row=row, column=1, padx=10, pady=10)
            row += 1

            tk.Label(form_frame, text="Password baru (kosongkan jika tidak diubah):", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            new_password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30, show="*")
            new_password_entry.grid(row=row, column=1, padx=10, pady=10)
            row += 1
```

```
            tk.Label(form_frame, text="Konfirmasi password baru:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
            confirm_password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30, show="*")
            confirm_password_entry.grid(row=row, column=1, padx=10, pady=10)
            row += 1

            button_frame = tk.Frame(dialog, bg=self.bg_color)
            button_frame.pack(pady=20)
```

```
        def save_changes():
            new_username = new_username_entry.get().strip()
            new_password = new_password_entry.get().strip()
            confirm_password = confirm_password_entry.get().strip()

            if new_username:
                if len(new_username) < 3:
                    messagebox.showerror("Error", "Username minimal 3 karakter!")
                    return

                for user in users:
                    if user['username'] == new_username and user['username'] != username:
                        messagebox.showerror("Error", "Username sudah digunakan!")
                        return

            if new_password:
                if len(new_password) < 6:
                    messagebox.showerror("Error", "Password minimal 6 karakter!")
                    return

                if new_password != confirm_password:
                    messagebox.showerror("Error", "Password tidak cocok!")
                    return

            try:
                for i, user in enumerate(users):
                    if user['username'] == username:
                        if new_username:
                            users[i]['username'] = new_username
                        if new_password:
                            users[i]['password'] = self.hash_password(new_password)
                        break

                self.save_json(self.users_file, users)

                if new_username:
                    tasks = self.load_json(self.tasks_file)
                    for task in tasks:
                        if task['username_pemilik'] == username:
                            task['username_pemilik'] = new_username
                    self.save_json(self.tasks_file, tasks)

                self.create_backup_files()
                messagebox.showinfo("Success", "Data mahasiswa berhasil diperbarui!")
                dialog.destroy()
                self.manage_students()
            except Exception as e:
                messagebox.showerror("Error", f"Error: {str(e)}")
```

```
        def delete_student():
            if not messagebox.askyesno("Konfirmasi", f"Apakah Anda yakin ingin menghapus mahasiswa {username}?\nSemua tugas yang dimiliki mahasiswa ini juga akan dihapus!"):
                return

            try:
                new_users = [user for user in users if user['username'] != username]
                self.save_json(self.users_file, new_users)

                tasks = self.load_json(self.tasks_file)
                new_tasks = [task for task in tasks if task['username_pemilik'] != username]
                self.save_json(self.tasks_file, new_tasks)

                self.create_backup_files()
                messagebox.showinfo("Success", f"Mahasiswa {username} berhasil dihapus!")
                dialog.destroy()
                self.manage_students()
            except Exception as e:
                messagebox.showerror("Error", f"Error: {str(e)}")

        tk.Button(button_frame, text="Simpan", font=("Times New Roman", 12, "bold"), bg=self.button_color, fg="white", width=12, command=save_changes, cursor="hand2").pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="Hapus", font=("Times New Roman", 12, "bold"), bg="#f44336", fg="white", width=12, command=delete_student, cursor="hand2").pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="Batal", font=("Times New Roman", 12), bg="#607D8B", fg="white", width=12, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=5)

    except Exception as e:
        messagebox.showerror("Error", f"Error: {str(e)}")
```

```
    def edit_task_dialog_admin(self, tree):
        try:
            selected_item = tree.selection()
            if not selected_item:
                return

            item = tree.item(selected_item[0])
            task_id = item['values'][0]
            self.edit_task_dialog_admin_by_id(task_id)
        except Exception as e:
            messagebox.showerror("Error", f"Error: {str(e)}")

def main():
    root = tk.Tk()
    app = TaskReminderApp(root)

    def on_closing():
        app.stop_reminder_system()
        root.destroy()

    root.protocol("WM_DELETE_WINDOW", on_closing)
    root.mainloop()

if __name__ == "__main__":
    main()
```

DAFTAR PUSTAKA

1. Microsoft Corporation. (2023). *Visual Studio Code Documentation*. Microsoft. https://code.visualstudio.com/docs
2. Python Software Foundation. (2024). *Python Documentation*. Python Software Foundation. https://docs.python.org/3/
3. Microsoft Corporation. (2023). *Microsoft Excel Documentation*. Microsoft. https://support.microsoft.com/excel
4. ECMA International. (2017). *The JSON Data Interchange Syntax (ECMA-404)*. https://www.json.org/json-en.html