

**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 melewatkkan 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 :

```

1  // Users
2  {"username": "string", "password": "hashed", "role": "string"}
3
4  // Tasks
5  {
6      "id_tugas": int,
7      "judul": "string",
8      "mata_kuliah": "string",
9      "deadline": "date",
10     "username_pemilik": "string",
11     "status": "string",
12     "dibuat_oleh": "string"
13 }

```

Arsitektur :

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

## 5. Limitasi Sistem

Kapasitas :

- Optimal :  $\leq$ 1000 user,  $\leq$ 10.000 tugas
- File size :  $\leq$ 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

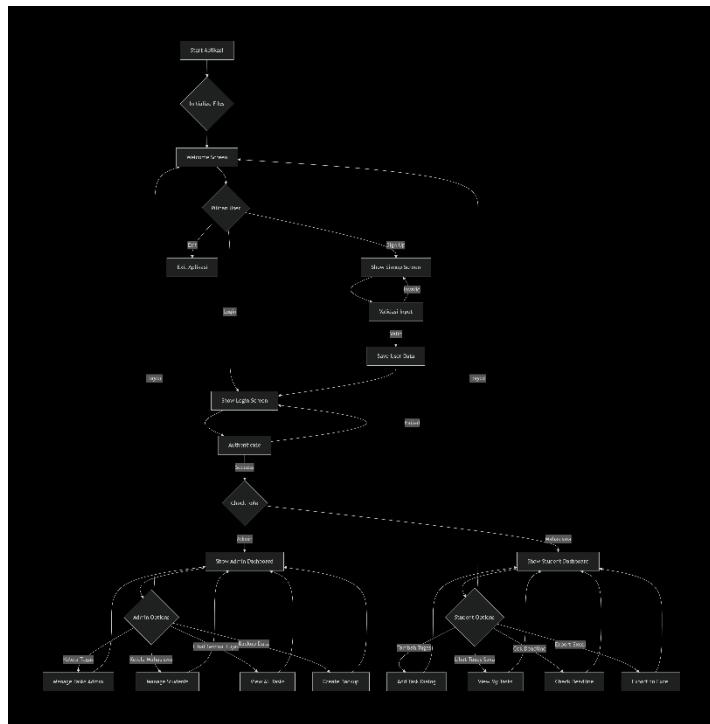
- Cross platform
- Backup otomatis

Kekurangan :

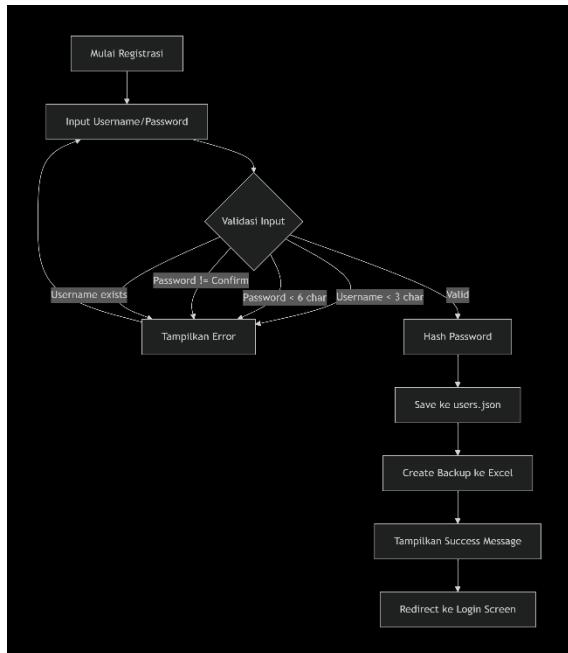
- Offline only
- Fitur dasar saja
- NO mobile version

## E. Diagram Alur

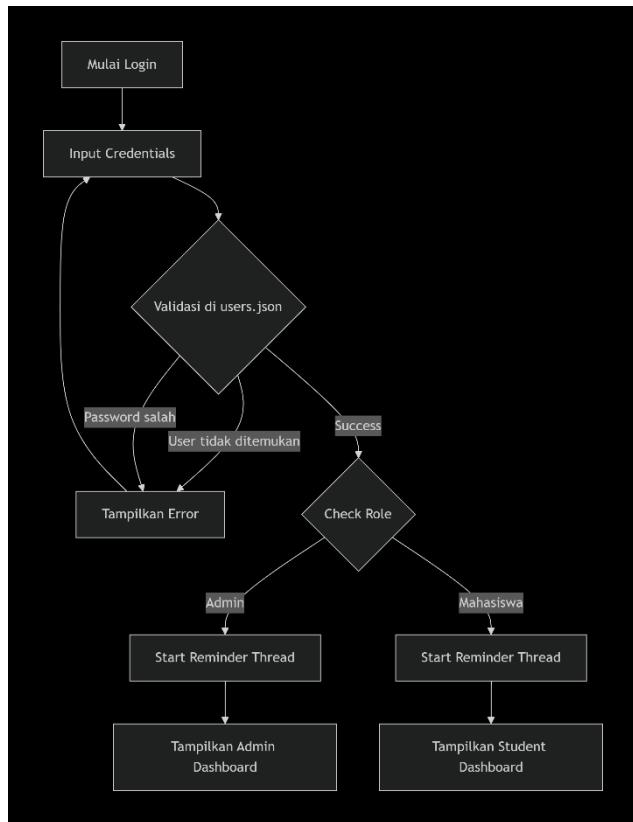
### 1. Alur utama



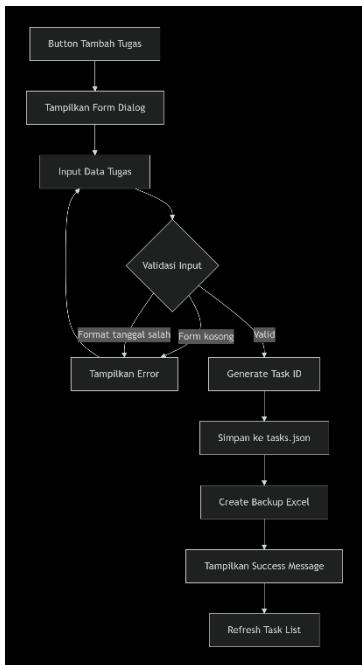
### 2. Alur Registrasi (Sign Up)



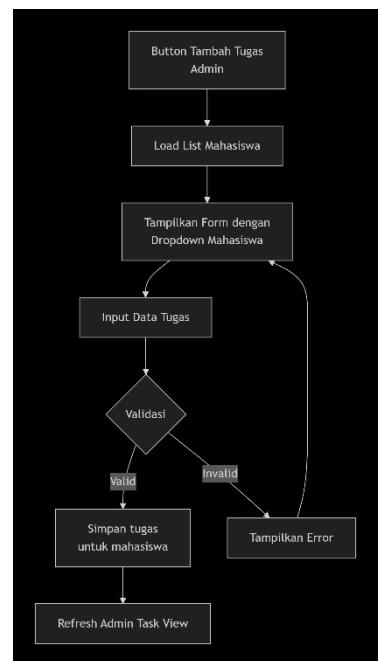
### 3. Alur Login



### 4. Alur Tambah Tugas

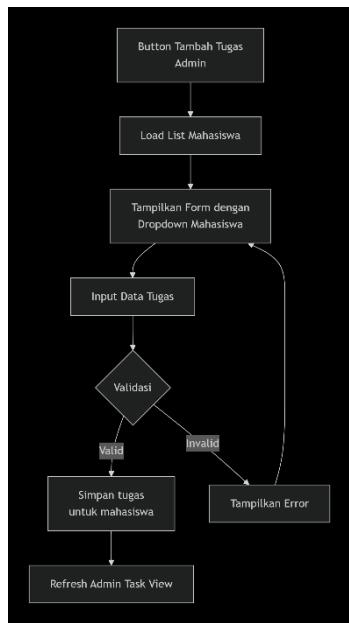


4.1 Mahasiswa

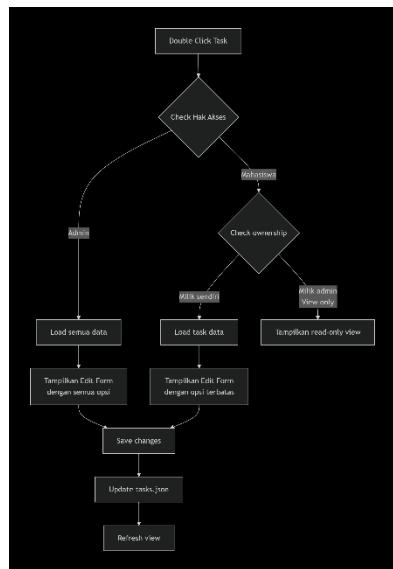


4.2. Admin

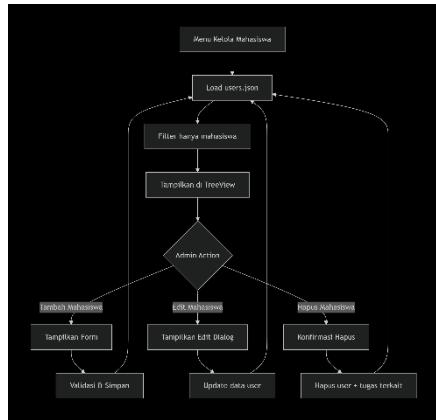
## 5. Alur Reminder System



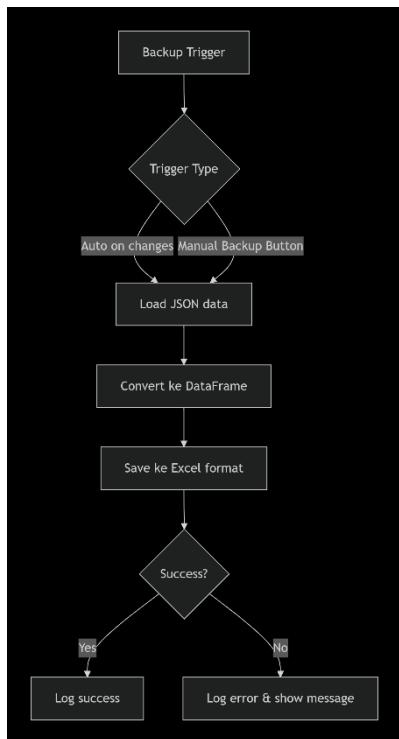
## 6. Alur Edit Tugas



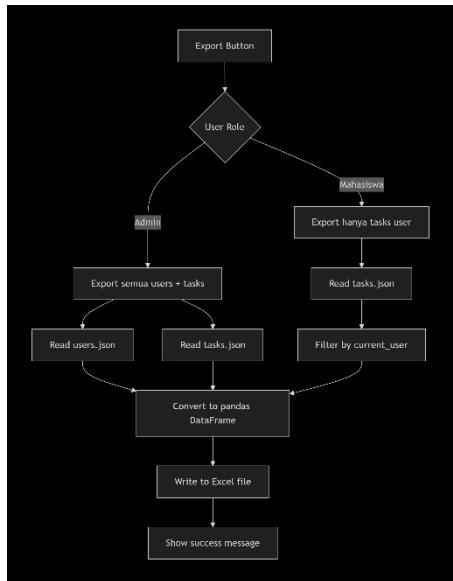
## 7. Alur Kelola Mahasiswa (Admin)



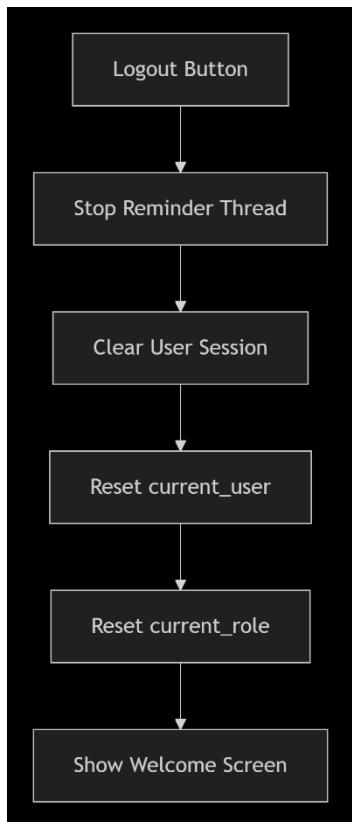
## 8. Alur Backup Sistem



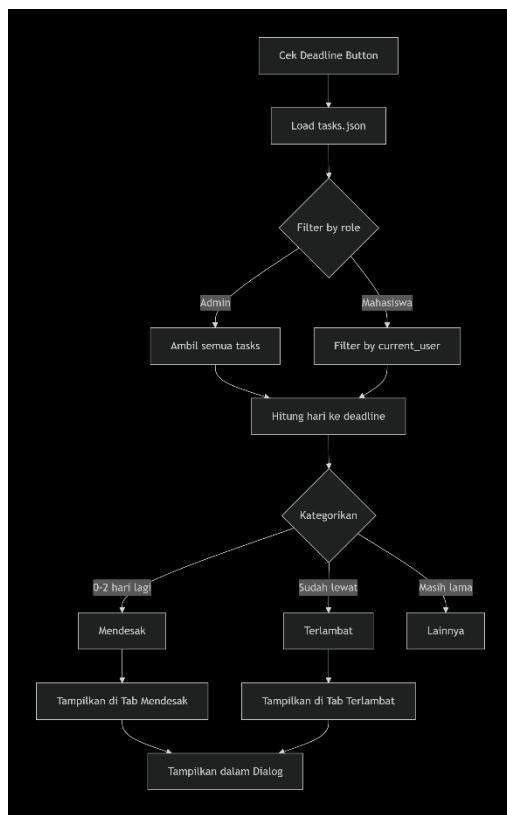
## 9. Alur Export to Excel



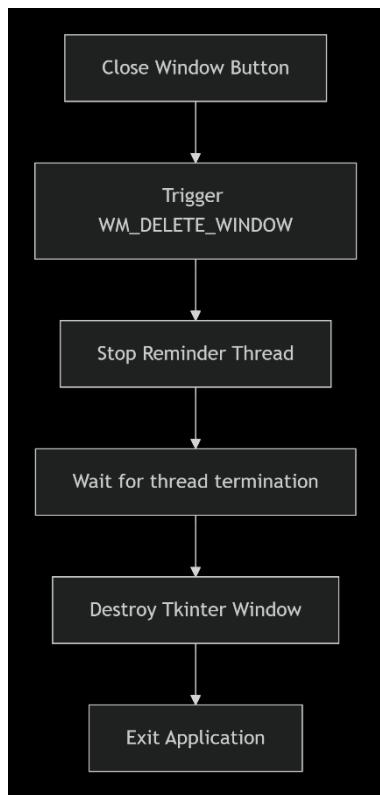
## 10. Alur Log Out



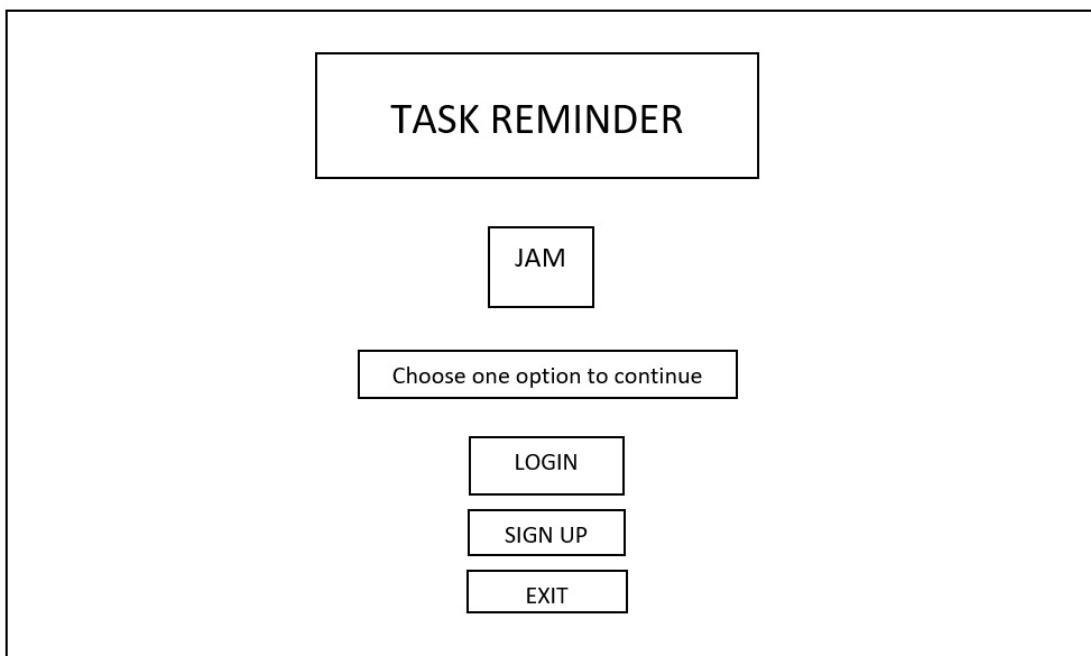
## 11. Alur Cek Deadline



## 12. Alur Shutdown



## F. Sketsa Desain Antarmuka



## IMPLEMENTASI

```
40     def show_welcome_screen(self):
41         self.clear_window()
42
43         welcome_label = tk.Label(self.root, text="TASK REMINDER", font=("Berlin Sans FB Demi", 50, "bold"), bg=self.bg_color, fg="#00008B")
44         welcome_label.pack(pady=50)
45
46         welcome_label = tk.Label(self.root, text="!", font=("Berlin Sans FB Demi", 60, "bold"), bg=self.bg_color, fg="#00008B")
47         welcome_label.pack(pady=10)
48
49         subtitle_label = tk.Label(self.root, text="Choose one option to continue : ", font=("Berlin Sans FB Demi", 16, "bold"), bg=self.bg_color, fg="#00008B")
50         subtitle_label.pack(pady=20)
51
52         button_frame = tk.Frame(self.root, bg=self.bg_color)
53         button_frame.pack(pady=50)
54
55         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)
56         login_button.pack(pady=5)
57
58         signup_button = tk.Button(button_frame, text="SIGN UP", font=("Times New Roman", 14, "bold"), bg="#00008B", fg="white", width=20, height=2, cursor="hand2", command=self.show_signup_screen)
59         signup_button.pack(pady=5)
60
61         exit_button = tk.Button(button_frame, text="EXIT", font=("Times New Roman", 14), bg="#F4A460", fg="white", width=22, height=2, cursor="hand2", command=self.root.quit)
62         exit_button.pack(pady=5)
```

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

```
130     def show_admin_dashboard(self):
131         self.clear_window()
132
133         header_frame = tk.Frame(self.root, bg=self.admin_color, height=80)
134         header_frame.pack(fill=tk.X)
135         header_frame.pack_propagate(False)
136
137         tk.Label(header_frame, text=f"ADMIN DASHBOARD\nWelcome, {self.current_user}").pack()
138
139         logout_button = tk.Button(header_frame, text="Logout", font=("Times New Roman", 14, "bold"), bg="#00008B", fg="white", width=15, height=2, cursor="hand2", command=self.logout)
140         logout_button.place(x=10, y=20)
141
142         self.show_admin_stats()
143
144         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

```
202     def check_automatic_reminders(self):
203         try:
204             tasks = self.load_json(self.tasks_file)
205             today = datetime.now().date()
206
207             for task in tasks:
208                 if self.current_role == 'mahasiswa' and task['username_pemilik'] != self.current_user:
209                     continue
210
211                 if task.get('status', 'Pending') == 'Completed':
212                     continue
213
214                 try:
215                     deadline_date = datetime.strptime(str(task['deadline']), '%Y-%m-%d').date()
216                     days_left = (deadline_date - today).days
217
218                     if days_left == 0 and not task.get('notified_today', False):
219                         self.send_notification("⚠️ Deadline Hari Ini!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")  
220                         task['notified_today'] = True
221                         self.update_task_in_json(task)
222
223                     elif days_left == 1 and not task.get('notified_tomorrow', False):
224                         self.send_notification("⚠️ Deadline Besok!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")  
225                         task['notified_tomorrow'] = True
226                         self.update_task_in_json(task)
227
228                     elif days_left < 0 and not task.get('notified_late', False):
229                         self.send_notification("⚠️ Tugas Terlambat!", f"Tugas: {task['judul']}\nTerlambat: {abs(days_left)} hari")
230                         task['notified_late'] = True
231                         self.update_task_in_json(task)
232
233                 except:
234                     continue
235             except Exception as e:
236                 print(f"Error checking reminders: {e}")
237
238         def send_notification(self, title, message):
239             try:
240                 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

```

❶ TaskReminderApp1.py ×
C:\Users\bagum\Downloads\REMINDER TUGAS\TaskReminderApp1.py (preview ⑧)
1  import tkinter as tk
2  from tkinter import ttk, messagebox, simpledialog
3  import json
4  import os
5  import hashlib
6  from datetime import datetime
7  import threading
8  import time
9  import pandas as pd
10 from plyer import notification
11
12 class TaskReminderApp:
13     def __init__(self, root):
14         self.root = root
15         self.root.title("TASK REMINDER")
16         self.root.geometry("1000x700")
17
18         self.users_file = 'users.json'
19         self.tasks_file = 'tasks.json'
20         self.backup_users_file = 'users_backup.xlsx'
21         self.backup_tasks_file = 'tasks_backup.xlsx'
22
23         self.current_user = None
24         self.current_role = None
25
26         self.bg_color = "#f5f7da"
27         self.button_color = "#008000"
28         self.button_hover = "#ddeeaf"
29         self.admin_color = "#2c5c92"
30         self.student_color = "#2c5c92"
31
32         self.root.configure(bg=self.bg_color)
33
34         self.reminder_running = False
35         self.reminder_thread = None
36
37         self.initialize_files()
38         self.show_welcome_screen()
39
40
41     def show_welcome_screen(self):
42         self.clear_window()
43
44         welcome_label = tk.Label(self.root, text="TASK REMINDER", font=("Berlin Sans FB Demi", 50, "bold"), bg=self.bg_color, fg="#008000")
45         welcome_label.pack(pady=50)
46
47         welcome_label = tk.Label(self.root, text="!", font=("Berlin Sans FB Demi", 60, "bold"), bg=self.bg_color, fg="#008000")
48         welcome_label.pack(pady=10)
49
50         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")
51         subtitle_label.pack(pady=20)
52
53         button_frame = tk.Frame(self.root, bg=self.bg_color)
54         button_frame.pack(pady=50)
55
56         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)
57         login_button.pack(pady=5)
58
59         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)
60         signup_button.pack(pady=5)
61
62         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)
63         exit_button.pack(pady=5)
64
65
66     def show_login_screen(self):
67         self.clear_window()
68
69         header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
70         header_frame.pack(fill=tk.X)
71         header_frame.pack_propagate(False)
72
73         tk.Label(header_frame, text="LOGIN", font=("Times New Roman", 18, "bold"), bg=self.admin_color, fg="white").pack(pady=15)
74
75         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")
76         back_button.place(x=10, y=15)
77
78         form_frame = tk.Frame(self.root, bg=self.bg_color)
79         form_frame.pack(pady=50)
80
81         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")
82         self.username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50)
83         self.username_entry.grid(row=0, column=1, padx=10, pady=35)
84
85         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")
86         self.password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50, show="")
87         self.password_entry.grid(row=1, column=1, padx=10, pady=35)
88
89         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")
90         self.role_var = tk.StringVar(value="mahasiswa")
91         role_combo = ttk.Combobox(form_frame, textvariable=self.role_var, values=["mahasiswa", "admin"], width=48, font=("Times New Roman", 12), state="readonly")
92         role_combo.grid(row=2, column=1, padx=10, pady=35)
93
94         login_btn_frame = tk.Frame(self.root, bg=self.bg_color)
95         login_btn_frame.pack(pady=30)
96
97         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)
98         login_button.pack()

```

```

96     def show_signup_screen(self):
97         self.clear_window()
98
99         header_frame = tk.Frame(self.root, bg=self.student_color, height=60)
100        header_frame.pack(fill=tk.X)
101        header_frame.pack_propagate(False)
102
103        tk.Label(header_frame, text="SIGN UP", font=("Times New Roman", 18, "bold"), bg=self.student_color, fg="white").pack(pady=15)
104        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")
105        back_button.place(x=10, y=15)
106
107        form_frame = tk.Frame(self.root, bg=self.bg_color)
108        form_frame.pack(pady=30)
109
110        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")
111        self.signup_username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50)
112        self.signup_username_entry.grid(row=0, column=1, padx=10, pady=35)
113
114        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")
115        self.signup_password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50, show="*")
116        self.signup_password_entry.grid(row=1, column=1, padx=10, pady=35)
117
118        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")
119        self.signup_confirm_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50, show="*")
120        self.signup_confirm_entry.grid(row=2, column=1, padx=10, pady=35)
121
122        signup_btn_frame = tk.Frame(self.root, bg=self.bg_color)
123        signup_btn_frame.pack(pady=30)
124
125        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)
126        signup_button.pack()
127
128

```

```

130    def show_admin_dashboard(self):
131        self.clear_window()
132
133        header_frame = tk.Frame(self.root, bg=self.admin_color, height=80)
134        header_frame.pack(fill=tk.X)
135        header_frame.pack_propagate(False)
136
137        tk.Label(header_frame, text="ADMIN DASHBOARD\nWelcome, " + self.current_user + "!", font=("Times New Roman", 18, "bold"), bg=self.admin_color, fg="white").pack(pady=10)
138
139        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")
140        logout_button.place(x=10, y=20)
141
142        self.show_admin_stats()
143
144        menu_frame = tk.Frame(self.root, bg=self.bg_color)
145        menu_frame.pack(pady=85)
146
147        buttons = [
148            ("➕ Kelola Tugas", self.manage_tasks_admin),
149            ("➕ Kelola Mahasiswa", self.manage_students),
150            ("➕ Lihat Semua Tugas", self.view_all_tasks_gui),
151            ("EXPORT", self.export_to_excel),
152            ("cek Deadline", self.check_deadline_gui),
153            ("Backup Data", self.create_backup_files)
154        ]
155
156        for i, (text, command) in enumerate(buttons):
157            row, col = i // 3, i % 3
158            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)
159            btn.grid(row=row, column=col, padx=20, pady=20)

```

```

161    def show_student_dashboard(self):
162        self.clear_window()
163
164        header_frame = tk.Frame(self.root, bg=self.student_color, height=80)
165        header_frame.pack(fill=tk.X)
166        header_frame.pack_propagate(False)
167
168        tk.Label(header_frame, text="STUDENT DASHBOARD\nWelcome, " + self.current_user + "!", font=("Times New Roman", 18, "bold"), bg=self.student_color, fg="white").pack(pady=10)
169
170        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")
171        logout_button.place(x=10, y=20)
172
173        self.show_student_stats()
174
175        menu_frame = tk.Frame(self.root, bg=self.bg_color)
176        menu_frame.pack(pady=85)
177
178        buttons = [
179            ("➕ Tambah Tugas", self.add_task_dialog),
180            ("➕ Lihat Tugas Saya", self.view_my_tasks_gui),
181            ("cek Deadline", self.check_deadline_gui),
182            ("EXPORT Tugas Saya", self.export_to_excel),
183            ("REFRESH", lambda: self.show_student_dashboard())
184        ]
185
186        for i, (text, command) in enumerate(buttons):
187            row, col = i // 3, i % 3
188            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)
189            btn.grid(row=row, column=col, padx=20, pady=20)

```

```

191     def initialize_files(self):
192         if not os.path.exists(self.users_file):
193             users = [
194                 {'username': 'admin', 'password': self.hash_password('admin123'), 'role': 'admin'},
195                 {'username': 'mahasiswa1', 'password': self.hash_password('mhs123'), 'role': 'mahasiswa'},
196                 {'username': 'mahasiswa2', 'password': self.hash_password('mhs123'), 'role': 'mahasiswa'}
197             ]
198             self.save_json(self.users_file, users_data)
199
200         if not os.path.exists(self.tasks_file):
201             self.save_json(self.tasks_file, [])
202
203         self.create_backup_files()
204
205     def save_json(self, filename, data):
206         try:
207             with open(filename, 'w', encoding='utf-8') as f:
208                 json.dump(data, f, indent=4, ensure_ascii=False)
209         except Exception as e:
210             print(f"Error saving {filename}: {e}")
211
212     def load_json(self, filename):
213         try:
214             if os.path.exists(filename):
215                 with open(filename, 'r', encoding='utf-8') as f:
216                     return json.load(f)
217             return []
218         except Exception as e:
219             print(f"Error loading {filename}: {e}")
220             return []
221
222     def create_backup_files(self):
223         try:
224             users = self.load_json(self.users_file)
225             if users:
226                 pd.DataFrame(users).to_excel(self.backup_users_file, index=False)
227
228             tasks = self.load_json(self.tasks_file)
229             if tasks:
230                 pd.DataFrame(tasks).to_excel(self.backup_tasks_file, index=False)
231         except Exception as e:
232             print(f"Error creating backup: {e}")
233
234     def hash_password(self, password):
235         return hashlib.sha256(password.encode()).hexdigest()
236
237     def clear_window(self):
238         for widget in self.root.winfo_children():
239             widget.destroy()

```

```

241     def start_reminder_system(self):
242         if self.reminder_running:
243             return
244
245         self.reminder_running = True
246         self.reminder_thread = threading.Thread(target=self.reminder_check_loop, daemon=True)
247         self.reminder_thread.start()
248
249     def stop_reminder_system(self):
250         self.reminder_running = False
251         if self.reminder_threads:
252             self.reminder_thread.join(timeout=1)
253
254     def reminder_check_loop(self):
255         while self.reminder_running:
256             try:
257                 self.check_automatic_reminders()
258                 time.sleep(30)
259             except Exception as e:
260                 print(f"Error in reminder loop: {e}")
261                 time.sleep(30)
262
263     def check_automatic_reminders(self):
264         try:
265             tasks = self.load_json(self.tasks_file)
266             today = datetime.now().date()

```

```

268         for task in tasks:
269             if self.current_role == 'mahasiswa' and task['username_pemilik'] != self.current_user:
270                 continue
271
272             if task.get('status', 'Pending') == 'Completed':
273                 continue
274
275             try:
276                 deadline_date = datetime.strptime(str(task['deadline']), '%Y-%m-%d').date()
277                 days_left = (deadline_date - today).days
278
279                 if days_left == 0 and not task.get('notified_today', False):
280                     self.send_notification("⚠ Deadline Hari Ini!", f"\nTugas: {task['judul']}\nDeadline: {task['deadline']}")"
281                     task['notified_today'] = True
282                     self.update_task_in_json(task)
283
284                 elif days_left == 1 and not task.get('notified_tomorrow', False):
285                     self.send_notification("⚠ Deadline Besok!", f"\nTugas: {task['judul']}\nDeadline: {task['deadline']}")"
286                     task['notified_tomorrow'] = True
287                     self.update_task_in_json(task)
288
289                 elif days_left < 0 and not task.get('notified_late', False):
290                     self.send_notification("⚠ Tugas Terlambat!", f"\nTugas: {task['judul']}\nTerlambat: {abs(days_left)} hari")
291                     task['notified_late'] = True
292                     self.update_task_in_json(task)
293
294             except:
295                 continue
296         except Exception as e:
297             print(f"Error checking reminders: {e}")

```

```

296     def send_notification(self, title, message):
297         try:
298             notification.notify(title=title, message=message, app_name="Task Reminder", timeout=10)
299         except:
300             if self.root and tk.Toplevel:
301                 self.root.after(0, lambda: messagebox.showwarning(title, message))
302
303     def update_task_in_json(self, updated_task):
304         try:
305             tasks = self.load_json(self.tasks_file)
306             for i, task in enumerate(tasks):
307                 if task['id_tugas'] == updated_task['id_tugas']:
308                     tasks[i] = updated_task
309                     break
310             self.save_json(self.tasks_file, tasks)
311         except Exception as e:
312             print(f"Error updating task in JSON: {e}")

```

```

314     def signup(self):
315         username = self.signup_username_entry.get().strip()
316         password = self.signup_password_entry.get().strip()
317         confirm_password = self.signup_confirm_entry.get().strip()
318
319         if not all([username, password, confirm_password]):
320             messagebox.showerror("Sign Up Error", "Semua field harus diisi!")
321             return
322
323         if len(username) < 3:
324             messagebox.showerror("Sign Up Error", "Username minimal 3 karakter!")
325             return
326
327         if len(password) < 6:
328             messagebox.showerror("Sign Up Error", "Password minimal 6 karakter!")
329             return
330
331         if password != confirm_password:
332             messagebox.showerror("Sign Up Error", "Password tidak cocok!")
333             return
334
335         try:
336             users = self.load_json(self.users_file)
337             usernames = [user['username'] for user in users]
338
339             if username in usernames:
340                 messagebox.showerror("Sign Up Error", "Username sudah digunakan!")
341                 return
342
343             new_user = {'username': username, 'password': self.hash_password(password), 'role': 'mahasiswa'}
344             users.append(new_user)
345             self.save_json(self.users_file, users)
346
347             self.create_backup_files()
348
349             messagebox.showinfo("Sign Up Success", f"Akun {username} berhasil dibuat!\nSilakan login dengan akun Anda.")
350             self.show_login_screen()
351
352         except Exception as e:
353             messagebox.showerror("Sign Up Error", f"Error saat mendaftar: {str(e)}")

```

```

355     def login(self):
356         username = self.username_entry.get()
357         password = self.password_entry.get()
358         role = self.role_var.get()
359
360         try:
361             users = self.load_json(self.users_file)
362             user_found = None
363             for user in users:
364                 if user['username'] == username and user['role'] == role:
365                     user_found = user
366                     break
367
368             if user_found:
369                 if self.hash_password(password) == user_found['password']:
370                     self.current_user = username
371                     self.current_role = role
372                     self.start_reminder_system()
373
374                     if role == 'admin':
375                         self.show_admin_dashboard()
376                     else:
377                         self.show_student_dashboard()
378
379                     messagebox.showinfo("Login Success", "Selamat datang, Admin!")
380
381             else:
382                 messagebox.showerror("Login Error", "Username atau role tidak ditemukan!")
383
384         except Exception as e:
385             messagebox.showerror("Error", f"Error saat login: {str(e)}")
386
387     def get_next_task_id(self):
388         tasks = self.load_json(self.tasks_file)
389         if not tasks:
390             return 1
391         max_id = max(task['id_tugas'] for task in tasks)
392         return max_id + 1
393

```

```

393     def add_task_dialog(self):
394         dialog = tk.Toplevel(self.root)
395         dialog.title("Tambah Tugas Baru")
396         dialog.geometry("500x500")
397         dialog.configure(bg=self.bg_color)
398         dialog.transient(self.root)
399         dialog.grab_set()
400
401         tk.Label(dialog, text="TAMBAH TUGAS BARU", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=10)
402
403         form_frame = tk.Frame(dialog, bg=self.bg_color)
404         form_frame.pack(pady=10)
405
406         row = 0
407
408         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")
409         title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
410         title_entry.grid(row=row, column=1, padx=10, pady=10)
411         row += 1
412
413         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")
414         course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
415         course_entry.grid(row=row, column=1, padx=10, pady=10)
416         row += 1
417
418         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")
419         deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
420         deadline_entry.insert(0, datetime.now().strftime("%Y-%m-%d"))
421         deadline_entry.grid(row=row, column=1, padx=10, pady=10)
422         row += 1
423
424         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")
425         desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=6)
426         desc_text.grid(row=row, column=1, padx=10, pady=10)
427         row += 1
428
429         button_frame = tk.Frame(dialog, bg=self.bg_color)
430         button_frame.pack(pady=10)

```

```

432     def save_task():
433         title = title_entry.get().strip()
434         course = course_entry.get().strip()
435         deadline = deadline_entry.get().strip()
436         description = desc_text.get("1.0", tk.END).strip()
437
438         if not all([title, course, deadline]):
439             messagebox.showerror("Error", "Semua field harus diisi!")
440             return
441
442         try:
443             datetime.strptime(deadline, '%Y-%m-%d')
444         except:
445             messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-00")
446             return
447
448         try:
449             tasks = self.load_json(self.tasks_file)
450             new_task = {
451                 'id_tugas': self.get_next_task_id(),
452                 'judul': title,
453                 'deskripsi': description,
454                 'mata_kuliah': course,
455                 'deadline': deadline,
456                 'username_pemilik': self.current_user,
457                 'dibuat_oleh': 'mahasiswa',
458                 'status': 'Pending',
459                 'created_at': datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
460                 'notified_today': False,
461                 'notified_tomorrow': False,
462                 'notified_late': False
463             }
464             tasks.append(new_task)
465             self.save_json(self.tasks_file, tasks)
466             self.create_backup_files()
467             messagebox.showinfo("Success", "Tugas berhasil ditambahkan!")
468             dialog.destroy()
469             self.view_my_tasks_gui()
470         except Exception as e:
471             messagebox.showerror("Error", f"Error: {str(e)}")
472
473         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)
474         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)

```

```

476     def add_task_dialog_admin(self):
477         dialog = tk.Toplevel(self.root)
478         dialog.title("Tambah Tugas Baru")
479         dialog.geometry("500x500")
480         dialog.configure(bg=self.bg_color)
481         dialog.transient(self.root)
482         dialog.grab_set()
483
484         tk.Label(dialog, text="TAMBAH TUGAS BARU (ADMIN)", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
485
486         try:
487             users = self.load_json(self.users_file)
488             students = [user for user in users if user['role'] == 'mahasiswa']
489             student_list = [student['username'] for student in students]
490         except:
491             student_list = []
492
493         form_frame = tk.Frame(dialog, bg=self.bg_color)
494         form_frame.pack(pady=10)
495
496         row = 0

```

```

497             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")
498             student_var = tk.StringVar(value=student_list[0] if student_list else "")
499             student_combo = ttk.Combobox(form_frame, textvariable=student_var, values=student_list, width=27, font=("Times New Roman", 12))
500             student_combo.grid(row=row, column=1, padx=10, pady=10)
501             row += 1
502
503             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")
504             title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
505             title_entry.grid(row=row, column=1, padx=10, pady=10)
506             row += 1
507
508             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")
509             course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
510             course_entry.grid(row=row, column=1, padx=10, pady=10)
511             row += 1
512
513             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")
514             deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
515             deadline_entry.insert(0, datetime.now().strftime("%Y-%m-%d"))
516             deadline_entry.grid(row=row, column=1, padx=10, pady=10)
517             row += 1
518
519             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")
520             desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
521             desc_text.grid(row=row, column=1, padx=10, pady=10)
522             row += 1
523
524             button_frame = tk.Frame(dialog, bg=self.bg_color)
525             button_frame.pack(pady=20)

```

```

526     def save_task():
527         student = student_var.get()
528         title = title_entry.get().strip()
529         course = course_entry.get().strip()
530         deadline = deadline_entry.get().strip()
531         description = desc_text.get("1.0", tk.END).strip()
532
533         if not all([student, title, course, deadline]):
534             messagebox.showerror("Error", "Semua field harus diisi!")
535             return
536
537         try:
538             datetime.strptime(deadline, '%Y-%m-%d')
539         except:
540             messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-00")
541             return
542
543         try:
544             tasks = self.load_json(self.tasks_file)
545             new_task = {
546                 'id_tugas': self.get_next_task_id(),
547                 'judul': title,
548                 'deskripsi': description,
549                 'mata_kuliah': course,
550                 'deadline': deadline,
551                 'username_pemilik': student,
552                 'dibuat_oleh': 'admin',
553                 'status': 'Pending',
554                 'created_at': datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
555                 'notified_today': False,
556                 'notified_tomorrow': False,
557                 'notified_late': False
558             }
559             tasks.append(new_task)
560             self.save_json(self.tasks_file, tasks)
561             self.create_backup_files()
562             messagebox.showinfo("Success", "Tugas berhasil ditambahkan!")
563             dialog.destroy()
564             self.manage_tasks_admin()
565         except Exception as e:
566             messagebox.showerror("Error", f"Error: {str(e)}")
567
568         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)
569         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)

```

```

593     def show_admin_stats(self):
594         try:
595             tasks = self.load_json(self.tasks_file)
596             users = self.load_json(self.users_file)
597
598             stats_frame = tk.Frame(self.root, bg=self.bg_color)
599             stats_frame.pack(pady=20)
600
601             total_tasks = len(tasks)
602             total_students = len([user for user in users if user['role'] == 'mahasiswa'])
603             pending_tasks = len([task for task in tasks if task.get('status', 'Pending') == 'Pending'])
604             completed_tasks = len([task for task in tasks if task.get('status', 'Pending') == 'Completed'])
605
606             stats = [
607                 f"Total Tugas: {total_tasks}",
608                 f"Total Mahasiswa: {total_students}",
609                 f"Tugas Pending: {pending_tasks}",
610                 f"Tugas Selesai: {completed_tasks}"
611             ]
612
613             for i, stat in enumerate(stats):
614                 tk.Label(stats_frame, text=stat, font=("Times New Roman", 12, "bold"), bg=self.bg_color).grid(row=i, column=0, padx=20)
615         except Exception as e:
616             print(f"Error loading stats: {e}")

```

```

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

```

```

634     def manage_students(self):
635         self.clear_window()
636
637         header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
638         header_frame.pack(fill=tk.X)
639         header_frame.pack_propagate(False)
640
641         tk.Label(header_frame, text="KELOLA DATA MAHASISWA", font=("Times New Roman", 16, "bold"), bg=self.admin_color, fg="white").pack(pady=15)
642
643         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")
644         back_button.place(x=10, y=15)
645
646         control_frame = tk.Frame(self.root, bg=self.bg_color)
647         control_frame.pack(pady=10)
648
649         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)
650         tk.Button(control_frame, text="Refresh", font=("Times New Roman", 11, "bold"), bg="#607080", fg="white", width=20, command=self.manage_students, cursor="hand2").pack(side=tk.LEFT, padx=5)
651
652         table_frame = tk.Frame(self.root)
653         table_frame.pack(pady=10, padx=20, fill=tk.BOTH, expand=True)
654
655         columns = ('Username', 'Role', 'Actions')
656
657         tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)
658
659         tree.heading('Username', text='Username')
660         tree.heading('Role', text='Role')
661         tree.heading('Actions', text='Actions')
662
663         tree.column('Username', width=200)
664         tree.column('Role', width=100)
665         tree.column('Actions', width=150)
666
667         scrollbar = tk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
668         tree.configure(scrollregion=scrollbar.set)
669
670         tree.grid(row=0, column=0, sticky='nsew')
671         scrollbar.grid(row=0, column=1, sticky='ns')
672
673         table_frame.grid_rowconfigure(0, weight=1)
674         table_frame.grid_columnconfigure(0, weight=1)
675
676     try:
677         users = self.load_json(self.users_file)
678         students = [user for user in users if user['role'] == 'mahasiswa']
679
680         for student in students:
681             tree.insert('', tk.END, values=(student['username'], student['role'], 'delete atau edit'))
682     except Exception as e:
683         messagebox.showerror("Error", f"Error loading data: {str(e)}")
684
685     tree.bind("<Double-1>", lambda e: self.edit_student_dialog(tree))
686
687
688
689     def manage_tasks_admin(self):
690         self.clear_window()
691
692         header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
693         header_frame.pack(fill=tk.X)
694         header_frame.pack_propagate(False)
695
696         tk.Label(header_frame, text="KELOLA TUGAS (ADMIN)", font=("Times New Roman", 16, "bold"), bg=self.admin_color, fg="white").pack(pady=15)
697
698         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")
699         back_button.place(x=10, y=15)
700
701         control_frame = tk.Frame(self.root, bg=self.bg_color)
702         control_frame.pack(pady=10, fill=tk.BOTH, expand=True)
703
704         columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Penilik', 'Status', 'Dibuat Oleh', 'Actions')
705
706         tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)
707
708         tree.heading('ID', text='ID')
709         tree.heading('Judul', text='Judul')
710         tree.heading('Mata Kuliah', text='Mata Kuliah')
711         tree.heading('Deadline', text='Deadline')
712         tree.heading('Penilik', text='Penilik')
713         tree.heading('Status', text='Status')
714         tree.heading('Dibuat Oleh', text='Dibuat Oleh')
715         tree.heading('Actions', text='Actions')
716
717         tree.column('ID', width=50)
718         tree.column('Judul', width=150)
719         tree.column('Mata Kuliah', width=120)
720         tree.column('Deadline', width=100)
721         tree.column('Penilik', width=100)
722         tree.column('Status', width=80)
723         tree.column('Dibuat Oleh', width=100)
724
725         tree.column('Actions', width=100)
726
727         scrollbar = tk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
728         tree.configure(scrollregion=scrollbar.set)
729
730         tree.grid(row=0, column=0, sticky='nsew')
731         scrollbar.grid(row=0, column=1, sticky='ns')
732
733         table_frame.grid_rowconfigure(0, weight=1)
734         table_frame.grid_columnconfigure(0, weight=1)
735
736
737
738     try:
739         tasks = self.load_json(self.tasks_file)
740
741         for task in tasks:
742             status = task.get('status', 'Pending')
743             if status != 'Pending':
744                 tree.insert('', tk.END, values=(
745                     task['id_tugas'],
746                     task['judul'],
747                     task['mata_kuliah'],
748                     task['deadline'],
749                     task['username_penilik'],
750                     status,
751                     task['dibuat_oleh'],
752                     edit
753                 ))
754     except Exception as e:
755         messagebox.showerror("Error", f"Error loading data: {str(e)}")
756
757     tree.bind("<Double-1>", lambda e: self.edit_task_dialog_admin(tree))

```

```

757     def view_all_tasks_gui(self):
758         self.clear_window()
759
760         header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
761         header_frame.pack(fill=tk.X)
762         header_frame.pack_propagate(False)
763
764         tk.Label(header_frame, text="SEMUA TUGAS", font=("Times New Roman", 16, "bold"), bg=self.admin_color, fg="white").pack(pady=15)
765
766         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")
767         back_button.place(x=10, y=15)
768
769         table_frame = tk.Frame(self.root)
770         table_frame.pack(fill=tk.BOTH, expand=True)
771
772         columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Status', 'Dibuat Oleh')
773         tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=28)
774
775         tree.heading('ID', text='ID')
776         tree.heading('Judul', text='Judul')
777         tree.heading('Mata Kuliah', text='Mata Kuliah')
778         tree.heading('Deadline', text='Deadline')
779         tree.heading('Pemilik', text='Pemilik')
780         tree.heading('Status', text='Status')
781         tree.heading('Dibuat Oleh', text='Dibuat Oleh')
782
783         tree.column('ID', width=50)
784         tree.column('Judul', width=200)
785         tree.column('Mata Kuliah', width=150)
786         tree.column('Deadline', width=100)
787         tree.column('Pemilik', width=100)
788         tree.column('Status', width=100)
789         tree.column('Dibuat Oleh', width=100)
790
791         scrollbar = tk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
792         tree.configure(yscroll=scrollbar.set)
793
794         tree.grid(row=0, column=0, sticky='nsew')
795         scrollbar.grid(row=0, column=1, sticky='ns')
796
797         table_frame.grid_rowconfigure(0, weight=1)
798         table_frame.grid_columnconfigure(0, weight=1)

```

```

800     try:
801         tasks = self.load_json(self.tasks_file)
802
803         for task in tasks:
804             tree.insert('', tk.END, values=(
805                 task['id_tugas'],
806                 task['judul'],
807                 task['mata_kuliah'],
808                 task['deadline'],
809                 task['username_pemilik'],
810                 task.get('status', 'Pending'),
811                 task['dibuat_oleh']
812             ))
813     except Exception as e:
814         messagebox.showerror("Error", f"Error loading data: {str(e)}")

```

```

817     def view_my_tasks_gui(self):
818         self.clear_window()
819
820         header_frame = tk.Frame(self.root, bg=self.student_color, height=60)
821         header_frame.pack(fill=tk.X)
822         header_frame.pack_propagate(False)
823
824         tk.Label(header_frame, text="TUGAS SAYA", font=("Times New Roman", 16, "bold"), bg=self.student_color, fg="white").pack(pady=15)
825
826         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")
827         back_button.place(x=10, y=15)
828
829         control_frame = tk.Frame(self.root, bg=self.bg_color)
830         control_frame.pack(pady=10)
831
832         tk.Button(control_frame, text="Refresh", font=("Times New Roman", 11, "bold"), bg="#607080", fg="white", width=20, command=self.view_my_tasks_gui, cursor="hand2").pack(side=tk.LEFT, padx=5)
833
834         table_frame = tk.Frame(self.root)
835         table_frame.pack(fill=tk.BOTH, expand=True)
836
837         columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Status', 'Dibuat Oleh', 'Actions')
838         tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)
839
840         tree.heading('ID', text='ID')
841         tree.heading('Judul', text='Judul')
842         tree.heading('Mata Kuliah', text='Mata Kuliah')
843         tree.heading('Deadline', text='Deadline')
844         tree.heading('Status', text='Status')
845         tree.heading('Dibuat Oleh', text='Dibuat Oleh')
846         tree.heading('Actions', text='Actions')
847
848         tree.column('ID', width=50)
849         tree.column('Judul', width=200)
850         tree.column('Mata Kuliah', width=150)
851         tree.column('Deadline', width=100)
852         tree.column('Status', width=100)
853         tree.column('Dibuat Oleh', width=100)
854         tree.column('Actions', width=100)
855
856         scrollbar = tk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
857         tree.configure(yscroll=scrollbar.set)
858
859         tree.grid(row=0, column=0, sticky='nsew')
860         scrollbar.grid(row=0, column=1, sticky='ns')
861
862         table_frame.grid_rowconfigure(0, weight=1)
863         table_frame.grid_columnconfigure(0, weight=1)

```

```

864     try:
865         tasks = self.load_json(self.tasks_file)
866         my_tasks = [task for task in tasks if task['username_pemilik'] == self.current_user]
867
868         for task in my_tasks:
869             actions = 'Edit | Ocate' if task['dibuat_oleh'] == 'mahasiswa' else 'View Only'
870             tree.insert('', tk.END, values=(
871                 task['id_tugas'],
872                 task['judul'],
873                 task['mata_kuliah'],
874                 task['deadline'],
875                 task.get('status', 'Pending'),
876                 task['dibuat_oleh'],
877                 actions
878             ))
879     except Exception as e:
880         messagebox.showerror("Error", f"Error loading data: {str(e)}")
881
882         tree.bind('<Double-1>', lambda e: self.view_task_details_gui(tree))

```

```

884     def check_deadline_gui(self):
885         try:
886             tasks = self.load_json(self.tasks_file)
887             today = datetime.now().date()
888
889             urgent_tasks = []
890             late_tasks = []
891
892             for task in tasks:
893                 if self.current_role == 'mahasiswa' and task['username_pemilik'] != self.current_user:
894                     continue
895
896                 try:
897                     deadline_date = datetime.strptime(str(task['deadline']), '%Y-%m-%d').date()
898                     days_left = (deadline_date - today).days
899
900                     if 0 <= days_left <= 2:
901                         urgent_tasks.append({
902                             'id': task['id_tugas'],
903                             'judul': task['judul'],
904                             'mata_kuliah': task['mata_kuliah'],
905                             'deadline': task['deadline'],
906                             'days_left': days_left,
907                             'pemilik': task['username_pemilik'],
908                             'status': task.get('status', 'Pending')
909                         })
910
911                     elif days_left < 0 and task.get('status', 'Pending') != 'Completed':
912                         late_tasks.append({
913                             'id': task['id_tugas'],
914                             'judul': task['judul'],
915                             'mata_kuliah': task['mata_kuliah'],
916                             'deadline': task['deadline'],
917                             'days_left': days_left,
918                             'pemilik': task['username_pemilik'],
919                             'status': task.get('status', 'Pending')
920                         })
921
922                 except:
923                     continue
924
925             dialog = tk.Toplevel(self.root)
926             dialog.title("Pengingat Deadline")
927             dialog.geometry("600x300")
928             dialog.configure(bg=self.bg_color)
929             dialog.transient(self.root)
930
931             tk.Label(dialog, text="⚠ PENGINGAT DEADLINE", font=("Times New Roman", 18, "bold"), bg=self.bg_color).pack(pady=20)
932
933             notebook = ttk.Notebook(dialog)
934             notebook.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)

```

```

935
936             urgent_frame = tk.Frame(notebook, bg=self.bg_color)
937             notebook.add(urgent_frame, text=f"Tugas Mendekat ({len(urgent_tasks)})")
938
939             if urgent_tasks:
940                 columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Hari Lagi', 'Status')
941                 tree_urgent = ttk.Treeview(urgent_frame, columns=columns, show='headings', height=10)
942
943                 for col in columns:
944                     tree_urgent.heading(col, text=col)
945                     tree_urgent.column(col, width=100)
946
947                     tree_urgent.column('Judul', width=150)
948                     tree_urgent.column('Mata Kuliah', width=120)
949
950                     scrollbar_urgent = ttk.Scrollbar(urgent_frame, orient=tk.VERTICAL, command=tree_urgent.yview)
951                     tree_urgent.configure(yscroll=scrollbar_urgent.set)
952
953                     tree_urgent.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
954                     scrollbar_urgent.pack(side=tk.RIGHT, fill=tk.Y)
955
956                     for task in urgent_tasks:
957                         days_left = "Hari Ini!" if task['days_left'] == 0 else f"{task['days_left']} hari"
958                         tree_urgent.insert('', tk.END, values=(
959                             task['id'],
960                             task['judul'],
961                             task['mata_kuliah'],
962                             task['deadline'],
963                             days_left,
964                             task['status']
965                         ))
966
967             else:
968                 tk.Label(urgent_frame, text="Tidak ada tugas mendekat (< 2 hari)", font=("Times New Roman", 14), bg=self.bg_color).pack(pady=50)
969
970             late_frame = tk.Frame(notebook, bg=self.bg_color)
971             notebook.add(late_frame, text=f"Tugas Terlambat ({len(late_tasks)})")

```

```

972
973             if late_tasks:
974                 columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Terlambat', 'Status')
975                 tree_late = ttk.Treeview(late_frame, columns=columns, show='headings', height=10)
976
977                 for col in columns:
978                     tree_late.heading(col, text=col)
979                     tree_late.column(col, width=100)
980
981                     tree_late.column('Judul', width=150)
982                     tree_late.column('Mata Kuliah', width=120)
983
984                     scrollbar_late = ttk.Scrollbar(late_frame, orient=tk.VERTICAL, command=tree_late.yview)
985                     tree_late.configure(yscroll=scrollbar_late.set)
986
987                     tree_late.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
988                     scrollbar_late.pack(side=tk.RIGHT, fill=tk.Y)
989
990                     for task in late_tasks:
991                         tree_late.insert('', tk.END, values=(
992                             task['id'],
993                             task['judul'],
994                             task['mata_kuliah'],
995                             task['deadline'],
996                             f'{abs(task["days_left"])} hari',
997                             task['status']
998                         ))
999
1000             else:
1001                 tk.Label(late_frame, text="Tidak ada tugas terlambat", font=("Times New Roman", 14), bg=self.bg_color).pack(pady=50)
1002
1003             tk.Button(dialog, text="Tutup", font=("Times New Roman", 12), bg="#607D8B", fg="white", width=20, command=dialog.destroy, cursor="hand2").pack(pady=20)
1004
1005         except Exception as e:
1006             messagebox.showerror("Error", f"Error: {str(e)}")
1007

```

```

1004     def view_task_details_gui(self, tree):
1005         try:
1006             selected_item = tree.selection()
1007             if not selected_item:
1008                 return
1009
1010             item = tree.item(selected_item[0])
1011             task_id = item['values'][0]
1012
1013             tasks = self.load_json(self.tasks_file)
1014             task = None
1015             for t in tasks:
1016                 if t['id_tugas'] == task_id:
1017                     task = t
1018                     break
1019
1020             if not task:
1021                 messagebox.showerror("Error", "Tugas tidak ditemukan!")
1022                 return
1023
1024             dialog = tk.Toplevel(self.root)
1025             dialog.title("Detail Tugas #(%s)" % task_id)
1026             dialog.geometry("600x500")
1027             dialog.configure(bg=self.bg_color)
1028             dialog.transient(self.root)
1029
1030             tk.Label(dialog, text="■ D E T A I L T U G A S #(%s)" % task_id, font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
1031
1032             notebook = ttk.Notebook(dialog)
1033             notebook.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)
1034
1035             info_frame = tk.Frame(notebook, bg=self.bg_color)
1036             notebook.add(info_frame, text="Informasi Dasar")
1037
1038             details = [
1039                 "Judul: (%s)" % task['judul'],
1040                 "Mata Kuliah: (%s)" % task['mata_kuliah'],
1041                 "Deadline: (%s)" % task['deadline'],
1042                 "Penulis: (%s)" % task['username_penulis'],
1043                 "Dibuat Oleh: (%s)" % task['dibuat_oleh'],
1044                 "Status: (%s)" % task.get('status', 'Pending'),
1045                 "Dibuat Pada: (%s)" % task.get('created_at', 'N/A')]
1046
1047
1048             for i, detail in enumerate(details):
1049                 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)
1050
1051             desc_frame = tk.Frame(notebook, bg=self.bg_color)
1052             notebook.add(desc_frame, text="Deskripsi")
1053
1054             desc_text = tk.Text(desc_frame, font=("Times New Roman", 11), wrap=tk.WORD, height=15)
1055             desc_scrollbar = tk.Scrollbar(desc_frame, command=desc_text.yview)
1056             desc_text.configure(yscrollcommand=desc_scrollbar.set)
1057
1058             desc_text.pack(side=tk.LEFT, fill=tk.BOTH, expand=True, padx=10, pady=10)
1059             desc_scrollbar.pack(side=tk.RIGHT, fill=tk.Y, pady=10)
1060
1061             desc_text.insert("1.0", task.get('deskripsi', 'Tidak ada deskripsi'))
1062             desc_text.configure(state='disabled')
1063
1064             button_frame = tk.Frame(dialog, bg=self.bg_color)
1065             button_frame.pack(pady=20)
1066
1067             user_can_edit = [
1068                 (self.current_role == 'admin') or
1069                 (self.current_role == 'mahasiswa' and task['dibuat_oleh'] == 'mahasiswa')
1070             ]
1071
1072
1073             if user_can_edit:
1074                 def edit_task():
1075                     dialog.destroy()
1076                     if self.current_role == 'admin':
1077                         self.edit_task_dialog_admin_by_id(task_id)
1078                     else:
1079                         self.edit_task_dialog_student(task_id)
1080
1081             def delete_task():
1082                 if messagebox.askyesno("Konfirmasi", "Apakah Anda yakin ingin menghapus tugas ini?"):
1083                     success = self.delete_task(task_id)
1084
1085                     if success:
1086                         dialog.destroy()
1087                         if self.current_role == 'admin':
1088                             self.manage_tasks_admin()
1089                         else:
1090                             self.view_my_tasks_gui()
1091
1092             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)
1093             tk.Button(button_frame, text="■ Hapus", font="(%s)" % "#4682B4", fg="white", width=15, command=delete_task, cursor="hand2").pack(side=tk.LEFT, padx=10)
1094
1095         except Exception as e:
1096             messagebox.showerror("Error", "Error menampilkan detail: %s" % str(e))
1097
1098
1099     def delete_task(self, task_id):
1100         try:
1101             tasks = self.load_json(self.tasks_file)
1102             new_tasks = [task for task in tasks if task['id_tugas'] != task_id]
1103
1104             if len(new_tasks) < len(tasks):
1105                 self.save_json(self.tasks_file, new_tasks)
1106                 messagebox.showinfo("Success", "Tugas berhasil dihapus!")
1107                 return True
1108             else:
1109                 messagebox.showerror("Error", "Tugas tidak ditemukan!")
1110             return False
1111
1112         except Exception as e:
1113             messagebox.showerror("Error", "Error menghapus tugas: %s" % str(e))
1114             return False

```

```

114 def edit_task_dialog_admin_by_id(self, task_id):
115     try:
116         tasks = self.load_json(self.tasks_file)
117         task = None
118         for t in tasks:
119             if t['id_tugas'] == task_id:
120                 task = t
121                 break
122
123         if not task:
124             messagebox.showerror("Error", "Tugas tidak ditemukan!")
125             return
126
127         users = self.load_json(self.users_file)
128         students = [user for user in users if user['role'] == 'mahasiswa']
129         student_list = [student['username'] for student in students]
130
131         dialog = tk.Toplevel(self.root)
132         dialog.title("Edit Tugas #%(task_id)s")
133         dialog.geometry("580x580")
134         dialog.configure(bg=self.bg_color)
135         dialog.transient(self.root)
136         dialog.grab_set()
137
138         tk.Label(dialog, text=f"EDIT TUGAS #%(task_id)s", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
139
140         form_frame = tk.Frame(dialog, bg=self.bg_color)
141         form_frame.pack(pady=10)
142
143         row = 0
144
145         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")
146         student_var = tk.StringVar(value=task['username_pemilik'])
147         student_combo = ttk.Combobox(form_frame, textvariable=student_var, values=student_list, width=27, font=("Times New Roman", 12))
148         student_combo.grid(row=row, column=1, padx=10, pady=10)
149         row += 1
150
151         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")
152         title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
153         title_entry.insert(0, task['judul'])
154         title_entry.grid(row=row, column=1, padx=10, pady=10)
155         row += 1
156
157         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")
158         course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
159         course_entry.insert(0, task['mata_kuliah'])
160         course_entry.grid(row=row, column=1, padx=10, pady=10)
161         row += 1

```

```

1163         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")
1164         deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1165         deadline_entry.insert(0, task['deadline'])
1166         deadline_entry.grid(row=row, column=1, padx=10, pady=10)
1167         row += 1
1168
1169         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")
1170         status_var = tk.StringVar(value=task.get('status', 'Pending'))
1171         status_combo = ttk.Combobox(form_frame, textvariable=status_var, values=['Pending', 'Completed'], width=27, font=("Times New Roman", 12))
1172         status_combo.grid(row=row, column=1, padx=10, pady=10)
1173         row += 1
1174
1175         tk.Label(form_frame, text="Keterangan:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
1176         desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
1177         desc_text.insert("1.0", Task.get('deskripsi', ''))
1178         desc_text.grid(row=row, column=1, padx=10, pady=10)
1179         row += 1
1180
1181         button_frame = tk.Frame(dialog, bg=self.bg_color)
1182         button_frame.pack(pady=20)

```

```

1184     def save_changes():
1185         student = student_var.get()
1186         title = title_entry.get().strip()
1187         course = course_entry.get().strip()
1188         deadline = deadline_entry.get().strip()
1189         status = status_var.get()
1190         description = desc_text.get("1.0", tk.END).strip()
1191
1192         if not all([student, title, course, deadline, status]):
1193             messagebox.showerror("Error", "Semua field harus diisi!")
1194             return
1195
1196         try:
1197             datetime.strptime(deadline, '%Y-%m-%d')
1198         except:
1199             messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
1200             return
1201
1202         try:
1203             task['username_pemilik'] = student
1204             task['judul'] = title
1205             task['mata_kuliah'] = course
1206             task['deadline'] = deadline
1207             task['status'] = status
1208             task['deskripsi'] = description
1209
1210             self.update_task_in_json(task)
1211             self.create_backup_files()
1212
1213             messagebox.showinfo("Success", "Tugas berhasil diperbarui!")
1214             dialog.destroy()
1215             self.manage_tasks_admin()
1216         except Exception as e:
1217             messagebox.showerror("Error", f"Error: {str(e)}")
1218
1219         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)
1220         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)
1221
1222     except Exception as e:
1223         messagebox.showerror("Error", f"Error: {str(e)}")

```

```

1225     def edit_task_dialog_student(self, task_id):
1226         try:
1227             tasks = self.load_json(self.tasks_file)
1228             task = None
1229             for t in tasks:
1230                 if t['id_tugas'] == task_id and t['username_pemilik'] == self.current_user:
1231                     task = t
1232                     break
1233
1234             if not task:
1235                 messagebox.showerror("Error", "Tugas tidak ditemukan atau tidak dapat diakses!")
1236             return
1237
1238             dialog = tk.Toplevel(self.root)
1239             dialog.title("#Edit Tugas #[task_id]")
1240             dialog.geometry("500x300")
1241             dialog.configure(bg=self.bg_color)
1242             dialog.transient(self.root)
1243             dialog.grab_set()
1244
1245             tk.Label(dialog, text="#EDIT TUGAS #[task_id]", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
1246
1247             form_frame = tk.Frame(dialog, bg=self.bg_color)
1248             form_frame.pack(pady=10)
1249
1250             row = 0
1251
1252             tk.Label(form_frame, text="Judul Kuliah:", font=("Times New Roman", 12), bg=self.bg_color).grid(row=row, column=0, padx=10, pady=10, sticky="w")
1253             title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1254             title_entry.insert(0, task['judul'])
1255             title_entry.grid(row=row, column=1, padx=10, pady=10)
1256
1257             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")
1258             deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1259             deadline_entry.insert(0, task['deadline'])
1260             deadline_entry.grid(row=row, column=1, padx=10, pady=10)
1261
1262             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")
1263             status_var = tk.StringVar(value=task.get('status', 'Pending'))
1264             status_combo = ttk.Combobox(form_frame, textvariable=status_var, values=['Pending', 'Completed'], width=27, font=("Times New Roman", 12))
1265             status_combo.grid(row=row, column=1, padx=10, pady=10)
1266
1267             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")
1268             desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
1269             desc_text.insert("1.0", task.get('deskripsi', ''))
1270             desc_text.grid(row=row, column=1, padx=10, pady=10)
1271
1272             button_frame = tk.Frame(dialog, bg=self.bg_color)
1273             button_frame.pack(pady=20)

```

```

1276
1277             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)
1278             tk.Button(button_frame, text="Batal", font=("Times New Roman", 12, "bold"), bg="red", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)
1279
1280             row += 1
1281
1282             button_frame = tk.Frame(dialog, bg=self.bg_color)
1283             button_frame.pack(pady=20)

```

```

1285     def save_changes():
1286         title = title_entry.get().strip()
1287         course = course_entry.get().strip()
1288         deadline = deadline_entry.get().strip()
1289         status = status_var.get()
1290         description = desc_text.get("1.0", tk.END).strip()
1291
1292         if not all([title, course, deadline, status]):
1293             messagebox.showerror("Error", "Semua field harus diisi!")
1294             return
1295
1296         try:
1297             datetime.strptime(deadline, '%Y-%m-%d')
1298         except:
1299             messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
1300             return
1301
1302         try:
1303             task['judul'] = title
1304             task[' Mata Kuliah'] = course
1305             task['Deadline'] = deadline
1306             task['Status'] = status
1307             task['deskripsi'] = description
1308
1309             self.update_task_in_json(task)
1310             self.create_backup_files()
1311
1312             messagebox.showinfo("Success", "Tugas berhasil diperbarui!")
1313             dialog.destroy()
1314             self.view_my_tasks_gui()
1315         except Exception as e:
1316             messagebox.showerror("Error", f"Error: {str(e)}")
1317
1318         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)
1319         tk.Button(button_frame, text="Batal", font=("Times New Roman", 12, "bold"), bg="red", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)
1320
1321     except Exception as e:
1322         messagebox.showerror("Error", f"Error: {str(e)}")

```

```

1324     def export_to_excel(self):
1325         try:
1326             users = self.load_json(self.users_file)
1327             pd.DataFrame(users).to_excel('export_users.xlsx', index=False)
1328
1329             tasks = self.load_json(self.tasks_file)
1330             pd.DataFrame(tasks).to_excel('export_tasks.xlsx', index=False)
1331
1332             messagebox.showinfo("Export Success", "Data berhasil dieksport ke Excel!")
1333         except Exception as e:
1334             messagebox.showerror("Export Error", f"Error saat export: {str(e)}")
1335
1336     def import_from_excel(self):
1337         try:
1338             file_path = simpledialog.askstring("Import", "Masukkan nama file Excel (users.xlsx):")
1339             if file_path and os.path.exists(file_path):
1340                 df = pd.read_excel(file_path)
1341                 data = df.to_dict('records')
1342                 self.save_json(self.users_file, data)
1343                 messagebox.showinfo("Import Success", "Data berhasil diimport dari Excel!")
1344             except Exception as e:
1345                 messagebox.showerror("Import Error", f"Error saat import: {str(e)}")
1346
1347     def logout(self):
1348         self.stop_reminder_system()
1349         self.current_user = None
1350         self.current_role = None
1351         self.show_welcome_screen()

```

```

1353     def add_student_dialog(self):
1354         dialog = tk.Toplevel(self.root)
1355         dialog.title("Tambah Mahasiswa")
1356         dialog.geometry("480x350")
1357         dialog.configure(bg=self.bg_color)
1358         dialog.transient(self.root)
1359         dialog.grab_set()
1360
1361         tk.Label(dialog, text="TAMBAH MAHASISWA", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
1362
1363         form_frame = tk.Frame(dialog, bg=self.bg_color)
1364         form_frame.pack(pady=10)
1365
1366         row = 0
1367
1368         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")
1369         username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1370         username_entry.grid(row=row, column=1, padx=10, pady=10)
1371         row += 1
1372
1373         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")
1374         password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30, show="")
1375         password_entry.grid(row=row, column=1, padx=10, pady=10)
1376         row += 1
1377
1378         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")
1379         confirm_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30, show="")
1380         confirm_entry.grid(row=row, column=1, padx=10, pady=10)
1381         row += 1
1382
1383         button_frame = tk.Frame(dialog, bg=self.bg_color)
1384         button_frame.pack(pady=20)

```

```

1386     def save_student():
1387         username = username_entry.get().strip()
1388         password = password_entry.get().strip()
1389         confirm = confirm_entry.get().strip()
1390
1391         if not all([username, password, confirm]):
1392             messagebox.showerror("Error", "Semua field harus diisi!")
1393             return
1394
1395         if len(username) < 3:
1396             messagebox.showerror("Error", "Username minimal 3 karakter!")
1397             return
1398
1399         if len(password) < 6:
1400             messagebox.showerror("Error", "Password minimal 6 karakter!")
1401             return
1402
1403         if password != confirm:
1404             messagebox.showerror("Error", "Password tidak cocok!")
1405             return
1406
1407         try:
1408             users = self.load_json(self.users_file)
1409             for user in users:
1410                 if user['username'] == username:
1411                     messagebox.showerror("Error", "Username sudah digunakan!")
1412                     return
1413
1414             new_student = {'username': username, 'password': self.hash_password(password), 'role': 'mahasiswa'}
1415             users.append(new_student)
1416             self.save_json(self.users_file, users)
1417             self.create_file()
1418             messagebox.showinfo("Success", f"Mahasiswa {username} berhasil ditambahkan!")
1419             dialog.destroy()
1420             self.manage_students()
1421         except Exception as e:
1422             messagebox.showerror("Error", f"Error: {str(e)}")
1423
1424         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)
1425         tk.Button(button_frame, text="Batal", font=("Times New Roman", 12), bg="#f0f0f0", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)

```

```

1427     def edit_student_dialog(self, tree):
1428         try:
1429             selected_item = tree.selection()
1430             if not selected_item:
1431                 return
1432
1433             item = tree.item(selected_item[0])
1434             username = item['values'][0]
1435
1436             users = self.load_json(self.users_file)
1437             student = None
1438             for user in users:
1439                 if user['username'] == username and user['role'] == 'mahasiswa':
1440                     student = user
1441                     break
1442
1443             if not student:
1444                 messagebox.showerror("Error", "Mahasiswa tidak ditemukan!")
1445                 return
1446
1447             dialog = tk.Toplevel(self.root)
1448             dialog.title("Edit Mahasiswa: ({username})".format(username=username))
1449             dialog.geometry("480x400")
1450             dialog.configure(bg=self.bg_color)
1451             dialog.transient(self.root)
1452             dialog.grab_set()
1453
1454             tk.Label(dialog, text="EDIT MAHASISWA: {username}", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
1455
1456             form_frame = tk.Frame(dialog, bg=self.bg_color)
1457             form_frame.pack(pady=10)
1458
1459             row = 0
1460
1461             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")
1462             tk.Label(form_frame, text=username, font="bold", bg=self.bg_color).grid(row=row, column=1, padx=10, pady=10, sticky="w")
1463             row += 1
1464
1465             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")
1466             new_username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1467             new_username_entry.grid(row=row, column=1, padx=10, pady=10)
1468             row += 1
1469
1470             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")
1471             new_password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30, show="")
1472             new_password_entry.grid(row=row, column=1, padx=10, pady=10)
1473             row += 1
1474
1475             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")
1476             confirm_password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30, show="")
1477             confirm_password_entry.grid(row=row, column=1, padx=10, pady=10)
1478             row += 1
1479
1480             button_frame = tk.Frame(dialog, bg=self.bg_color)
1481             button_frame.pack(pady=20)

```

```

1483     def save_changes():
1484         new_username = new_username_entry.get().strip()
1485         new_password = new_password_entry.get().strip()
1486         confirm_password = confirm_password_entry.get().strip()
1487
1488         if new_username:
1489             if len(new_username) < 3:
1490                 messagebox.showerror("Error", "Username minimal 3 karakter!")
1491             return
1492
1493             for user in users:
1494                 if user['username'] == new_username and user['username'] != username:
1495                     messagebox.showerror("Error", "Username sudah digunakan!")
1496                     return
1497
1498             if new_password:
1499                 if len(new_password) < 6:
1500                     messagebox.showerror("Error", "Password minimal 6 karakter!")
1501                     return
1502
1503                 if new_password != confirm_password:
1504                     messagebox.showerror("Error", "Password tidak cocok!")
1505                     return
1506
1507             try:
1508                 for i, user in enumerate(users):
1509                     if user['username'] == username:
1510                         if new_username:
1511                             users[i]['username'] = new_username
1512                         if new_password:
1513                             users[i]['password'] = self.hash_password(new_password)
1514                         break
1515
1516             self.save_json(self.users_file, users)
1517
1518             if new_username:
1519                 tasks = self.load_json(self.tasks_file)
1520                 for task in tasks:
1521                     if task['username_pemilik'] == username:
1522                         task['username_pemilik'] = new_username
1523
1524                 self.save_json(self.tasks_file, tasks)
1525
1526                 self.create_backup_files()
1527                 messagebox.showinfo("Success", "Data mahasiswa berhasil diperbarui!")
1528                 dialog.destroy()
1529                 self.manage_students()
1530             except Exception as e:
1531                 messagebox.showerror("Error", f"Error: {str(e)}")
1532

```

```

1532     def delete_student():
1533         if not messagebox.askyesno("Konfirmasi", f"Apakah Anda yakin ingin menghapus mahasiswa {username}?\nSemua tugas yang dimiliki mahasiswa ini juga akan dihapus!"):
1534             return
1535
1536         try:
1537             new_users = [user for user in users if user['username'] != username]
1538             self.save_json(self.users_file, new_users)
1539
1540             tasks = self.load_json(self.tasks_file)
1541             new_tasks = [task for task in tasks if task['username_pemilik'] != username]
1542             self.save_json(self.tasks_file, new_tasks)
1543
1544             self.create_backup_files()
1545             messagebox.showinfo("Success", f"Mahasiswa {username} berhasil dihapus!")
1546             dialog.destroy()
1547             self.manage_students()
1548         except Exception as e:
1549             messagebox.showerror("Error", f"Error: {str(e)}")
1550
1551         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)
1552         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)
1553         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)
1554
1555     except Exception as e:
1556         messagebox.showerror("Error", f"Error: {str(e)}")
1557

```

```

1557
1558     def edit_task_dialog_admin(self, tree):
1559         try:
1560             selected_item = tree.selection()
1561             if not selected_item:
1562                 return
1563
1564             item = tree.item(selected_item[0])
1565             task_id = item['values'][0]
1566             self.edit_task_dialog_admin_by_id(task_id)
1567         except Exception as e:
1568             messagebox.showerror("Error", f"Error: {str(e)}")
1569
1570     def main():
1571         root = tk.Tk()
1572         app = TaskReminderApp(root)
1573
1574         def on_closing():
1575             app.stop_reminder_system()
1576             root.destroy()
1577
1578         root.protocol("WM_DELETE_WINDOW", on_closing)
1579         root.mainloop()
1580
1581     if __name__ == "__main__":
1582         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>