

**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 :

```

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 :  $\leq 10$ MB 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

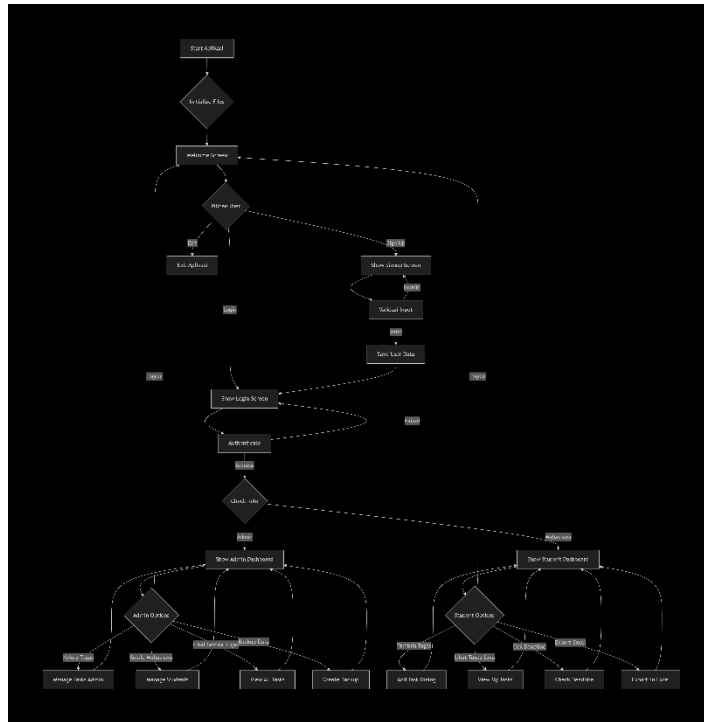
Kekurangan :

- Offline only
- Fitur dasar saja
- NO mobile version

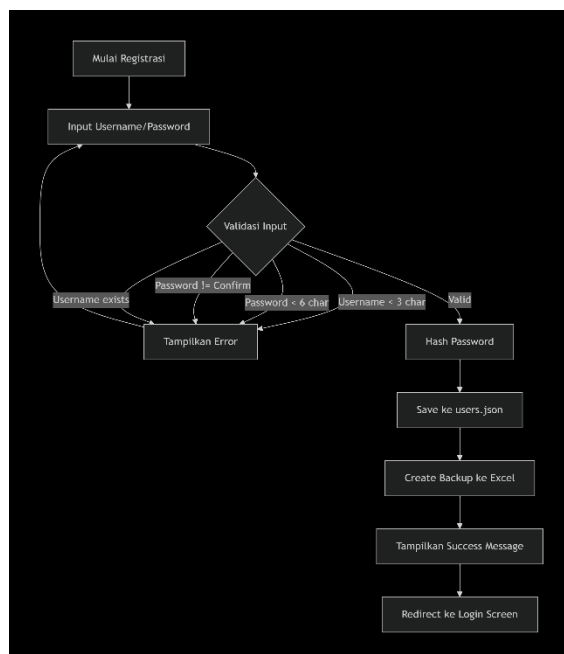


### E. Diagram Alur

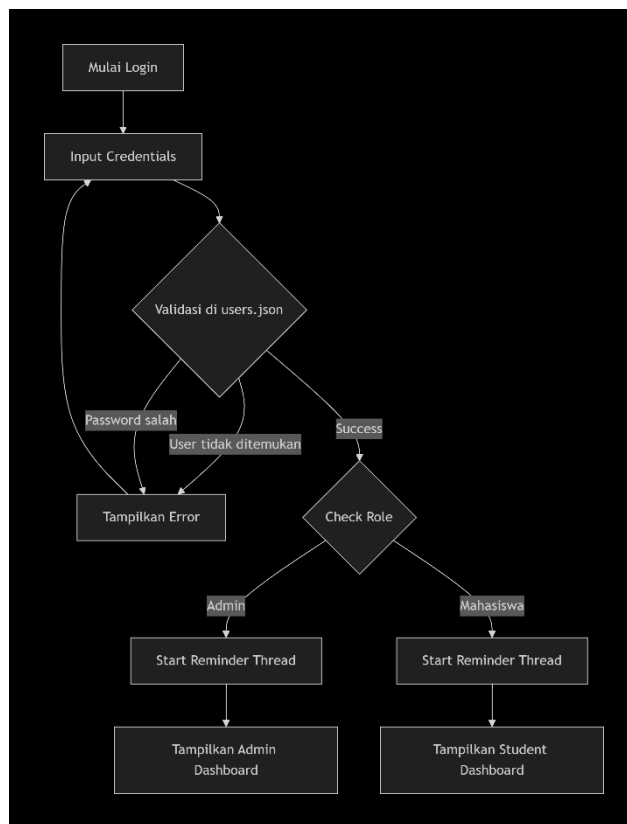
## 1. Alur utama



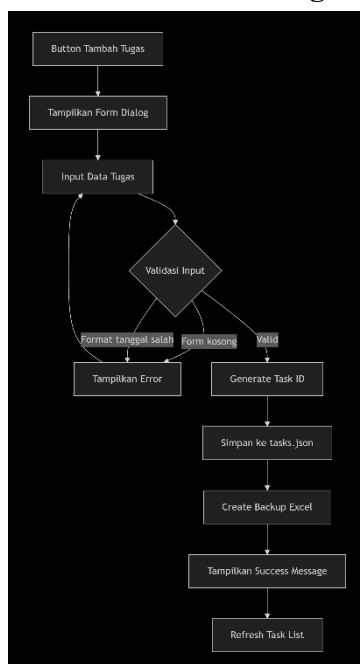
## 2. Alur Registrasi (Sign Up)



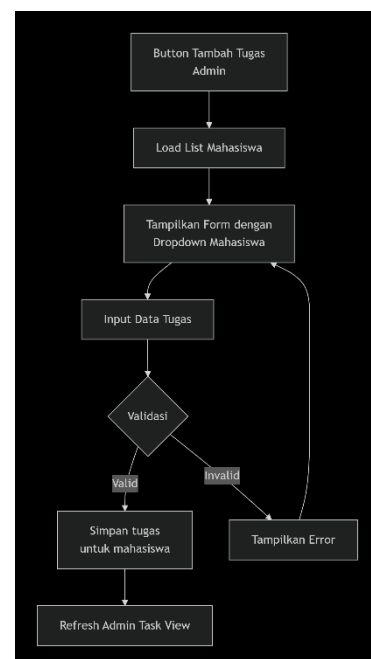
### 3. Alur Login



### 4. Alur Tambah Tugas

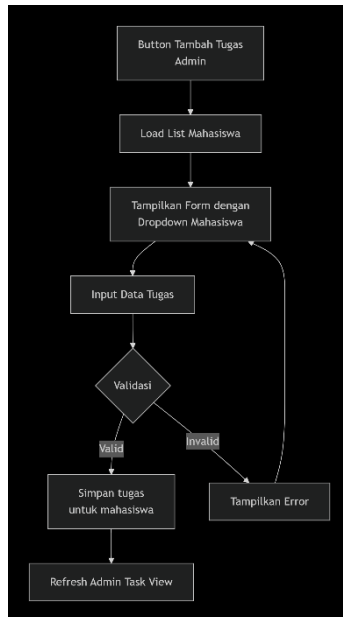


#### 4.1 Mahasiswa

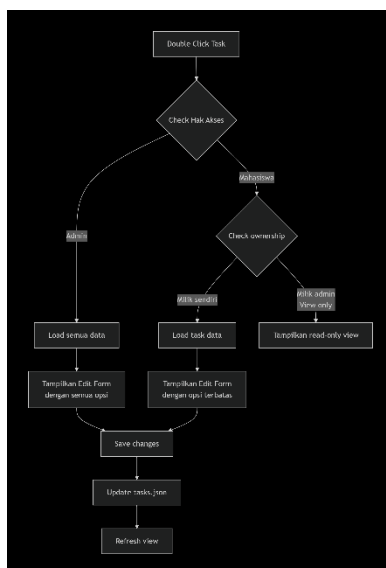


#### 4.2. Admin

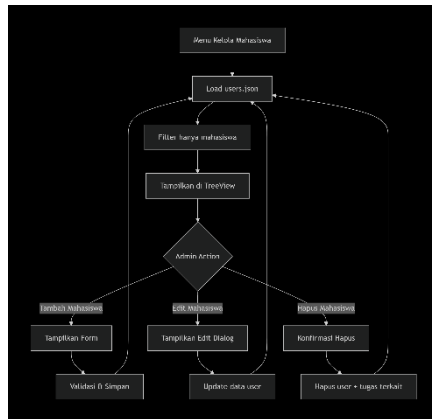
## 5. Alur Reminder System



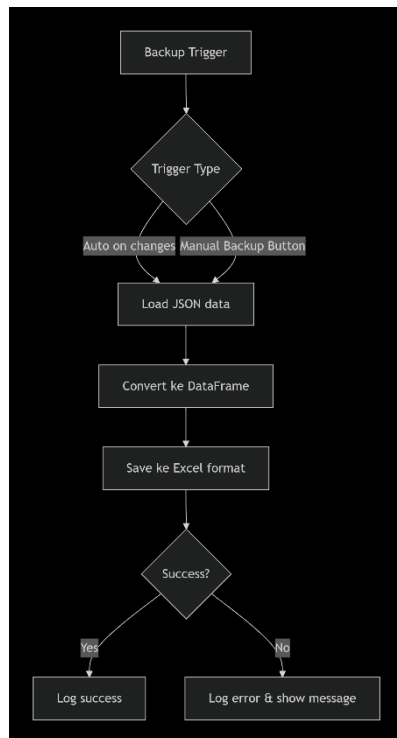
## 6. Alur Edit Tugas



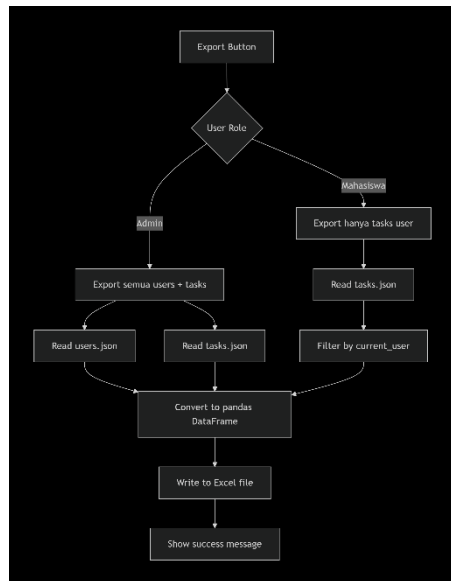
## 7. Alur Kelola Mahasiswa (Admin)



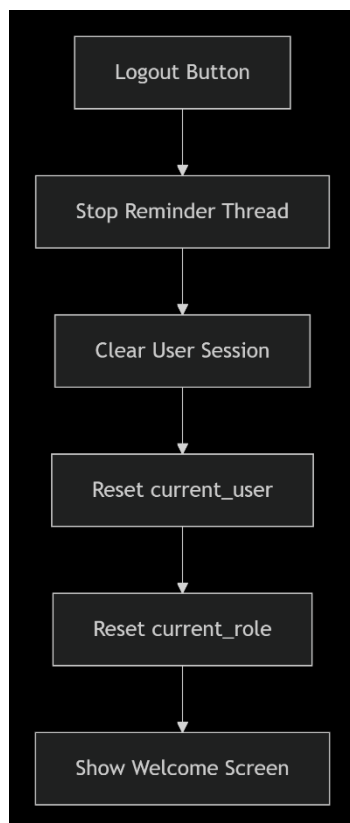
## 8. Alur Backup Sistem



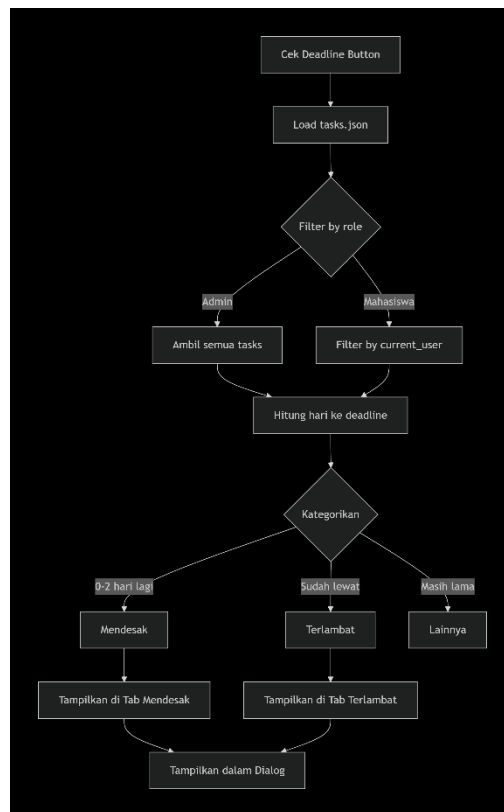
## 9. Alur Export to Excel



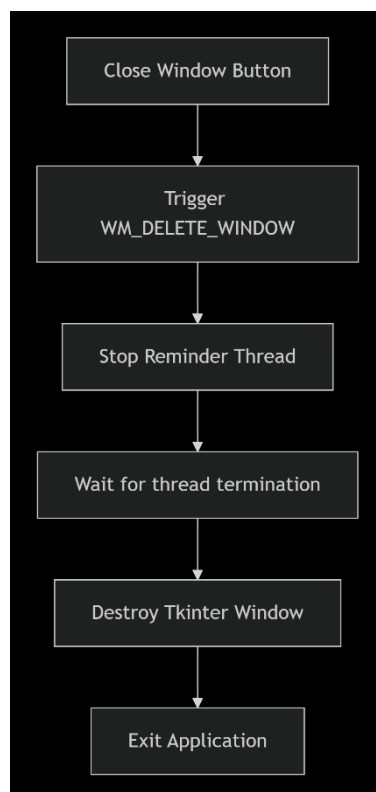
## 10. Alur Log Out



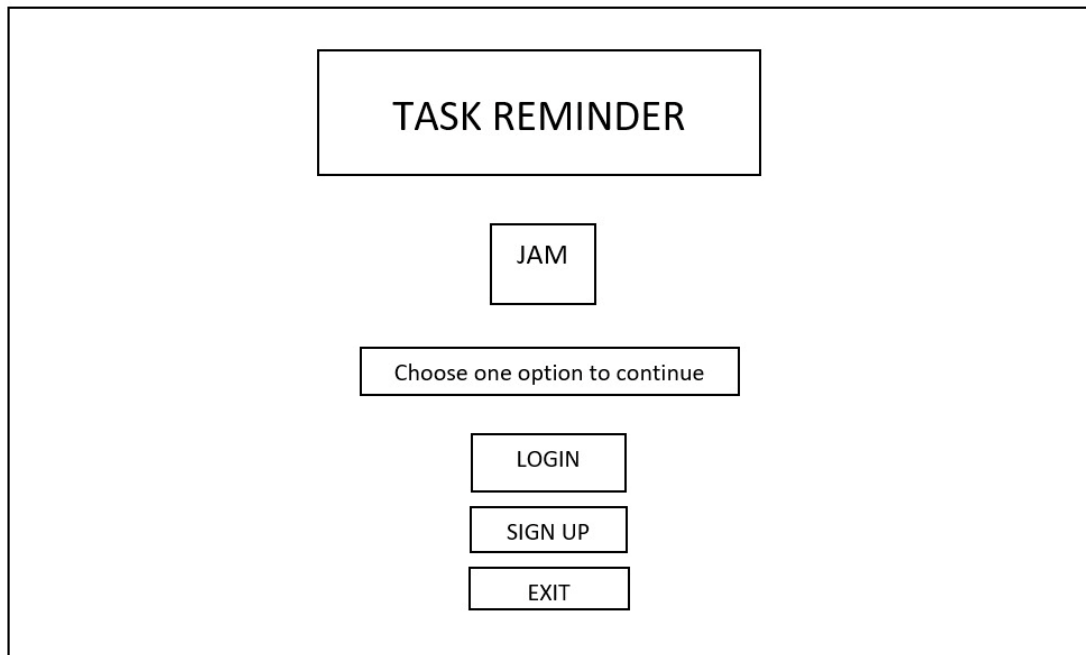
## 11. Alur Cek Deadline



## 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="#2c5c92")
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="#000000")
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="#2c5c92")
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="#2c5c92", 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="#f44336", 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}", font=("Times New Roman", 14), bg=self.admin_color, fg="white").pack(pady=5)
138
139     logout_button = tk.Button(header_frame, text="Logout", font=("Times New Roman", 14), bg=self.button_color, fg="white", width=20, 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

```
262 def check_automatic_reminders(self):
263     try:
264         tasks = self.load_json(self.tasks_file)
265         today = datetime.now().date()
266
267         for task in tasks:
268             if self.current_role == 'mahasiswa' and task['username_pemilik'] != self.current_user:
269                 continue
270
271             if task.get('status', 'Pending') == 'Completed':
272                 continue
273
274             try:
275                 deadline_date = datetime.strptime(str(task['deadline']), '%Y-%m-%d').date()
276                 days_left = (deadline_date - today).days
277
278                 if days_left == 0 and not task.get('notified_today', False):
279                     self.send_notification("📌 Deadline Hari Ini!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")
280                     task['notified_today'] = True
281                     self.update_task_in_json(task)
282                 elif days_left == 1 and not task.get('notified_tomorrow', False):
283                     self.send_notification("📌 Deadline Besok!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")
284                     task['notified_tomorrow'] = True
285                     self.update_task_in_json(task)
286                 elif days_left < 0 and not task.get('notified_late', False):
287                     self.send_notification("⚠️ Tugas Terlambat!", f"Tugas: {task['judul']}\nTerlambat: {abs(days_left)} hari")
288                     task['notified_late'] = True
289                     self.update_task_in_json(task)
290
291             except:
292                 continue
293         except Exception as e:
294             print(f"Error checking reminders: {e}")
295
296     def send_notification(self, title, message):
297         try:
298             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 X
C:\Users\bagum\Downloads\REMINDER TUGAS\TaskReminderApp1.py (preview 80)
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 = "#d8b48c"
28         self.button_hover = "#d4c085"
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()
```

```
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="#2c5c92")
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="#d8b48c")
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="#2c5c92")
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="#2c5c92", 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="#f4a336", fg="white", width=22, height=2, cursor="hand2", command=self.root.quit)
62     exit_button.pack(pady=5)
63
```

```
64 def show_login_screen(self):
65     self.clear_window()
66
67     header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
68     header_frame.pack(fill=tk.X)
69     header_frame.pack_propagate(False)
70
71     tk.Label(header_frame, text="LOGIN", font=("Times New Roman", 18, "bold"), bg=self.admin_color, fg="white").pack(pady=15)
72
73     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")
74     back_button.place(x=10, y=15)
75
76     form_frame = tk.Frame(self.root, bg=self.bg_color)
77     form_frame.pack(pady=50)
78
79     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")
80     self.username_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50)
81     self.username_entry.grid(row=0, column=1, padx=10, pady=35)
82
83     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")
84     self.password_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=50, show="*")
85     self.password_entry.grid(row=1, column=1, padx=10, pady=35)
86
87     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")
88     self.role_var = tk.StringVar(value="mahasiswa")
89     role_combo = ttk.Combobox(form_frame, textvariable=self.role_var, values=["mahasiswa", "admin"], width=48, font=("Times New Roman", 12), state="readonly")
90     role_combo.grid(row=2, column=1, padx=10, pady=35)
91
92     login_btn_frame = tk.Frame(self.root, bg=self.bg_color)
93     login_btn_frame.pack(pady=30)
94
95     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)
96     login_button.pack()
```

```

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

```

```

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 Welcome, (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 to Excel", 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=10, 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 Welcome, (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=10, pady=20)

```

```

191 def initialize_files(self):
192     if not os.path.exists(self.users_file):
193         users_data = [
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_thread:
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"Tugas: {task['judul']}\nDeadline: {task['deadline']}")
281                     task['notified_today'] = True
282                     self.update_task_in_json(task)
283                 elif days_left == 1 and not task.get('notified_tomorrow', False):
284                     self.send_notification("⚠ Deadline Besok!", f"Tugas: {task['judul']}\nDeadline: {task['deadline']}")
285                     task['notified_tomorrow'] = True
286                     self.update_task_in_json(task)
287                 elif days_left < 0 and not task.get('notified_late', False):
288                     self.send_notification("⚠ Tugas Terlambat!", f"Tugas: {task['judul']}\nTerlambat: {abs(days_left)} hari!")
289                     task['notified_late'] = True
290                     self.update_task_in_json(task)
291             except:
292                 continue
293     except Exception as e:
294         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)}")
354
355

```

```

356 def login(self):
357     username = self.username_entry.get()
358     password = self.password_entry.get()
359     role = self.role_var.get()
360
361     try:
362         users = self.load_json(self.users_file)
363         user_found = None
364         for user in users:
365             if user['username'] == username and user['role'] == role:
366                 user_found = user
367                 break
368
369         if user_found:
370             if self.hash_password(password) == user_found['password']:
371                 self.current_user = username
372                 self.current_role = role
373                 self.start_reminder_system()
374
375                 if role == 'admin':
376                     self.show_admin_dashboard()
377                 else:
378                     self.show_student_dashboard()
379             else:
380                 messagebox.showerror("Login Error", "Password salah!")
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

```

```

394 def add_task_dialog(self):
395     dialog = Tk.Toplevel(self.root)
396     dialog.title("Tambah Tugas Baru")
397     dialog.geometry("500x500")
398     dialog.configure(bg=self.bg_color)
399     dialog.transient(self.root)
400     dialog.grab_set()
401
402     tk.Label(dialog, text="TAMBAH TUGAS BARU", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
403
404     form_frame = tk.Frame(dialog, bg=self.bg_color)
405     form_frame.pack(pady=10)
406
407     row = 0
408
409     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")
410     title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
411     title_entry.grid(row=row, column=1, padx=10, pady=10)
412     row += 1
413
414     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")
415     course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
416     course_entry.grid(row=row, column=1, padx=10, pady=10)
417     row += 1
418
419     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")
420     deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
421     deadline_entry.insert(0, datetime.now().strftime("%Y-%m-%d"))
422     deadline_entry.grid(row=row, column=1, padx=10, pady=10)
423     row += 1
424
425     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")
426     desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=6)
427     desc_text.grid(row=row, column=1, padx=10, pady=10)
428     row += 1
429
430     button_frame = tk.Frame(dialog, bg=self.bg_color)
431     button_frame.pack(pady=20)

```

```

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-DD")
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             'diambil_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, "bold"), bg="#f4a336", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)
475

```

```

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

```

```

498     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")
499     student_var = tk.StringVar(values=student_list[0] if student_list else "")
500     student_combo = tk.Combobox(form_frame, textvariable=student_var, values=student_list, width=27, font=("Times New Roman", 12))
501     student_combo.grid(row=row, column=1, padx=10, pady=10)
502     row += 1
503
504     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")
505     title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
506     title_entry.grid(row=row, column=1, padx=10, pady=10)
507     row += 1
508
509     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")
510     course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
511     course_entry.grid(row=row, column=1, padx=10, pady=10)
512     row += 1
513
514     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")
515     deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
516     deadline_entry.insert(0, datetime.now().strftime("%Y-%m-%d"))
517     deadline_entry.grid(row=row, column=1, padx=10, pady=10)
518     row += 1
519
520     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")
521     desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
522     desc_text.grid(row=row, column=1, padx=10, pady=10)
523     row += 1
524
525     button_frame = tk.Frame(dialog, bg=self.bg_color)
526     button_frame.pack(pady=20)
527

```

```

528 def save_task():
529     student = student_var.get()
530     title = title_entry.get().strip()
531     course = course_entry.get().strip()
532     deadline = deadline_entry.get().strip()
533     description = desc_text.get("1.0", tk.END).strip()
534
535     if not all([student, title, course, deadline]):
536         messagebox.showerror("Error", "Semua field harus diisi!")
537         return
538
539     try:
540         datetime.strptime(deadline, '%Y-%m-%d')
541     except:
542         messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
543         return
544
545     try:
546         tasks = self.load_json(self.tasks_file)
547         new_task = {
548             'id_tugas': self.get_next_task_id(),
549             'judul': title,
550             'deskripsi': description,
551             'mata_kuliah': course,
552             'deadline': deadline,
553             'username_pemilik': student,
554             'diambil_oleh': 'admin',
555             'status': 'Pending',
556             'created_at': datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
557             'notified_today': False,
558             'notified_tomorrow': False,
559             'notified_late': False
560         }
561         tasks.append(new_task)
562         self.save_json(self.tasks_file, tasks)
563         self.create_backup_files()
564         messagebox.showinfo("Success", "Tugas berhasil ditambahkan!")
565         dialog.destroy()
566         self.manage_tasks_admin()
567     except Exception as e:
568         messagebox.showerror("Error", f"Error: {str(e)}")
569
570 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)
571 tk.Button(button_frame, text="Batal", font=("Times New Roman", 12, "bold"), bg="#f4a336", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)
572

```

```

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=i, column=0, 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 <= 7:
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=i, column=0, padx=20)
631     except Exception as e:
632         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="#667788", 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     tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)
657
658     tree.heading('Username', text='Username')
659     tree.heading('Role', text='Role')
660     tree.heading('Actions', text='Actions')
661
662     tree.column('Username', width=200)
663     tree.column('Role', width=100)
664     tree.column('Actions', width=150)
665
666     scrollbar = ttk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
667     tree.configure(yscroll=scrollbar.set)
668
669     tree.grid(row=0, column=0, sticky='nsew')
670     scrollbar.grid(row=0, column=1, sticky='ns')
671
672     table_frame.grid_rowconfigure(0, weight=1)
673     table_frame.grid_columnconfigure(0, weight=1)
674
675     try:
676         users = self.load_json(self.users_file)
677         students = [user for user in users if user['role'] == 'mahasiswa']
678
679         for student in students:
680             tree.insert('', tk.END, values=(student['username'], student['role'], 'delete atau edit'))
681     except Exception as e:
682         messagebox.showerror("Error", f"Error loading data: {str(e)}")
683
684     tree.bind('<Double-1>', lambda e: self.edit_student_dialog(tree))

```

```

686 def manage_tasks_admin(self):
687     self.clear_window()
688
689     header_frame = tk.Frame(self.root, bg=self.admin_color, height=60)
690     header_frame.pack(fill=tk.X)
691     header_frame.pack_propagate(False)
692
693     tk.Label(header_frame, text="KELOLA TUGAS (ADMIN)", font=("Times New Roman", 16, "bold"), bg=self.admin_color, fg="white").pack(pady=15)
694
695     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")
696     back_button.place(x=10, y=15)
697
698     control_frame = tk.Frame(self.root, bg=self.bg_color)
699     control_frame.pack(pady=10)
700
701     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)
702     tk.Button(control_frame, text="Refresh", font=("Times New Roman", 11, "bold"), bg="#667788", fg="white", width=20, command=self.manage_tasks_admin, cursor="hand2").pack(side=tk.LEFT, padx=5)
703
704     table_frame = tk.Frame(self.root)
705     table_frame.pack(pady=10, padx=20, fill=tk.BOTH, expand=True)
706
707     columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Pemilik', 'Status', 'Dibuat Oleh', 'Actions')
708     tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)
709
710     tree.heading('ID', text='ID')
711     tree.heading('Judul', text='Judul')
712     tree.heading('Mata Kuliah', text='Mata Kuliah')
713     tree.heading('Deadline', text='Deadline')
714     tree.heading('Pemilik', text='Pemilik')
715     tree.heading('Status', text='Status')
716     tree.heading('Dibuat Oleh', text='Dibuat Oleh')
717     tree.heading('Actions', text='Actions')
718
719     tree.column('ID', width=30)
720     tree.column('Judul', width=150)
721     tree.column('Mata Kuliah', width=120)
722     tree.column('Deadline', width=100)
723     tree.column('Pemilik', width=100)
724     tree.column('Status', width=80)
725     tree.column('Dibuat Oleh', width=100)
726     tree.column('Actions', width=100)
727
728     scrollbar = ttk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
729     tree.configure(yscroll=scrollbar.set)
730
731     tree.grid(row=0, column=0, sticky='nsew')
732     scrollbar.grid(row=0, column=1, sticky='ns')
733
734     table_frame.grid_rowconfigure(0, weight=1)
735     table_frame.grid_columnconfigure(0, weight=1)

```

```

737     try:
738         tasks = self.load_json(self.tasks_file)
739
740         for task in tasks:
741             status = task.get('status', 'Pending')
742             tree.insert('', tk.END, values=(
743                 task['id_tugas'],
744                 task['judul'],
745                 task['mata_kuliah'],
746                 task['deadline'],
747                 task['username_pemilik'],
748                 status,
749                 task['dibuat_oleh'],
750                 'Edit'
751             ))
752     except Exception as e:
753         messagebox.showerror("Error", f"Error loading data: {str(e)}")
754
755     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="SEKUA 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(pady=10, padx=20, fill=tk.BOTH, expand=True)
771
772     columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Pemilik', 'Status', 'Dibuat Oleh')
773     tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=20)
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 = ttk.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)}")

```

```

816 def view_my_tasks_gui(self):
817     self.clear_window()
818
819     header_frame = tk.Frame(self.root, bg=self.student_color, height=60)
820     header_frame.pack(fill=tk.X)
821     header_frame.pack_propagate(False)
822
823     tk.Label(header_frame, text="TUGAS SAYA", font=("Times New Roman", 16, "bold"), bg=self.student_color, fg="white").pack(pady=15)
824
825     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")
826     back_button.place(x=10, y=15)
827
828     control_frame = tk.Frame(self.root, bg=self.bg_color)
829     control_frame.pack(pady=10)
830
831     tk.Button(control_frame, text="Refresh", font=("Times New Roman", 11, "bold"), bg="#607088", fg="white", width=20, command=self.view_my_tasks_gui, cursor="hand2").pack(side=tk.LEFT, padx=5)
832
833     table_frame = tk.Frame(self.root)
834     table_frame.pack(pady=10, padx=20, fill=tk.BOTH, expand=True)
835
836     columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Status', 'Dibuat Oleh', 'Actions')
837     tree = ttk.Treeview(table_frame, columns=columns, show='headings', height=15)
838
839     tree.heading('ID', text='ID')
840     tree.heading('Judul', text='Judul')
841     tree.heading('Mata Kuliah', text='Mata Kuliah')
842     tree.heading('Deadline', text='Deadline')
843     tree.heading('Status', text='Status')
844     tree.heading('Dibuat Oleh', text='Dibuat Oleh')
845     tree.heading('Actions', text='Actions')
846
847     tree.column('ID', width=50)
848     tree.column('Judul', width=200)
849     tree.column('Mata Kuliah', width=150)
850     tree.column('Deadline', width=100)
851     tree.column('Status', width=100)
852     tree.column('Dibuat Oleh', width=100)
853     tree.column('Actions', width=100)
854
855     scrollbar = ttk.Scrollbar(table_frame, orient=tk.VERTICAL, command=tree.yview)
856     tree.configure(yscroll=scrollbar.set)
857
858     tree.grid(row=0, column=0, sticky='nsew')
859     scrollbar.grid(row=0, column=1, sticky='ns')
860
861     table_frame.grid_rowconfigure(0, weight=1)
862     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 | Delete" 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))
883
884

```



```

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                 elif days_left < 0 and task.get('status', 'Pending') != 'Completed':
911                     late_tasks.append({
912                         'id': task['id_tugas'],
913                         'judul': task['judul'],
914                         'mata_kuliah': task['mata_kuliah'],
915                         'deadline': task['deadline'],
916                         'days_left': days_left,
917                         'pemilik': task['username_pemilik'],
918                         'status': task.get('status', 'Pending')
919                     })
920             except:
921                 continue
922
923         dialog = tk.Toplevel(self.root)
924         dialog.title("Pengingat Deadline")
925         dialog.geometry("600x300")
926         dialog.config(bg=self.bg_color)
927         dialog.transient(self.root)
928
929         tk.Label(dialog, text="⚠ PENGINGAT DEADLINE", font=("Times New Roman", 18, "bold"), bg=self.bg_color).pack(pady=20)
930
931         notebook = ttk.Notebook(dialog)
932         notebook.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)

```

```

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

```

```

976         if late_tasks:
977             columns = ('ID', 'Judul', 'Mata Kuliah', 'Deadline', 'Terlambat', 'Status')
978             tree_late = ttk.Treeview(late_frame, columns=columns, show='headings', height=10)
979
980             for col in columns:
981                 tree_late.heading(col, text=col)
982                 tree_late.column(col, width=100)
983
984             tree_late.column('Judul', width=150)
985             tree_late.column('Mata Kuliah', width=120)
986
987             scrollbar_late = ttk.Scrollbar(late_frame, orient=tk.VERTICAL, command=tree_late.yview)
988             tree_late.config(yscroll=scrollbar_late.set)
989
990             tree_late.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
991             scrollbar_late.pack(side=tk.RIGHT, fill=tk.Y)
992
993             for task in late_tasks:
994                 tree_late.insert('', tk.END, values=(
995                     task['id'],
996                     task['judul'],
997                     task['mata_kuliah'],
998                     task['deadline'],
999                     f"abs(task['days_left']) hari",
1000                     task['status']
1001                 ))
1002         else:
1003             tk.Label(late_frame, text="Tidak ada tugas terlambat", font=("Times New Roman", 14), bg=self.bg_color).pack(pady=50)
1004
1005         tk.Button(dialog, text="Tutup", font=("Times New Roman", 12), bg="#607088", fg="white", width=20, command=dialog.destroy, cursor="hand2").pack(pady=20)
1006     except Exception as e:
1007         messagebox.showerror("Error", f"Error: {str(e)}")

```

```

1004 def view_task_details_gui(self, tree):
1005     try:
1006         selected_item = tree.selection()
1007         if not selected_item:
1008             return
1009         item = tree.item(selected_item[0])
1010         task_id = item['values'][10]
1011
1012         tasks = self.load_json(self.tasks_file)
1013         task = None
1014         for t in tasks:
1015             if t['id_tugas'] == task_id:
1016                 task = t
1017                 break
1018
1019         if not task:
1020             messagebox.showerror("Error", "Tugas tidak ditemukan!")
1021             return
1022
1023         dialog = tk.Toplevel(self.root)
1024         dialog.title(f"Detail Tugas #{task_id}")
1025         dialog.geometry("600x500")
1026         dialog.configure(bg=self.bg_color)
1027         dialog.transient(self.root)
1028
1029         tk.Label(dialog, text=f"DETAIL TUGAS #{task_id}", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
1030
1031         notebook = ttk.Notebook(dialog)
1032         notebook.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)
1033
1034         info_frame = tk.Frame(notebook, bg=self.bg_color)
1035         notebook.add(info_frame, text="Informasi Dasar")
1036
1037         details = [
1038             f"Judul: {task['judul']}",
1039             f"Nota kuliah: {task['nota_kuliah']}",
1040             f"Deadline: {task['deadline']}",
1041             f"Pemilik: {task['username_pemilik']}",
1042             f"Dibuat Oleh: {task['dibuat_oleh']}",
1043             f"Status: {task.get('status', 'Pending')}",
1044             f"Dibuat Pada: {task.get('created_at', 'N/A')}"
1045         ]
1046     
```

```

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     if user_can_edit:
1073         def edit_task():
1074             dialog.destroy()
1075             if self.current_role == 'admin':
1076                 self.edit_task_dialog_admin_by_id(task_id)
1077             else:
1078                 self.edit_task_dialog_student(task_id)
1079
1080         def delete_task():
1081             if messagebox.askyesno("Konfirmasi", "Apakah Anda yakin ingin menghapus tugas ini?"):
1082                 success = self.delete_task(task_id)
1083                 if success:
1084                     dialog.destroy()
1085                     if self.current_role == 'admin':
1086                         self.manage_tasks_admin()
1087                     else:
1088                         self.view_my_tasks_gui()
1089
1090         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)
1091         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)
1092
1093         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)
1094
1095     except Exception as e:
1096         messagebox.showerror("Error", f"Error menampilkan detail: {str(e)}")
1097     
```

```

1098 def delete_task(self, task_id):
1099     try:
1100         tasks = self.load_json(self.tasks_file)
1101         new_tasks = [task for task in tasks if task['id_tugas'] != task_id]
1102
1103         if len(new_tasks) < len(tasks):
1104             self.save_json(self.tasks_file, new_tasks)
1105             messagebox.showinfo("Success", "Tugas berhasil dihapus!")
1106             return True
1107         else:
1108             messagebox.showerror("Error", "Tugas tidak ditemukan!")
1109             return False
1110     except Exception as e:
1111         messagebox.showerror("Error", f"Error menghapus tugas: {str(e)}")
1112         return False
1113     
```

```

1114 def edit_task_dialog_admin_by_id(self, task_id):
1115     try:
1116         tasks = self.load_json(self.tasks_file)
1117         task = None
1118         for t in tasks:
1119             if t['id_tugas'] == task_id:
1120                 task = t
1121                 break
1122         if not task:
1123             messagebox.showerror("Error", "Tugas tidak ditemukan!")
1124             return
1125         users = self.load_json(self.users_file)
1126         students = [user for user in users if user['role'] == 'mahasiswa']
1127         student_list = [student['username'] for student in students]
1128         dialog = tk.Toplevel(self.root)
1129         dialog.title(f"Edit Tugas #{task_id}")
1130         dialog.geometry("500x500")
1131         dialog.configure(bg=self.bg_color)
1132         dialog.transient(self.root)
1133         dialog.grab_set()
1134         tk.Label(dialog, text=f"EDIT TUGAS #{task_id}", font=("Times New Roman", 16, "bold"), bg=self.bg_color).pack(pady=20)
1135         form_frame = tk.Frame(dialog, bg=self.bg_color)
1136         form_frame.pack(pady=10)
1137         row = 0
1138         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")
1139         student_var = tk.StringVar(value=task['username_pemilik'])
1140         student_combo = ttk.Combobox(form_frame, textvariable=student_var, values=student_list, width=27, font=("Times New Roman", 12))
1141         student_combo.grid(row=row, column=1, padx=10, pady=10)
1142         row += 1
1143         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")
1144         title_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1145         title_entry.insert(0, task['judul'])
1146         title_entry.grid(row=row, column=1, padx=10, pady=10)
1147         row += 1
1148         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")
1149         course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1150         course_entry.insert(0, task['mata_kuliah'])
1151         course_entry.grid(row=row, column=1, padx=10, pady=10)
1152         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         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")
1169         status_var = tk.StringVar(value=task.get('status', 'Pending'))
1170         status_combo = ttk.Combobox(form_frame, textvariable=status_var, values=['Pending', 'Completed'], width=27, font=("Times New Roman", 12))
1171         status_combo.grid(row=row, column=1, padx=10, pady=10)
1172         row += 1
1173         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")
1174         desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
1175         desc_text.insert(1.0, task.get('deskripsi', ''))
1176         desc_text.grid(row=row, column=1, padx=10, pady=10)
1177         row += 1
1178         button_frame = tk.Frame(dialog, bg=self.bg_color)
1179         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     if not all([student, title, course, deadline, status]):
1192         messagebox.showerror("Error", "Semua field harus diisi!")
1193         return
1194     try:
1195         datetime.strptime(deadline, '%Y-%m-%d')
1196     except:
1197         messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
1198         return
1199     try:
1200         task['username_pemilik'] = student
1201         task['judul'] = title
1202         task['mata_kuliah'] = course
1203         task['deadline'] = deadline
1204         task['status'] = status
1205         task['deskripsi'] = description
1206         self.update_task_in_json(task)
1207         self.create_backup_files()
1208         messagebox.showinfo("Success", "Tugas berhasil diperbarui!")
1209         dialog.destroy()
1210         self.manage_tasks_admin()
1211     except Exception as e:
1212         messagebox.showerror("Error", f"Error: {str(e)}")
1213     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)
1214     tk.Button(button_frame, text="Batal", font=("Times New Roman", 12, "bold"), bg="#f44336", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)
1215 except Exception as e:
1216     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_penilik'] == 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 #{} {}".format(task_id, task['judul']))
1240         dialog.geometry("800x580")
1241         dialog.configure(bg=self.bg_color)
1242         dialog.transient(self.root)
1243         dialog.grab_set()
1244
1245         tk.Label(dialog, text="EDIT TUGAS #{} {}".format(task_id, task['judul']), 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 Tugas:", 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         row += 1
1257
1258         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")
1259         course_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1260         course_entry.insert(0, task['mata_kuliah'])
1261         course_entry.grid(row=row, column=1, padx=10, pady=10)
1262         row += 1
1263
1264         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")
1265         deadline_entry = tk.Entry(form_frame, font=("Times New Roman", 12), width=30)
1266         deadline_entry.insert(0, task['deadline'])
1267         deadline_entry.grid(row=row, column=1, padx=10, pady=10)
1268         row += 1
1269
1270         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")
1271         status_var = tk.StringVar(value=task.get('status', 'Pending'))
1272         status_combo = tk.Comboobox(form_frame, textvariable=status_var, values=('Pending', 'Completed'), width=27, font=("Times New Roman", 12))
1273         status_combo.grid(row=row, column=1, padx=10, pady=10)
1274         row += 1
1275
1276         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")
1277         desc_text = tk.Text(form_frame, font=("Times New Roman", 12), width=30, height=5)
1278         desc_text.insert("1.0", task.get('deskripsi', ''))
1279         desc_text.grid(row=row, column=1, padx=10, pady=10)
1280         row += 1
1281
1282         button_frame = tk.Frame(dialog, bg=self.bg_color)
1283         button_frame.pack(pady=20)
1284
1285     except Exception as e:
1286         messagebox.showerror("Error", f"Error: {str(e)}")
1287
1288 def save_changes():
1289     title = title_entry.get().strip()
1290     course = course_entry.get().strip()
1291     deadline = deadline_entry.get().strip()
1292     status = status_var.get()
1293     description = desc_text.get("1.0", tk.END).strip()
1294
1295     if not all([title, course, deadline, status]):
1296         messagebox.showerror("Error", "Semua field harus diisi!")
1297         return
1298
1299     try:
1300         datetime.strptime(deadline, '%Y-%m-%d')
1301     except:
1302         messagebox.showerror("Error", "Format tanggal salah! Gunakan YYYY-MM-DD")
1303         return
1304
1305     try:
1306         task['judul'] = title
1307         task['mata_kuliah'] = course
1308         task['deadline'] = deadline
1309         task['status'] = status
1310         task['deskripsi'] = description
1311
1312         self.update_task_in_json(task)
1313         self.create_backup_files()
1314
1315         messagebox.showinfo("Success", "Tugas berhasil diperbarui!")
1316         dialog.destroy()
1317         self.view_my_tasks_gui()
1318     except Exception as e:
1319         messagebox.showerror("Error", f"Error: {str(e)}")
1320
1321 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)
1322 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)
1323
1324 except Exception as e:
1325     messagebox.showerror("Error", f"Error: {str(e)}")
1326
1327 def export_to_excel(self):
1328     try:
1329         users = self.load_json(self.users_file)
1330         pd.DataFrame(users).to_excel("export_users.xlsx", index=False)
1331
1332         tasks = self.load_json(self.tasks_file)
1333         pd.DataFrame(tasks).to_excel("export_tasks.xlsx", index=False)
1334
1335         messagebox.showinfo("Export Success", "Data berhasil diekspor ke Excel!")
1336     except Exception as e:
1337         messagebox.showerror("Export Error", f"Error saat export: {str(e)}")
1338
1339 def import_from_excel(self):
1340     try:
1341         file_path = simpledialog.askstring("Import", "Masukkan nama file Excel (users.xlsx):")
1342         if file_path and os.path.exists(file_path):
1343             df = pd.read_excel(file_path)
1344             data = df.to_dict('records')
1345             self.save_json(self.users_file, data)
1346             messagebox.showinfo("Import Success", "Data berhasil diimport dari Excel!")
1347         else:
1348             messagebox.showerror("Import Error", f"Error saat import: {str(e)}")
1349     except Exception as e:
1350         messagebox.showerror("Import Error", f"Error saat import: {str(e)}")
1351
1352 def logout(self):
1353     self.stop_reminder_system()
1354     self.current_user = None
1355     self.current_role = None
1356     self.show_welcome_screen()

```

```

1353 def add_student_dialog(self):
1354     dialog = tk.Toplevel(self.root)
1355     dialog.title("Tambah Mahasiswa")
1356     dialog.geometry("400x300")
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)

```

```

1385
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_backup_files()
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, "bold"), bg="#FFA330", fg="white", width=15, command=dialog.destroy, cursor="hand2").pack(side=tk.LEFT, padx=10)

```

```

1426
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(f"Edit Mahasiswa: {username}")
1449         dialog.geometry("400x400")
1450         dialog.configure(bg=self.bg_color)
1451         dialog.transient(self.root)
1452         dialog.grab_set()
1453
1454         tk.Label(dialog, text=f"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=("Times New Roman", 12, "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             self.save_json(self.tasks_file, tasks)
1524
1525         self.create_backup_files()
1526         messagebox.showinfo("Success", "Data mahasiswa berhasil diperbarui!")
1527         dialog.destroy()
1528         self.manage_students()
1529     except Exception as e:
1530         messagebox.showerror("Error", f"Error: {str(e)}")

```

```

1532     def delete_student():
1533         if not messagebox.askyesno("Konfirmasi", f"Apakah Anda yakin ingin menghapus mahasiswa (username)?"):
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, "bold"), bg="#607080", 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)}")

```

```

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()
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>