LAPORAN AKHIR

MATA KULIAH STRUKTUR DATA DAN ALGORITMA

Disusun oleh:

Achmad Ryvaldy (22031554027)

Rihadatul ‘Aisy Nur Jannah (22031554028)

Ahmad Rafi Syaifudin (22031554030)

SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2023

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Saat ini, banyak industri yang telah menggunakan mesin-mesin canggih. Mesin sebagai komponen utama dalam proses produksi sangat penting untuk dijaga perawatan atau pemeliharannya. Dalam kegiatan industri, seringkali masalah akan timbul jika mesin, peralatan, dan aset lainnya tidak mendapatkan perawatan yang tepat (Wardhana, M.A.W., Pratama, K.D.P., dan Muryani, S., 2022). Oleh karena itu, pemeliharaan secara berkala dilakukan untuk mencegah terjadinya masalah tersebut. Apabila tidak terjaga, akan terjadi penurunan kinerja mesin yang akan menghambat proses produksi.

Di era yang sudah berkembang seperti saat ini, dibutuhkan aplikasi penjadwalan pemeliharaan mesin yang efisien dan fleksibel. Oleh karena itu, dibuatlah aplikasi "MachineCare" guna untuk menjadwalkan perawatan atau pemeliharaan mesin yang efisien dan fleksibel. Aplikasi ini dibuat dengan menggunakan bahasa pemrograman Python dan modul GUI yang terdapat pada Python. Selain itu, aplikasi ini menggunakan algoritma binary search dan bubble sort.

1.2 Rumusan Masalah

1. Bagaimana cara pemeliharaan mesin agar tidak terjadi kerusakan akibat kurang perawatan?
2. Bagaimana cara mencegah kerusakan mesin yang tidak terduga?

1.3 Tujuan

1. Untuk mengetahui cara pemeliharaan mesin dengan membuat aplikasi "MachineCare" yang menyediakan fitur penjadwalan perawatan mesin secara efisien dan fleksibel.
2. Untuk mengetahui cara mencegah kerusakan mesin yang tidak terduga dengan menggunakan aplikasi "MesinCare" melalui fitur pelacakan jadwal perawatan yang telah dilakukan serta jadwal perawatan yang akan datang.

1.4 Manfaaat

Aplikasi ini dibuat untuk membantu pengguna dalam melakukan pemantauan dan pelacakan terhadap jadwal perawatan yang telah dilakukan dan jadwal perawatan yang akan datang. Dengan demikian, pengguna dapat menghindari kerusakan mesin yang tidak terduga dan meningkatkan produktivitas dan efisiensi operasional mereka.

BAB II

LANDASAN TEORI

2.1 Perawatan atau Pemeliharaan

Perawatan atau pemeliharaan merupakan kegiatan yang diperlukan untuk menjaga atau mempertahankan kualitas dan kinerja mesin agar tetap berfungsi dengan baik seperti saat pertama kali dioperasikan (Ansori dan Mustajib, 2013 dalam Prihastono, E. dan Prakoso, B., 2017). Proses perawatan bertujuan untuk pencegahan guna mengurangi kerusakan peralatan dengan memastikan tingkat keandalan dan kesiapan yang optimal serta meminimalkan biaya perawatan (Prihastono, E. dan Prakoso, B., 2017). Oleh karena itu, perawatan atau pemeliharaan penting untuk dilakukan, jika terjadi kendala pada mesin maka proses produksi akan terhambat.

2.2 Python

Python merupakan bahasa pemrograman yang menyediakan struktur data tingkat tinggi dengan perancangann yang fokus pada keterbacaan kode agar sintaks lebih mudah dipahami (Aqmila, D., 2023). Python dapat digunakan secara bebas. Baris kode yang digunakan pada python lebih sedikit sehingga para pemula lebih mudah untuk memahaminya. Selain itu, struktur sintaks pada python mendekati bahasa manusia yang menjadikannya mudah untuk dipelajari.

2.3 GUI (Graphical User Interface)

GUI adalah sebuah antarmuka visual yang memungkinkan pengguna untuk berinteraksi dengan komputer dengan cara memberikan perintah melalui gambar tanpa perlu mengetik perintah tersebut (Mustakim, M., Fitrianingsih, N. and Fitriati, I., 2019). GUI terdiri dari widgets (button, menu, checkbox, dll). Modul GUI pada Python adalah tkinter.

2.4 Binary Search

Binary search merupakan pencarian data pada array yang sudah terurut. Jika array belum terurut, maka pencarian binary tidak dapat dilakukan. Binary search berguna untuk memperkecil jumlah operasi perbandingan, beban komputasi lebih kecil, dan melakukan proses pembagian ruang pencarian secara berulang-ulang (Darmawantoro, R.Y., Utami, Y.R.W., dan Kustanto, K., 2022).

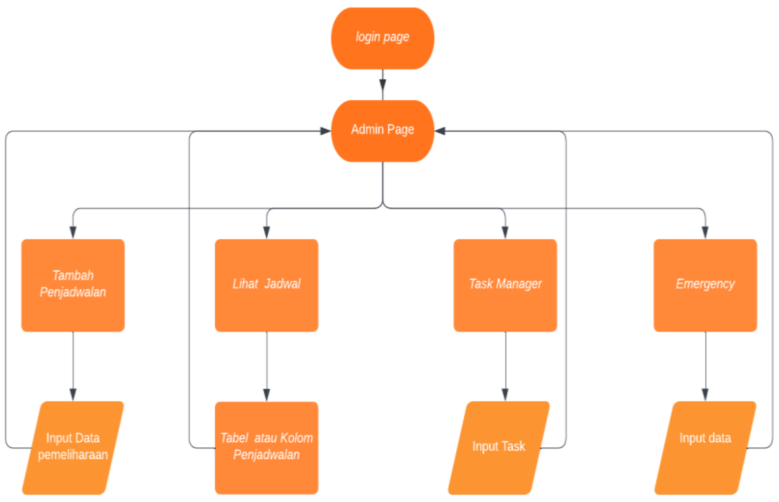
2.5 Bubble Sort

Bubble sort merupakan algoritma sorting yang mengambil nilai paling besar dan diletakkan di paling kanan dengan membandingkan elemen sekarang dan elemen berikutnya (Haryanda, dkk., 2023). Bubble sort termasuk algoritma sorting yang paling sederhana. Dalam proses pengurutan secara menaik (ascending), jika suatu elemen sekarang memiliki nilai yang lebih besar daripada elemen berikutnya, maka elemen tersebut akan ditukar posisinya. Sedangkan dalam pengurutan secara menurun (descending), jika suatu elemen sekarang memiliki nilai yang lebih kecil daripada elemen berikutnya, maka kedua elemen tersebut akan ditukar posisinya (Haryanda, dkk., 2023).

BAB III

DESAIN PROYEK

3.1 Diagram Alir



3.2 Rancangan System

- Login

  Halaman login digunakan untuk masuk ke aplikasi. Di halaman login admin akan diminta untuk memasukan username dan password yang telah diberikan.

- Tambah Penjadwalan

  Halaman penjadwalan digunakan untuk input beberapa data yang telah diisi oleh admin. Data akan tersimpan di file csv.

- Lihat Jadwal

  Halaman lihat jadwal digunakan untuk melihat jadwal berdasarkan waktu terdekat atau waktu terjauh, mencari jadwal, dan menghapus janji temu di tabel.

- Task Manager

  Halaman task manager digunakan untuk membuat daftar tugas, mengelola tugas untuk mekanik, dan mencari tugas.

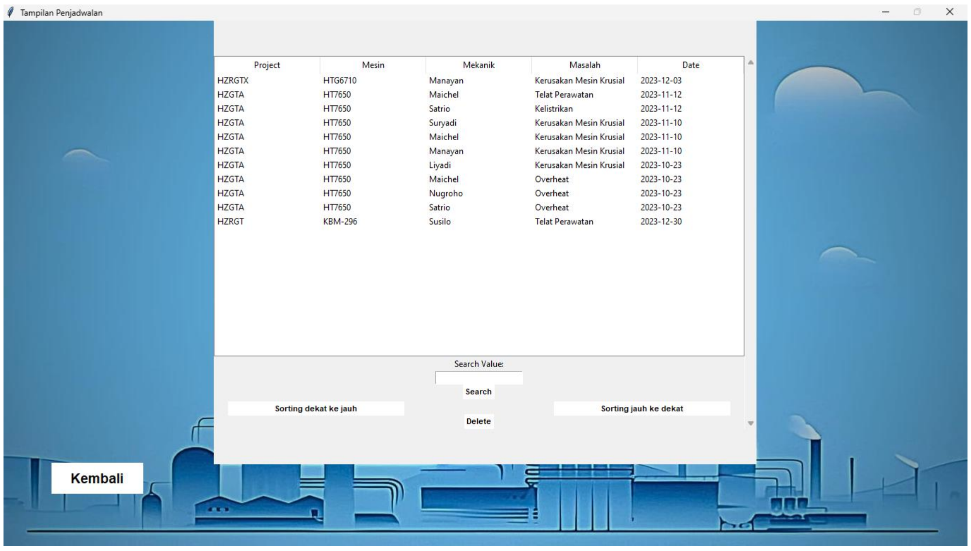
- Emergency

  Halaman emergency digunakan untuk memanggil mekanik dan otomatis memasukkan data janji temu ke data penjadwalan secara real time.

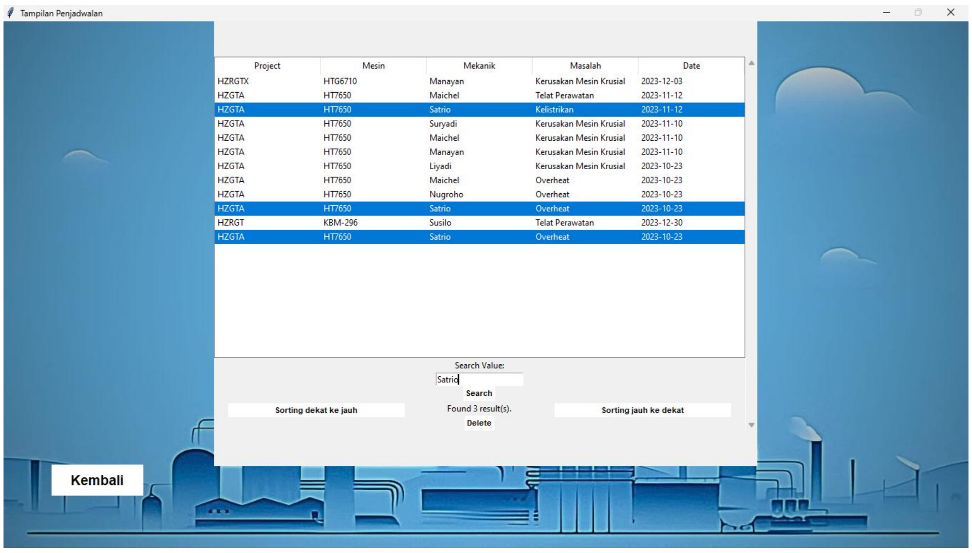
BAB IV

IMPLEMENTASI

4.1 Binary Search



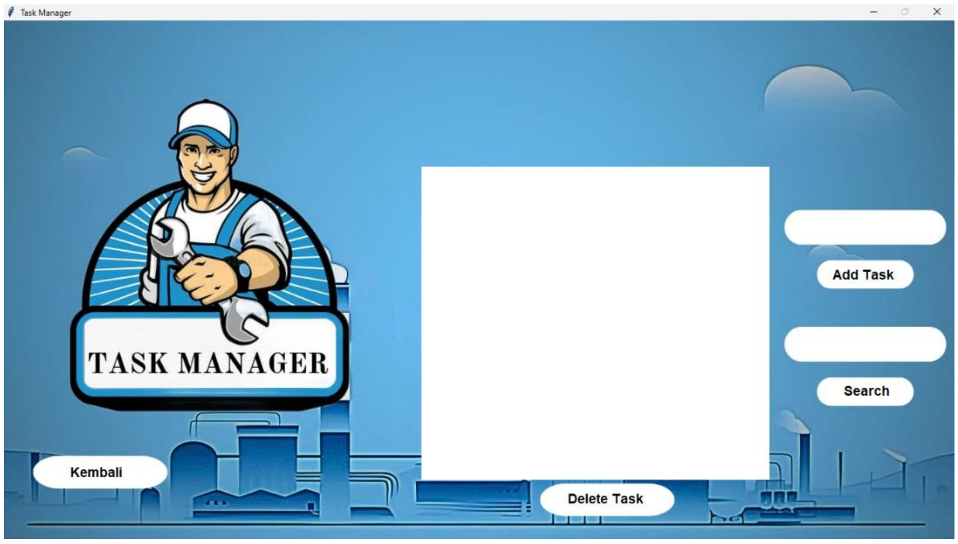
Project	Mesin	Mekanik	Masalah	Date
HZRGTX	HTG6710	Manayan	Kerusakan Mesin Krusial	2023-12-03
HZGTA	HT7650	Maichel	Telat Perawatan	2023-11-12
HZGTA	HT7650	Satrio	Kelistrikan	2023-11-12
HZGTA	HT7650	Suryadi	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Maichel	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Manayan	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Liyadi	Kerusakan Mesin Krusial	2023-10-23
HZGTA	HT7650	Maichel	Overheat	2023-10-23
HZGTA	HT7650	Nugroho	Overheat	2023-10-23
HZGTA	HT7650	Satrio	Overheat	2023-10-23
HZRGT	KBM-296	Susilo	Telat Perawatan	2023-12-30

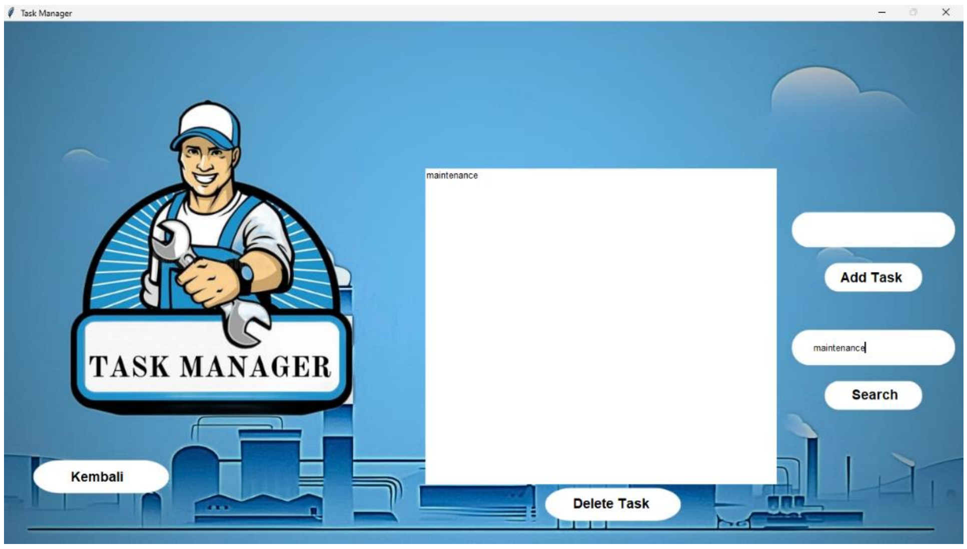


Project	Mesin	Mekanik	Masalah	Date
HZRGTX	HTG6710	Manayan	Kerusakan Mesin Krusial	2023-12-03
HZGTA	HT7650	Maichel	Telat Perawatan	2023-11-12
HZGTA	HT7650	Satrio	Kelistrikan	2023-11-12
HZGTA	HT7650	Suryadi	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Maichel	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Manayan	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Liyadi	Kerusakan Mesin Krusial	2023-10-23
HZGTA	HT7650	Maichel	Overheat	2023-10-23
HZGTA	HT7650	Nugroho	Overheat	2023-10-23
HZGTA	HT7650	Satrio	Overheat	2023-10-23
HZRGT	KBM-296	Susilo	Telat Perawatan	2023-12-30
HZGTA	HT7650	Satrio	Overheat	2023-10-23

Found 3 result(s).

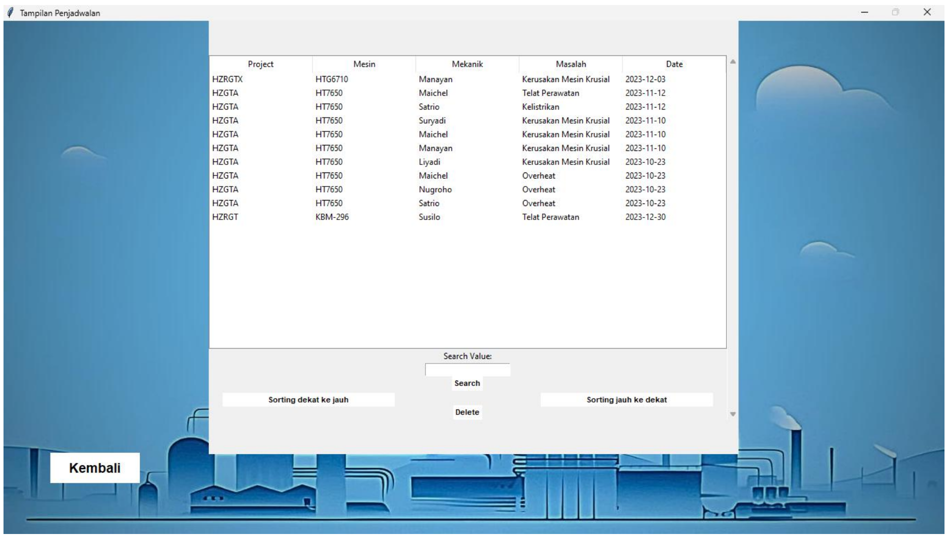
Penggunaan algoritma binary search pada fitur lihat jadwal.



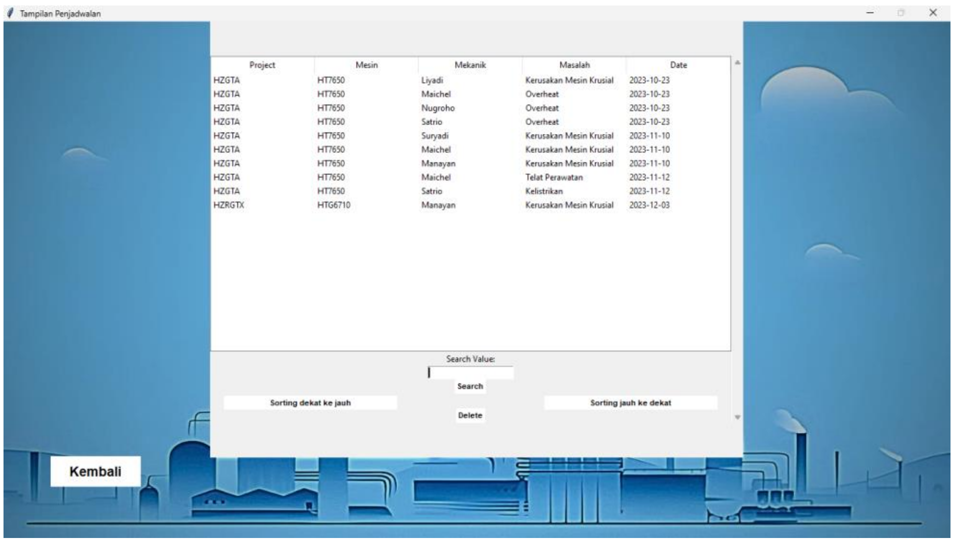


Penggunaan algoritma binary search pada fitur task manager.

4.2 Bubble Sort



Project	Mesin	Mekanik	Masalah	Date
HZRGTX	HTG6710	Manayan	Kerusakan Mesin Krusial	2023-12-03
HZGTA	HT7650	Maichel	Telat Perawatan	2023-11-12
HZGTA	HT7650	Satrio	Kelistrikan	2023-11-12
HZGTA	HT7650	Suryadi	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Maichel	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Manayan	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Liyadi	Kerusakan Mesin Krusial	2023-10-23
HZGTA	HT7650	Maichel	Overheat	2023-10-23
HZGTA	HT7650	Nugroho	Overheat	2023-10-23
HZGTA	HT7650	Satrio	Overheat	2023-10-23
HZRGT	KBM-296	Susilo	Telat Perawatan	2023-12-30



Project	Mesin	Mekanik	Masalah	Date
HZGTA	HT7650	Liyadi	Kerusakan Mesin Krusial	2023-10-23
HZGTA	HT7650	Maichel	Overheat	2023-10-23
HZGTA	HT7650	Nugroho	Overheat	2023-10-23
HZGTA	HT7650	Satrio	Overheat	2023-10-23
HZGTA	HT7650	Suryadi	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Maichel	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Manayan	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Maichel	Telat Perawatan	2023-11-12
HZGTA	HT7650	Satrio	Kelistrikan	2023-11-12
HZRGTX	HTG6710	Manayan	Kerusakan Mesin Krusial	2023-12-03

Penggunaan algoritma bubble sort pada fitur lihat jadwal untuk sorting berdasarkan waktu terdekat dan waktu terjauh.

KESIMPULAN

Aplikasi "MachineCare" merupakan solusi efisien dan fleksibel untuk menjadwalkan perawatan atau pemeliharaan mesin. Aplikasi ini dibuat dengan menggunakan bahasa pemrograman Python dan modul GUI tkinter. Selain itu, aplikasi ini juga menggunakan algoritma binary search dan bubble sort yang memungkinkan pengguna untuk melacak jadwal perawatan yang telah dilakukan dan jadwal perawatan yang akan datang. Dengan pemeliharaan yang baik, diharapkan penurunan kinerja mesin dapat dihindari. Dalam pengembangannya, aplikasi "MachineCare" dapat terus dikembangkan dengan menambah fitur-fitur lain yang lebih relevan guna untuk mendukung kegiatan industry dalam menjaga kinerja mesin.

DAFTAR PUSTAKA

Wardhana, M.A.W., Pratama, K.D.P. and Muryani, S., 2022. Aplikasi Informasi Pemeliharaan Alat Produksi Pada PT. Teguh Karya Perima. Jurnal Infortech, 4(2), pp.148-155.

Prihastono, E. and Prakoso, B., 2017. Perawatan preventif untuk mempertahankan utilitas performance pada mesin cooling tower di cv. arhu tapselindo bandung. Dinamika Teknik Industri.

Aqmila, D., 2023. Perancangan Media Pembelajaran Bahasa Pemrograman Python Menggunakan Aplikasi Scratch Untuk Siswa Sekolah Menengah Pertama (SMP) (Doctoral dissertation, UIN Ar-Raniry Fakultas Tarbiyah dan Keguruan).

Mustakim, M., Fitrianingsih, N. and Fitriati, I., 2019. Pengembangan Aplikasi E-Raport Berbasis Graphical User Interface (GUI) dengan Menggunakan VB. Net 2010 di SMKN 10 Bima. Jurnal Pendidikan Mipa, 9(1), pp.67-75.

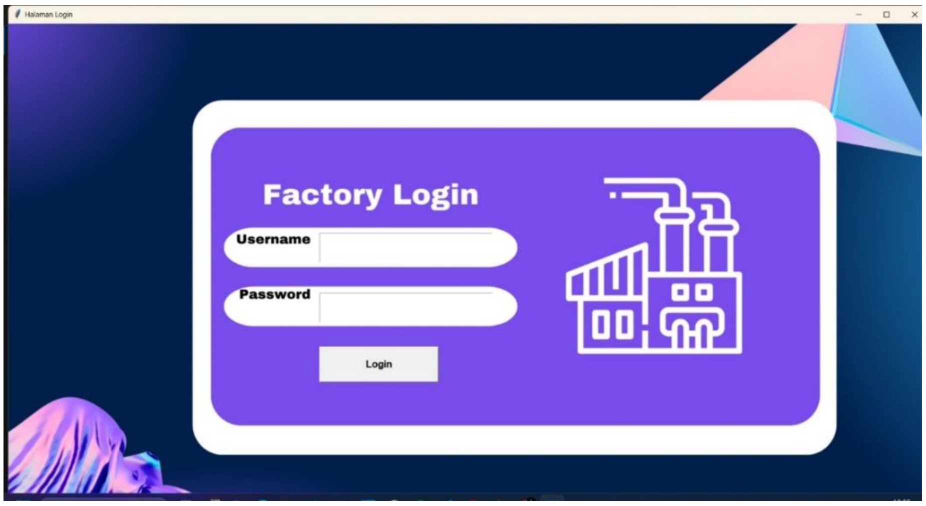
Darmawantoro, R.Y., Utami, Y.R.W. and Kustanto, K., 2022. Implementasi Binary Search Untuk Data Obat di Apotek. Jurnal Teknologi Informasi dan Komunikasi (TIKomSiN), 10(1), pp.76-84.

Haryanda, H., Nasution, M.F., Hutabarat, D., Razzaq, A. and Syahputra, A., 2023. Implementasi Metode Bubble Sort pada Aplikasi Pencarian Rute Berdasarkan Jarak Tempuh Transportasi Umum. Blend Sains Jurnal Teknik, 1(3), pp.213-219.

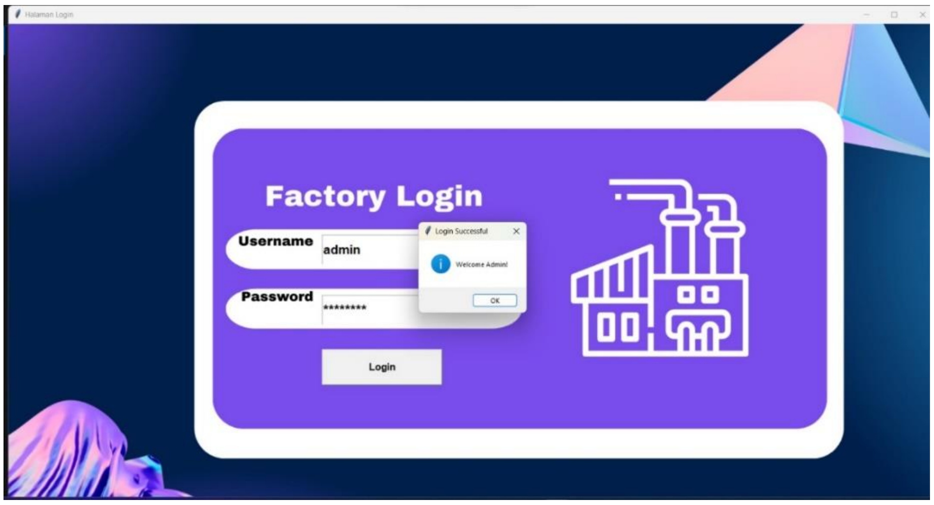
LAMPIRAN

Lampiran 1. *User Manual*

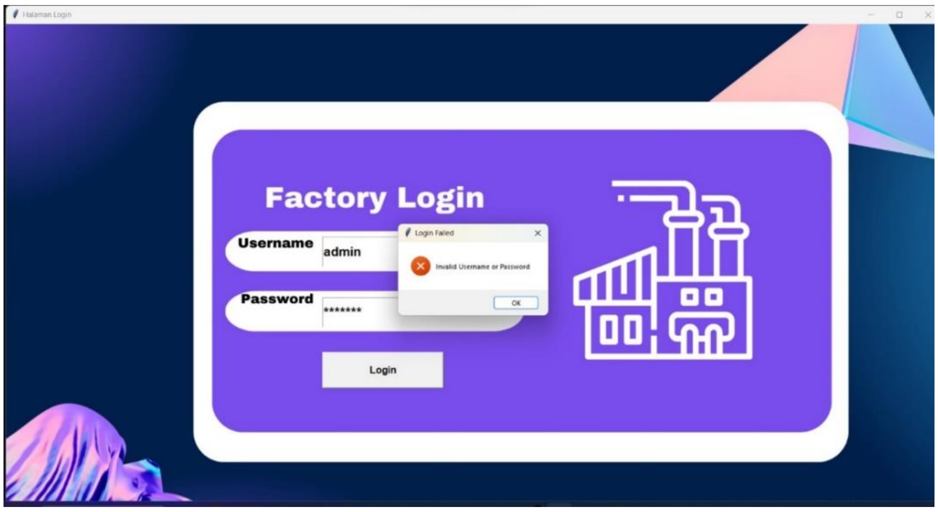
1. Halaman Login



Halaman login digunakan untuk admin masuk ke dalam aplikasi dengan memasukkan username dan password yang telah diberikan.

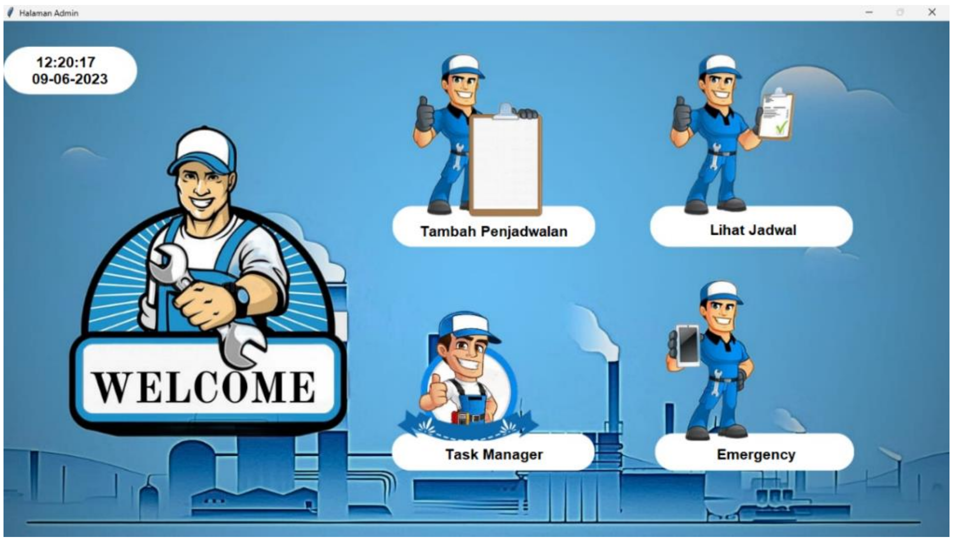


Setelah admin memasukkan username dan password dengan benar. Maka, login dinyatakan berhasil dengan munculnya notifikasi "Welcome Admin!".



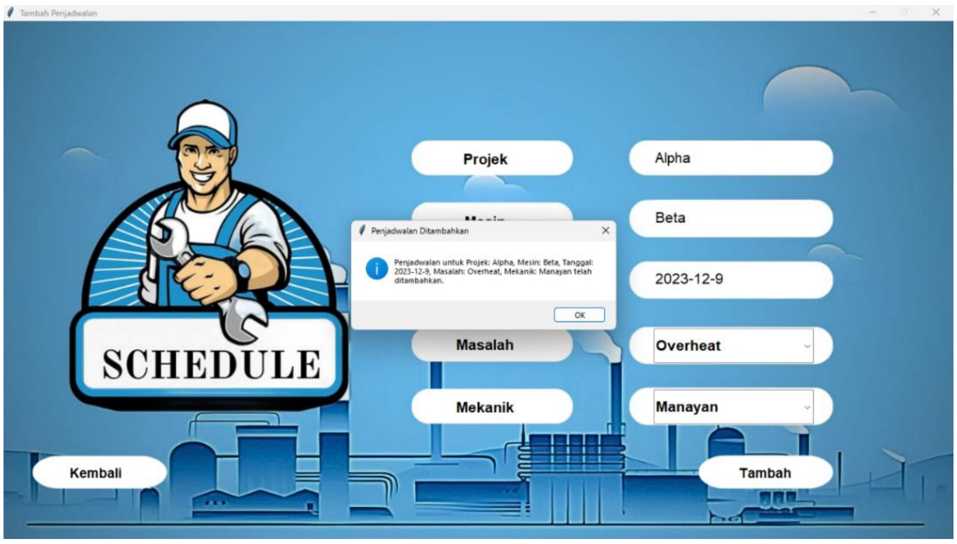
Jika username dan password yang dimasukkan salah, maka akan muncul notifikasi “Invalid Username or Password”.

2. Halaman Admin



Jika login berhasil, halaman akan beralih ke halaman admin. Di halaman admin terdapat 4 fitur pilihan, yaitu tambah penjadwalan, lihat jadwal, task manager, dan emergency.

3. Halaman Tambah Penjadwalan


Tambah Penjadwalan

Projek: Alpha

Mesin: Beta

2023-12-9

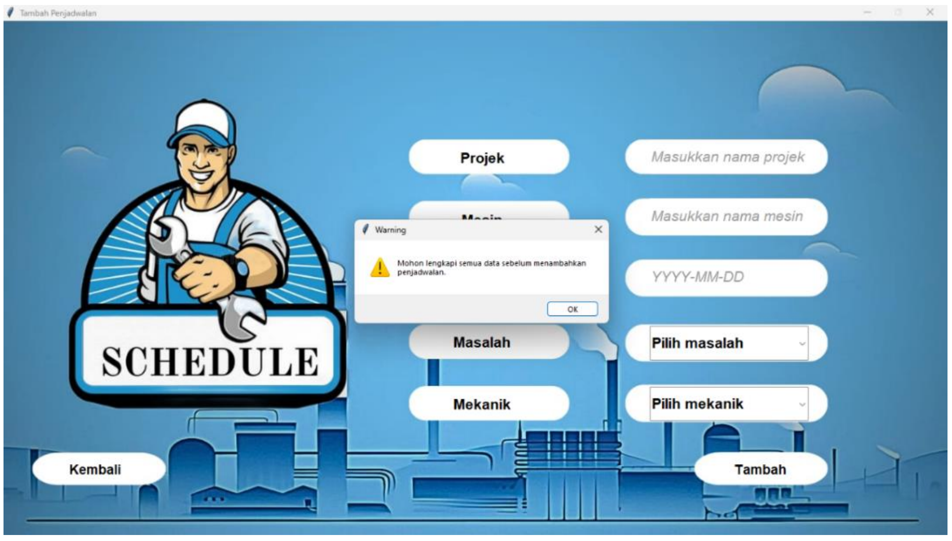
Masalah: Overheat

Mekanik: Manayan

Penjadwalan Ditambahkan

Penjadwalan untuk Projek: Alpha, Mesin: Beta, Tanggal: 2023-12-9, Masalah: Overheat, Mekanik: Manayan telah ditambahkan.

Di halaman tambah penjadwalan terdapat beberapa data yang harus diisi oleh pengguna. Data-data tersebut akan tersimpan di file csv yang telah tersedia.


Tambah Penjadwalan

Projek: Masukkan nama projek

Mesin: Masukkan nama mesin

YYYY-MM-DD

Masalah: Pilih masalah

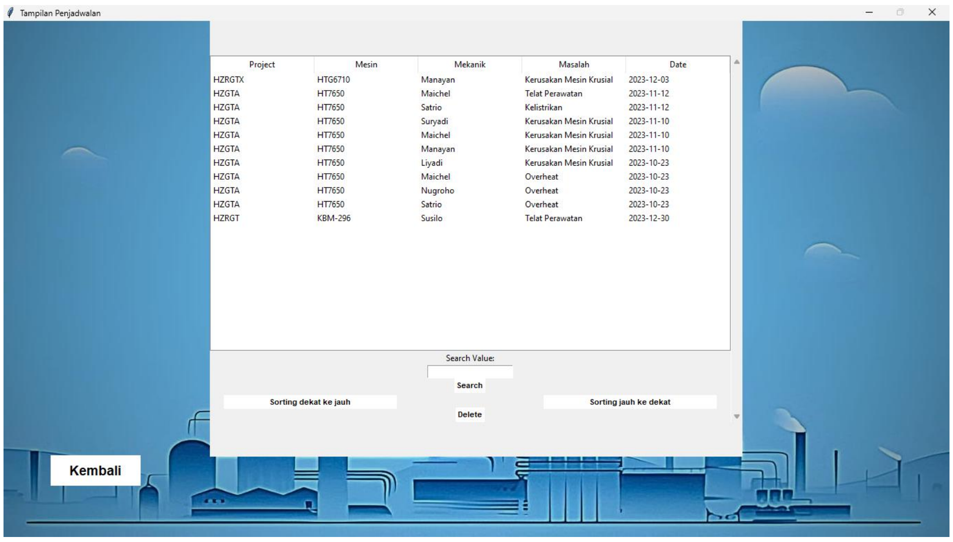
Mekanik: Pilih mekanik

Warning

Mohon lengkapi semua data sebelum menambahkan penjadwalan.

Jika admin tidak melengkapi data-data yang telah diberikan, maka akan muncul peringatan.

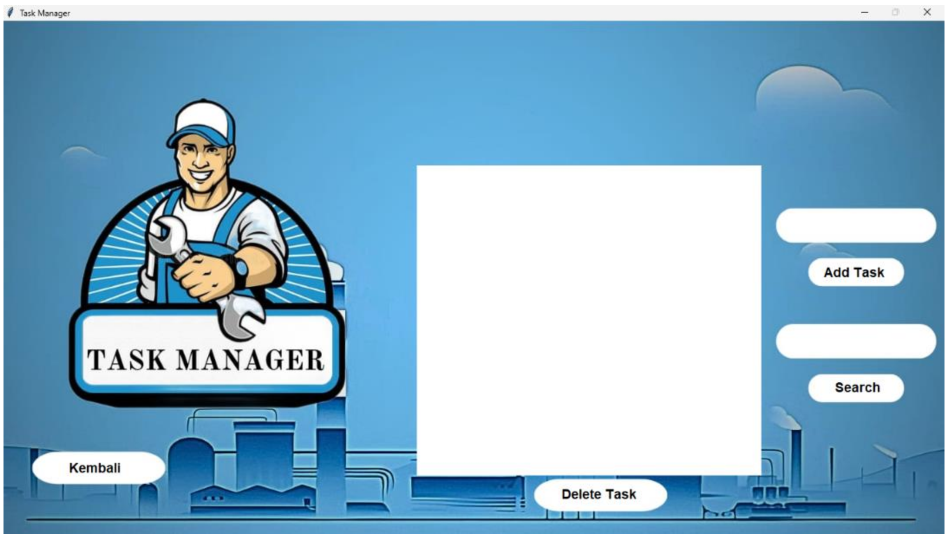
4. Halaman Lihat Jadwal

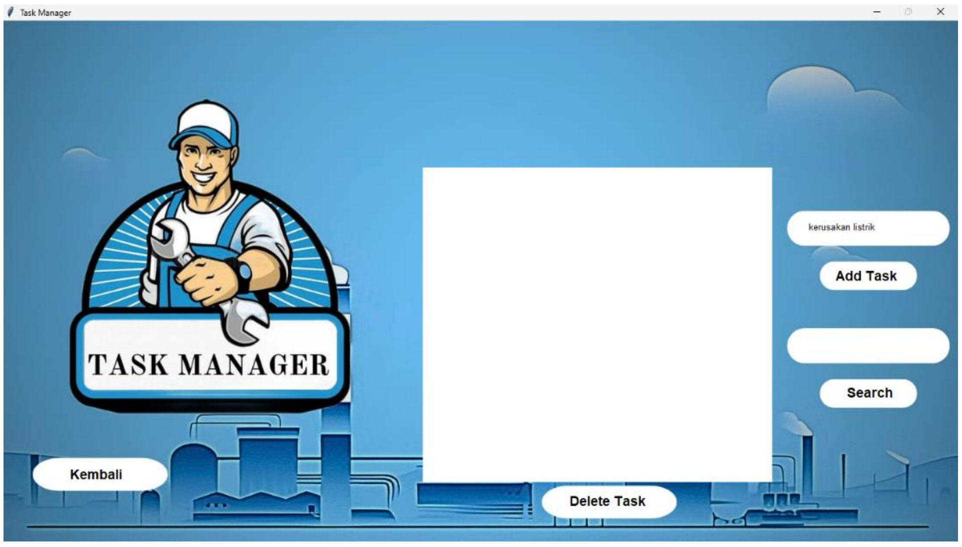


Project	Mesin	Mekanik	Masalah	Date
HZRGTX	HTG6710	Manayan	Kerusakan Mesin Krusial	2023-12-03
HZGTA	HT7650	Maichel	Telat Perawatan	2023-11-12
HZGTA	HT7650	Satrio	Kelistrikan	2023-11-12
HZGTA	HT7650	Suryadi	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Maichel	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Manayan	Kerusakan Mesin Krusial	2023-11-10
HZGTA	HT7650	Liyadi	Kerusakan Mesin Krusial	2023-10-23
HZGTA	HT7650	Maichel	Overheat	2023-10-23
HZGTA	HT7650	Nugroho	Overheat	2023-10-23
HZGTA	HT7650	Satrio	Overheat	2023-10-23
HZRGT	KBM-296	Susilo	Telat Perawatan	2023-12-30

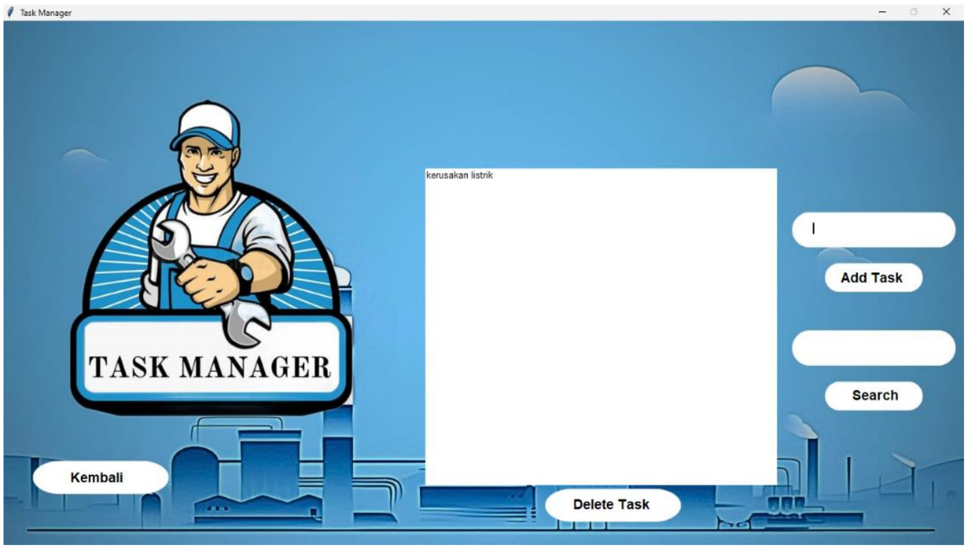
Pada halaman lihat jadwal, pengguna dapat melihat jadwal berdasarkan waktu terdekat atau waktu terjauh dan bisa mencari jadwal. Selain itu, pengguna juga dapat menghapus jadwal yang ada di tabel.

5. Halaman Task Manager



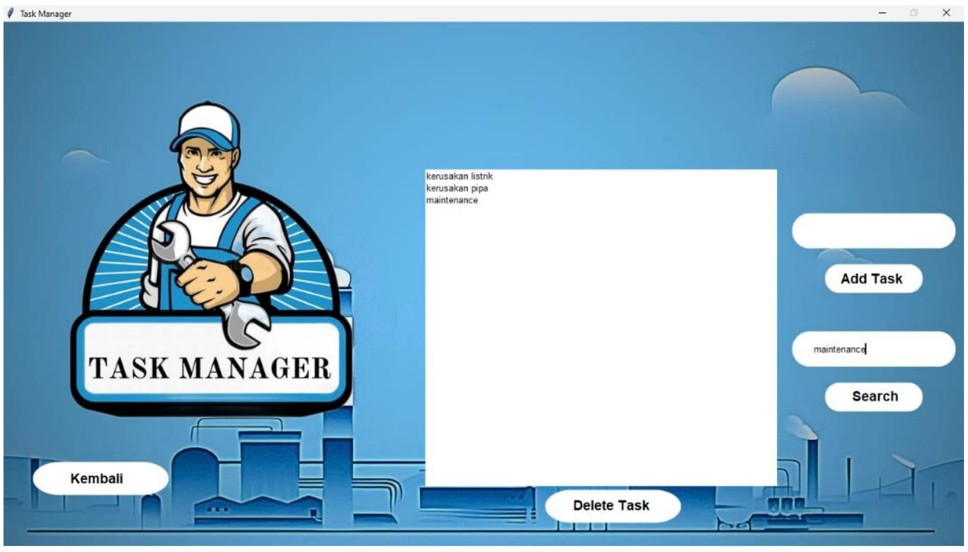


kerusakan listrik



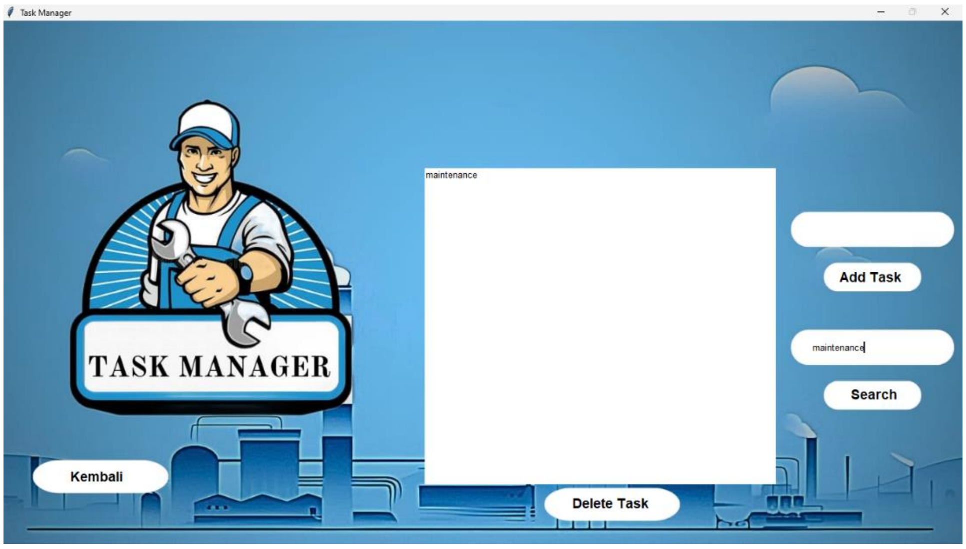
kerusakan listrik

Pada halaman task manager, pengguna bisa membuat daftar tugas.

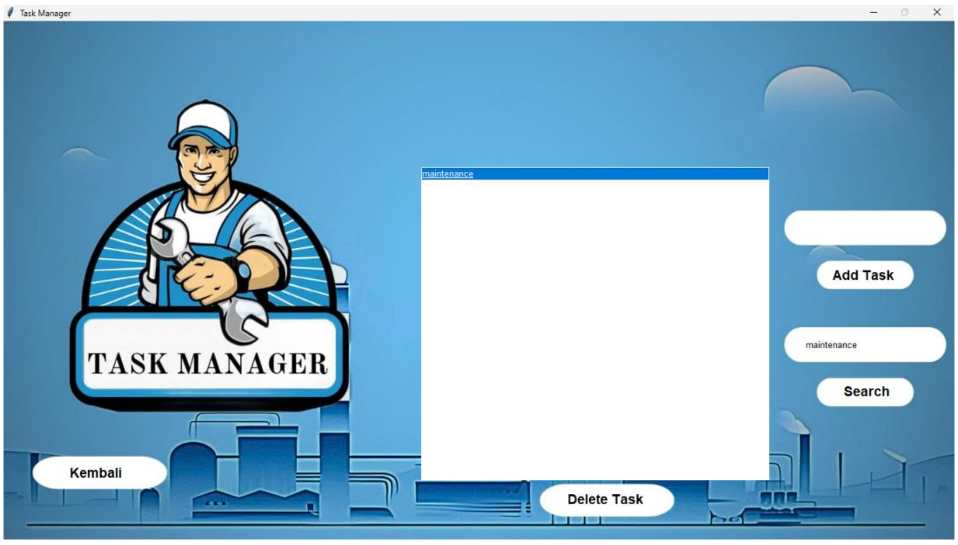


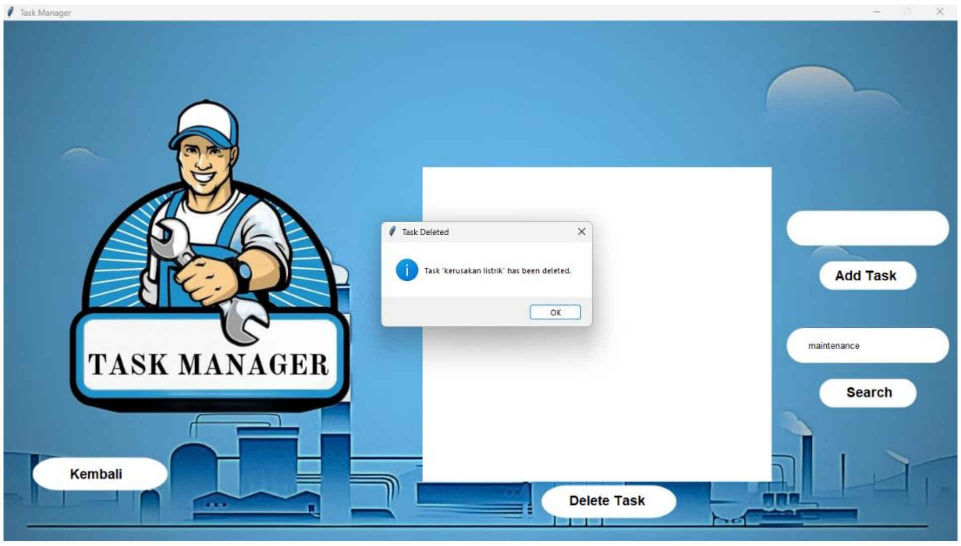
kerusakan listrik
kerusakan pipa
maintenance

maintenance



Selain menambahkan tugas, pengguna juga bisa mencari tugas yang telah ditambahkan sebelumnya.

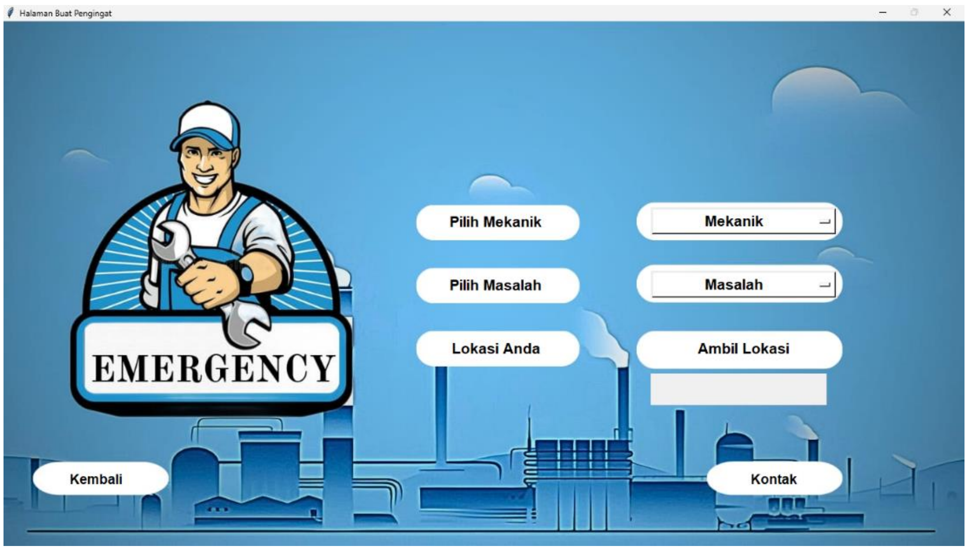


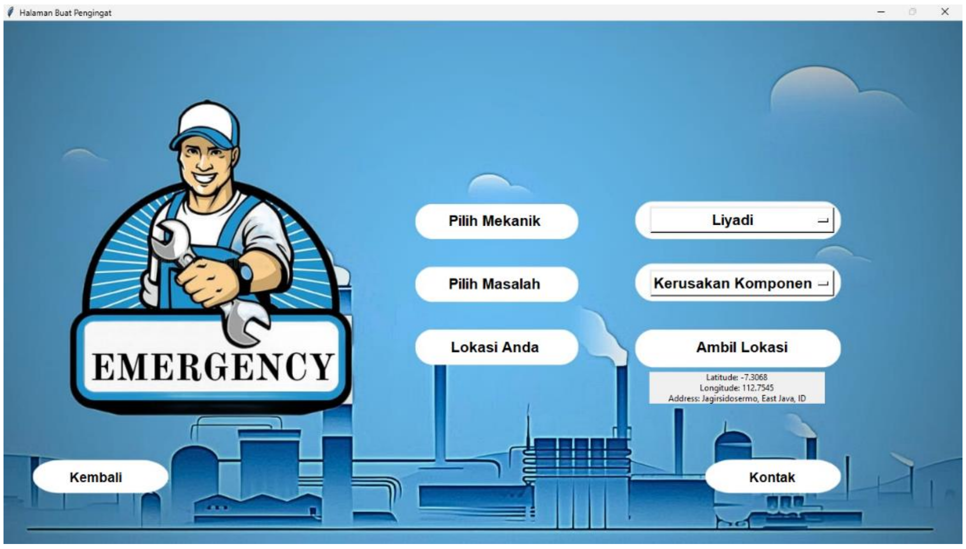


Task 'kerusakan listrik' has been deleted.

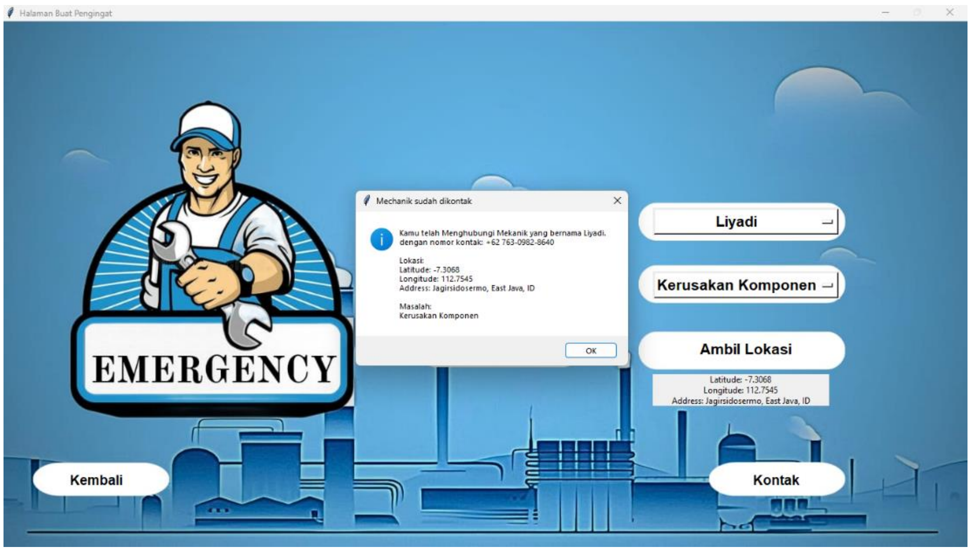
Selain menambahkan tugas dan mencari tugas, fitur task manager juga dapat menghapus tugas yang telah ditambahkan.

6. Halaman Emergency





Latitude: -7.3068
Longitude: 112.7545
Address: Jagirsidosermo, East Java, ID



Mechanik sudah dikontak

Kamu telah Menghubungi Mekanik yang bernama Liyadi.
dengan nomor kontak: +62 763-0982-8640

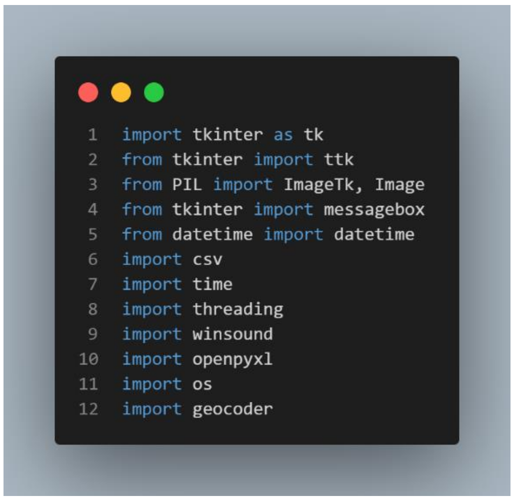
Lokasi:
Latitude: -7.3068
Longitude: 112.7545
Address: Jagirsidosermo, East Java, ID

Masalah:
Kerusakan Komponen

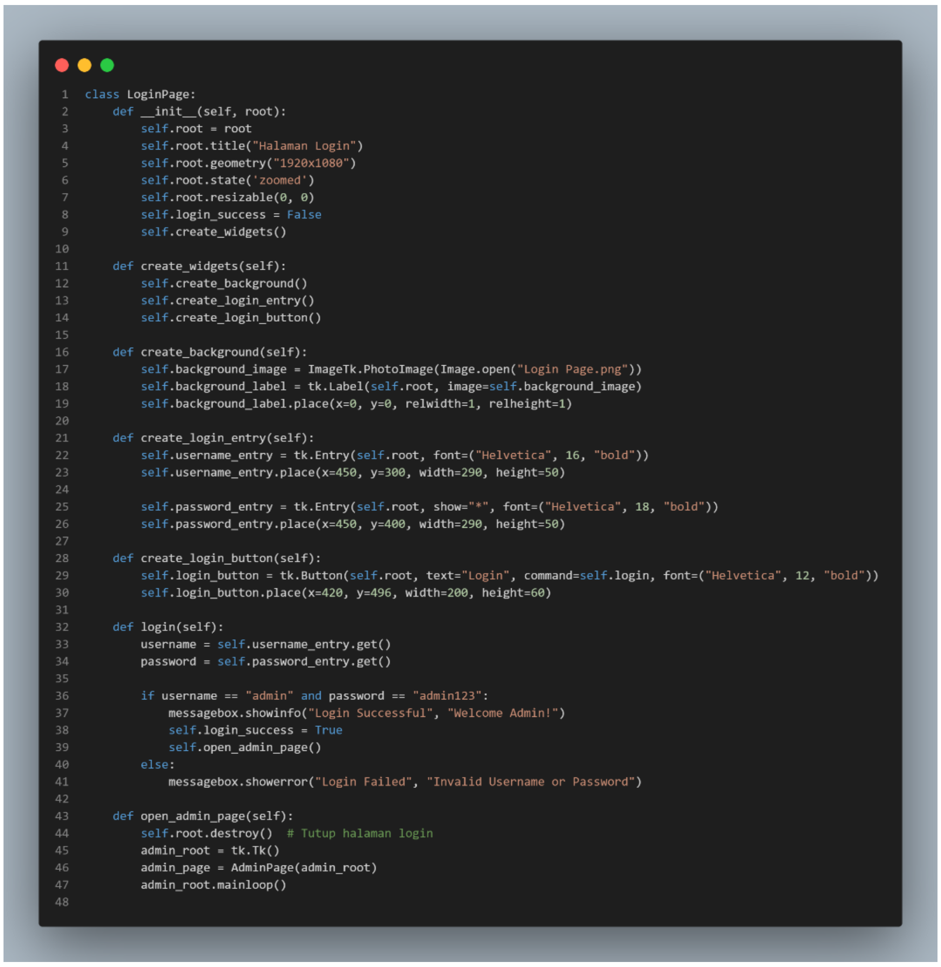
Latitude: -7.3068
Longitude: 112.7545
Address: Jagirsidosermo, East Java, ID

Pada halaman emergency, pengguna diminta memasukkan data-data untuk membuat janji dengan mekanik. Pengguna juga dapat menghubungi mekanik tersebut.

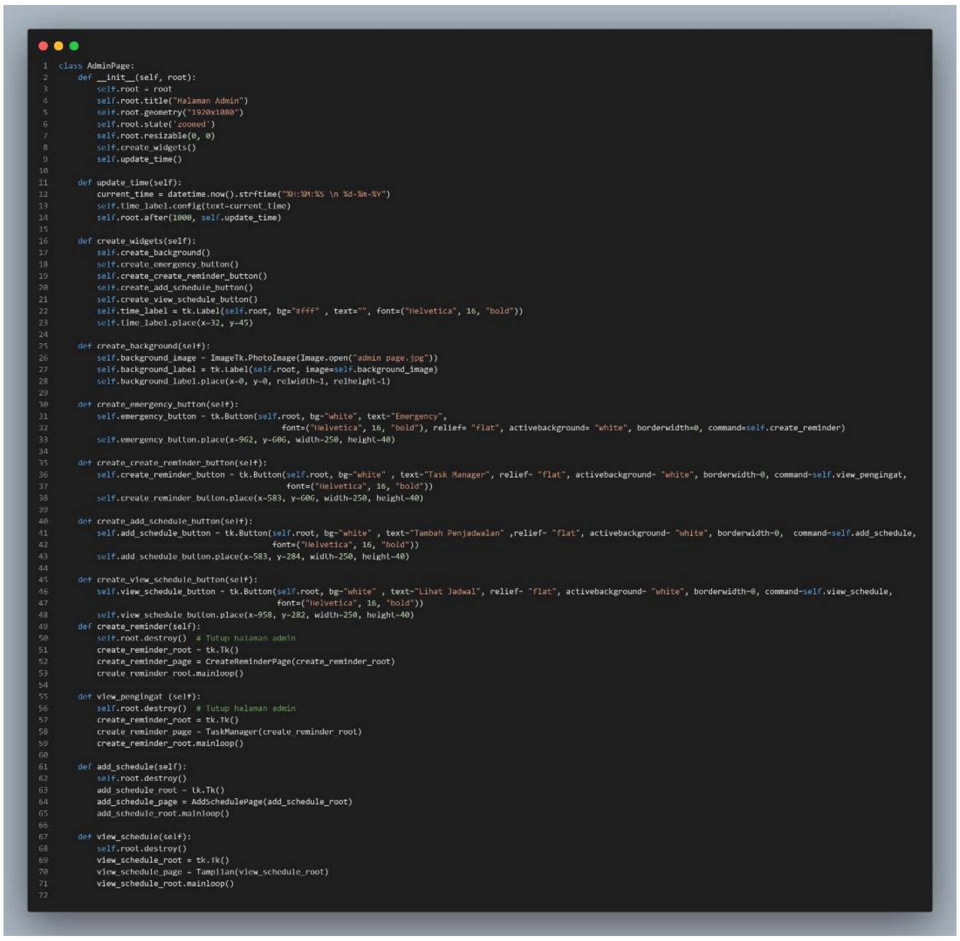
Lampiran 2. *Listing Code*



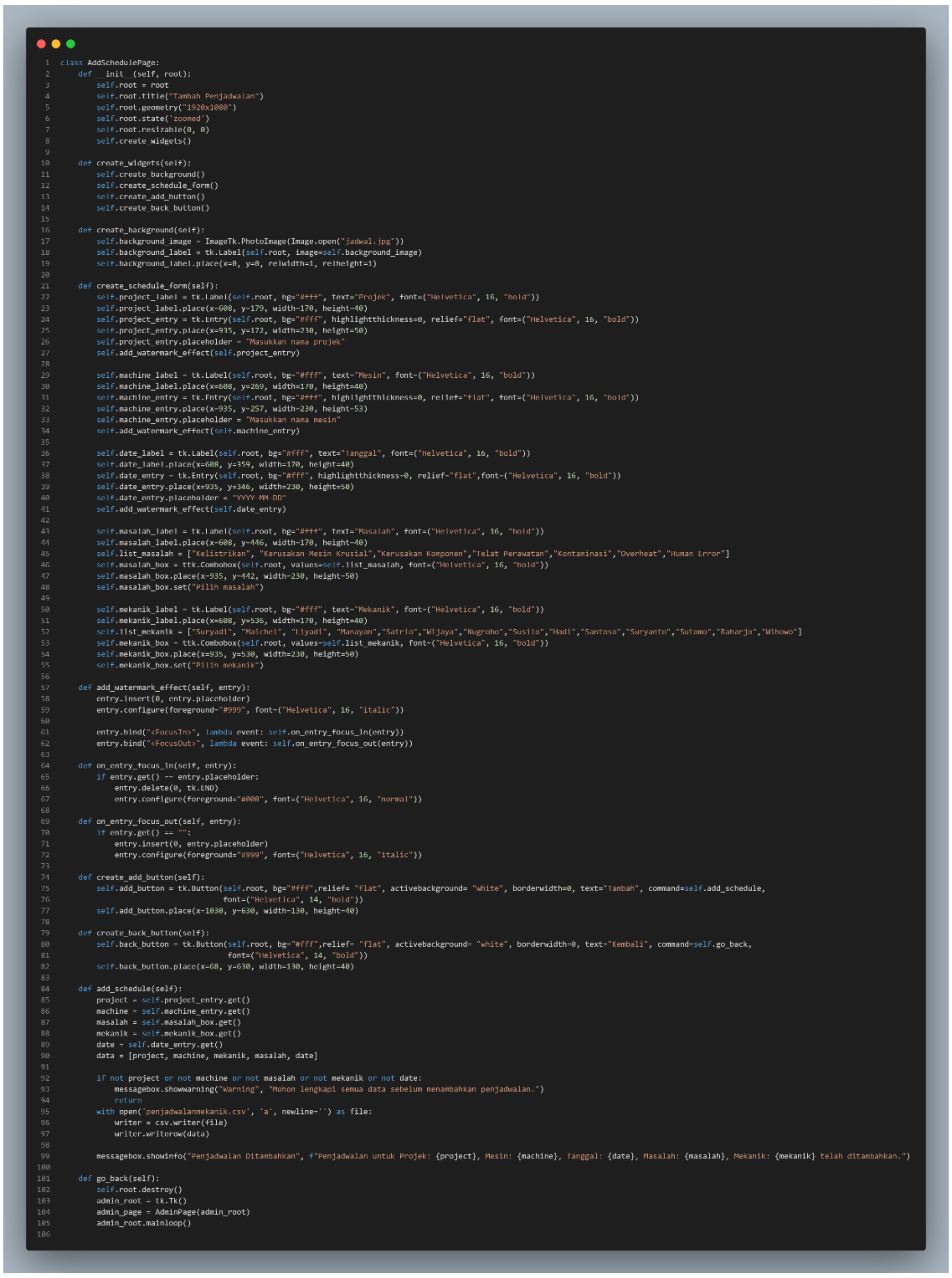
```python
import tkinter as tk
from tkinter import ttk
from PIL import ImageTk, Image
from tkinter import messagebox
from datetime import datetime
import csv
import time
import threading
import winsound
import openpyxl
import os
import geocoder
```



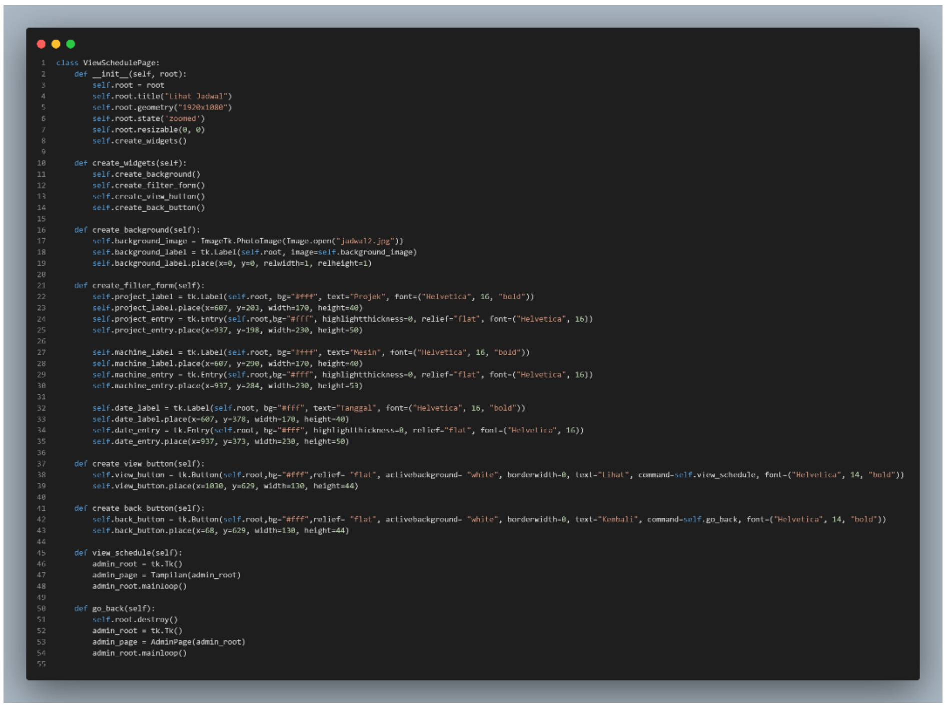
```python
class LoginPage:
    def __init__(self, root):
        self.root = root
        self.root.title("Halaman Login")
        self.root.geometry("1920x1080")
        self.root.state('zoomed')
        self.root.resizable(0, 0)
        self.login_success = False
        self.create_widgets()

    def create_widgets(self):
        self.create_background()
        self.create_login_entry()
        self.create_login_button()

    def create_background(self):
        self.background_image = ImageTk.PhotoImage(Image.open("Login Page.png"))
        self.background_label = tk.Label(self.root, image=self.background_image)
        self.background_label.place(x=0, y=0, relwidth=1, relheight=1)

    def create_login_entry(self):
        self.username_entry = tk.Entry(self.root, font=("Helvetica", 16, "bold"))
        self.username_entry.place(x=450, y=300, width=290, height=50)

        self.password_entry = tk.Entry(self.root, show="*", font=("Helvetica", 18, "bold"))
        self.password_entry.place(x=450, y=400, width=290, height=50)

    def create_login_button(self):
        self.login_button = tk.Button(self.root, text="Login", command=self.login, font=("Helvetica", 12, "bold"))
        self.login_button.place(x=420, y=496, width=200, height=60)

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

        if username == "admin" and password == "admin123":
            messagebox.showinfo("Login Successful", "Welcome Admin!")
            self.login_success = True
            self.open_admin_page()
        else:
            messagebox.showerror("Login Failed", "Invalid Username or Password")

    def open_admin_page(self):
        self.root.destroy()  # Tutup halaman login
        admin_root = tk.Tk()
        admin_page = AdminPage(admin_root)
        admin_root.mainloop()

```



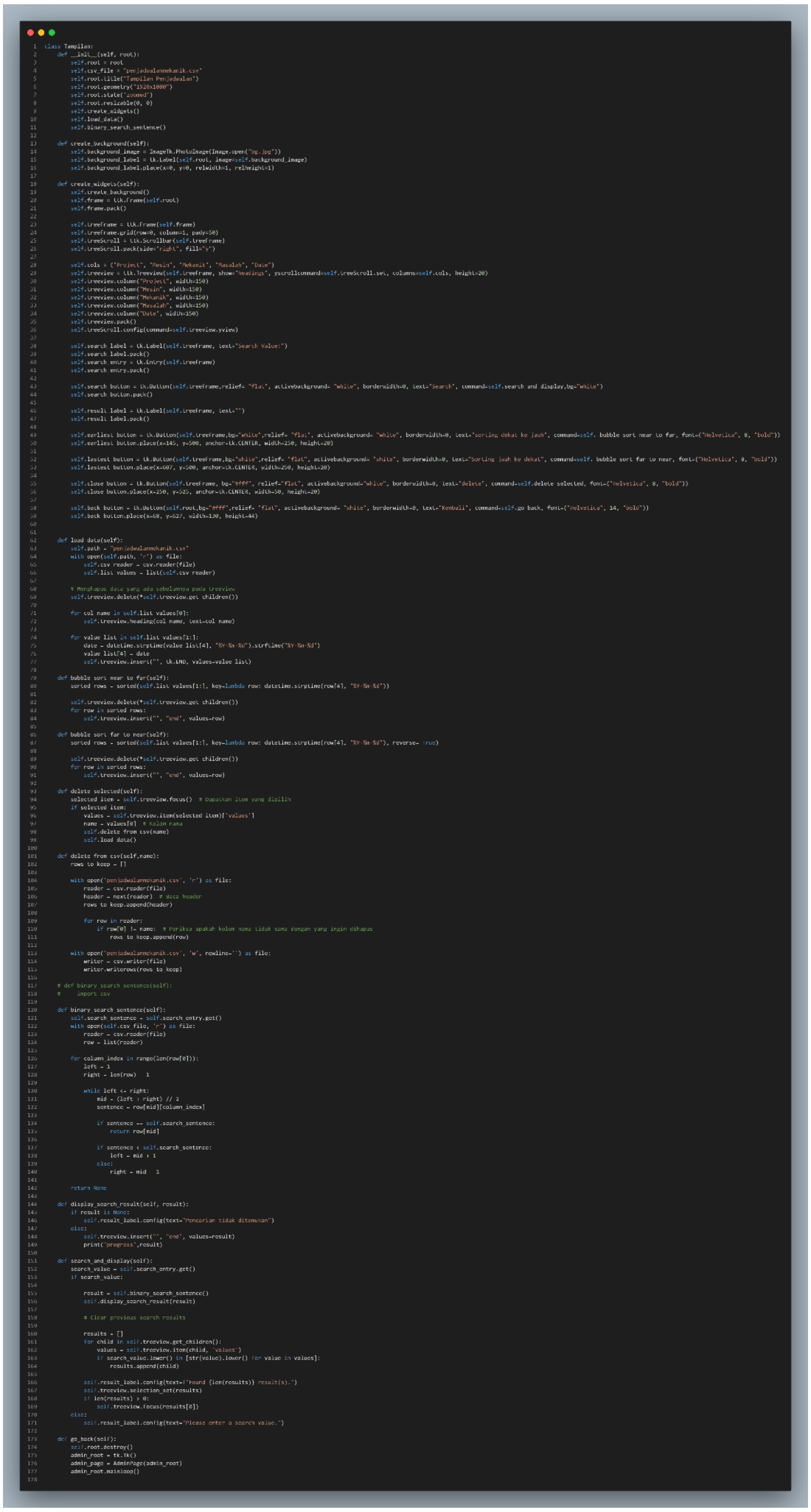
```python
class AdminPage:
    def __init__(self, root):
        self.root = root
        self.root.title("Halaman Admin")
        self.root.geometry("1920x1080")
        self.root.state('zoomed')
        self.root.resizable(0, 0)
        self.create_widgets()
        self.update_time()

    def update_time(self):
        current_time = datetime.now().strftime("%H:%M:%S \n %d-%m-%Y")
        self.time_label.config(text=current_time)
        self.root.after(1000, self.update_time)

    def create_widgets(self):
        self.create_background()
        self.create_emergency_button()
        self.create_create_reminder_button()
        self.create_add_schedule_button()
        self.create_view_schedule_button()
        self.time_label = tk.Label(self.root, bg="#fff" , text="", font=("Helvetica", 16, "bold"))
        self.time_label.place(x=32, y=45)

    def create_background(self):
        self.background_image = ImageTk.PhotoImage(Image.open("admin page.jpg"))
        self.background_label = tk.Label(self.root, image=self.background_image)
        self.background_label.place(x=0, y=0, relwidth=1, relheight=1)

    def create_emergency_button(self):
        self.emergency_button = tk.Button(self.root, bg="white", text="Emergency",
                                          font=("Helvetica", 16, "bold"), relief= "flat", activebackground= "white", borderwidth=0, command=self.create_reminder)
        self.emergency_button.place(x=962, y=606, width=250, height=40)

    def create_create_reminder_button(self):
        self.create_reminder_button = tk.Button(self.root, bg="white" , text="Task Manager", relief= "flat", activebackground= "white", borderwidth=0, command=self.view_pengingat,
                                          font=("Helvetica", 16, "bold"))
        self.create_reminder_button.place(x=583, y=606, width=250, height=40)

    def create_add_schedule_button(self):
        self.add_schedule_button = tk.Button(self.root, bg="white" , text="Tambah Penjadwalan" ,relief= "flat", activebackground= "white", borderwidth=0,  command=self.add_schedule,
                                          font=("Helvetica", 16, "bold"))
        self.add_schedule_button.place(x=583, y=284, width=250, height=40)

    def create_view_schedule_button(self):
        self.view_schedule_button = tk.Button(self.root, bg="white" , text="Lihat Jadwal", relief= "flat", activebackground= "white", borderwidth=0, command=self.view_schedule,
                                          font=("Helvetica", 16, "bold"))
        self.view_schedule_button.place(x=958, y=282, width=250, height=40)
    def create_reminder(self):
        self.root.destroy()  # Tutup halaman admin
        create_reminder_root = tk.Tk()
        create_reminder_page = CreateReminderPage(create_reminder_root)
        create_reminder_root.mainloop()

    def view_pengingat (self):
        self.root.destroy()  # Tutup halaman admin
        create_reminder_root = tk.Tk()
        create_reminder_page = TaskManager(create_reminder_root)
        create_reminder_root.mainloop()

    def add_schedule(self):
        self.root.destroy()
        add_schedule_root = tk.Tk()
        add_schedule_page = AddSchedulePage(add_schedule_root)
        add_schedule_root.mainloop()

    def view_schedule(self):
        self.root.destroy()
        view_schedule_root = tk.Tk()
        view_schedule_page = Tampilan(view_schedule_root)
        view_schedule_root.mainloop()
```



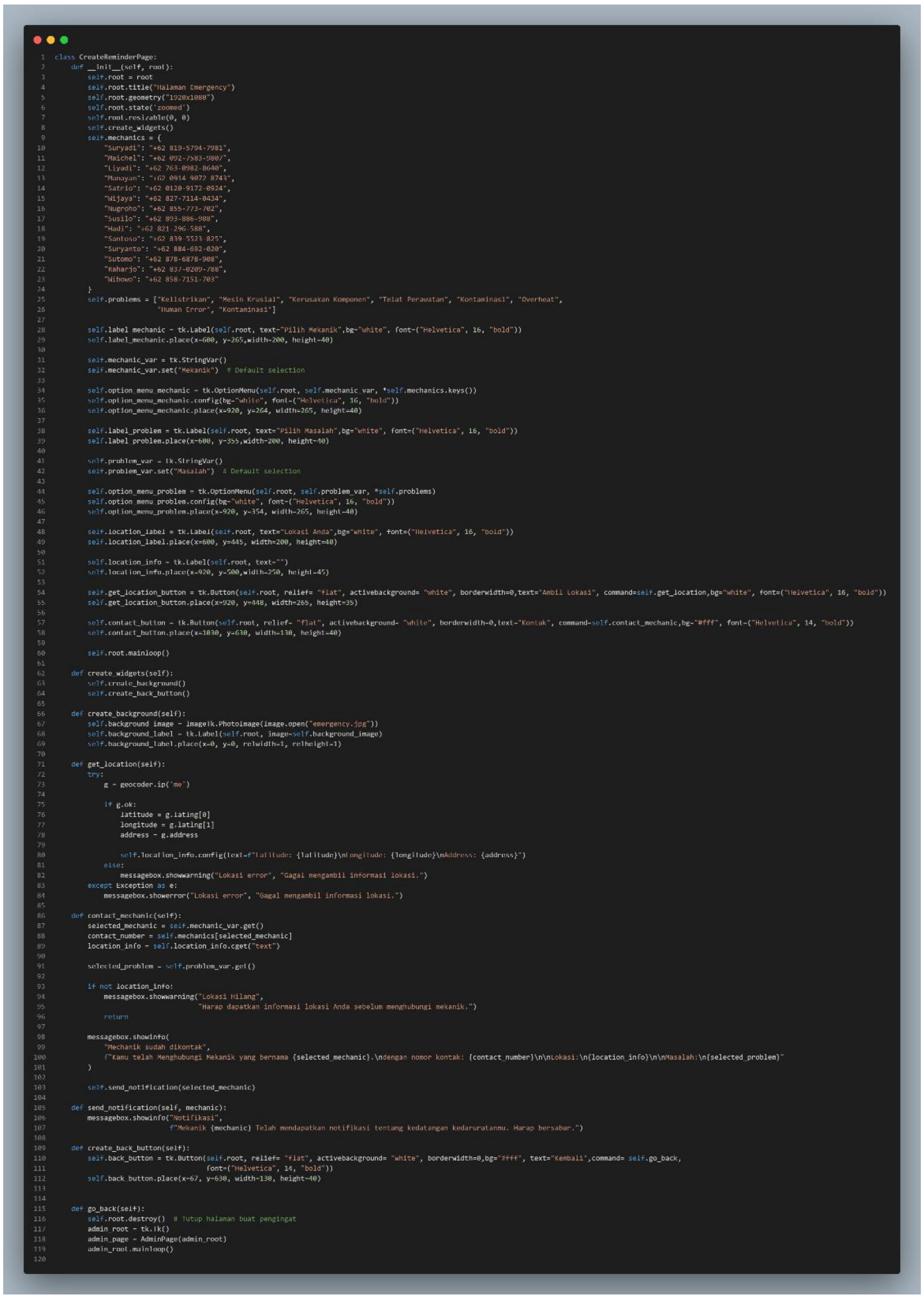
```python
class AddSchedulePage:
    def __init__(self, root):
        self.root = root
        self.root.title("Tambah Penjadwalan")
        self.root.geometry("1920x1080")
        self.root.state('zoomed')
        self.root.resizable(0, 0)
        self.create_widgets()

    def create_widgets(self):
        self.create_background()
        self.create_schedule_form()
        self.create_add_button()
        self.create_back_button()

    def create_background(self):
        self.background_image = ImageTk.PhotoImage(Image.open("jadwal.jpg"))
        self.background_label = tk.Label(self.root, image=self.background_image)
        self.background_label.place(x=0, y=0, relwidth=1, relheight=1)

    def create_schedule_form(self):
        self.project_label = tk.Label(self.root, bg="#fff", text="Projek", font=("Helvetica", 16, "bold"))
        self.project_label.place(x=608, y=179, width=170, height=40)
        self.project_entry = tk.Entry(self.root, bg="#fff", highlightthickness=0, relief="flat", font=("Helvetica", 16, "bold"))
        self.project_entry.place(x=935, y=172, width=230, height=50)
        self.project_entry.placeholder = "Masukkan nama projek"
        self.add_watermark_effect(self.project_entry)

        self.machine_label = tk.Label(self.root, bg="#fff", text="Mesin", font=("Helvetica", 16, "bold"))
        self.machine_label.place(x=608, y=269, width=170, height=40)
        self.machine_entry = tk.Entry(self.root, bg="#fff", highlightthickness=0, relief="flat", font=("Helvetica", 16, "bold"))
        self.machine_entry.place(x=935, y=257, width=230, height=53)
        self.machine_entry.placeholder = "Masukkan nama mesin"
        self.add_watermark_effect(self.machine_entry)

        self.date_label = tk.Label(self.root, bg="#fff", text="Tanggal", font=("Helvetica", 16, "bold"))
        self.date_label.place(x=608, y=359, width=170, height=40)
        self.date_entry = tk.Entry(self.root, bg="#fff", highlightthickness=0, relief="flat",font=("Helvetica", 16, "bold"))
        self.date_entry.place(x=935, y=346, width=230, height=50)
        self.date_entry.placeholder = "YYYY-MM-DD"
        self.add_watermark_effect(self.date_entry)

        self.masalah_label = tk.Label(self.root, bg="#fff", text="Masalah", font=("Helvetica", 16, "bold"))
        self.masalah_label.place(x=608, y=446, width=170, height=40)
        self.list_masalah = ["Kelistrikan", "Kerusakan Mesin Krusial","Kerusakan Komponen","Telat Perawatan","Kontaminasi","Overheat","Human Error"]
        self.masalah_box = ttk.Combobox(self.root, values=self.list_masalah, font=("Helvetica", 16, "bold"))
        self.masalah_box.place(x=935, y=442, width=230, height=50)
        self.masalah_box.set("Pilih masalah")

        self.mekanik_label = tk.Label(self.root, bg="#fff", text="Mekanik", font=("Helvetica", 16, "bold"))
        self.mekanik_label.place(x=608, y=536, width=170, height=40)
        self.list_mekanik = ["Suryadi", "Maichel", "Liyadi", "Manayan","Satrio","Wijaya","Nugroho","Susilo","Hadi","Santoso","Suryanto","Sutomo","Raharjo","Wibowo"]
        self.mekanik_box = ttk.Combobox(self.root, values=self.list_mekanik, font=("Helvetica", 16, "bold"))
        self.mekanik_box.place(x=935, y=530, width=230, height=50)
        self.mekanik_box.set("Pilih mekanik")

    def add_watermark_effect(self, entry):
        entry.insert(0, entry.placeholder)
        entry.configure(foreground="#999", font=("Helvetica", 16, "italic"))

        entry.bind("<FocusIn>", lambda event: self.on_entry_focus_in(entry))
        entry.bind("<FocusOut>", lambda event: self.on_entry_focus_out(entry))

    def on_entry_focus_in(self, entry):
        if entry.get() == entry.placeholder:
            entry.delete(0, tk.END)
            entry.configure(foreground="#000", font=("Helvetica", 16, "normal"))

    def on_entry_focus_out(self, entry):
        if entry.get() == "":
            entry.insert(0, entry.placeholder)
            entry.configure(foreground="#999", font=("Helvetica", 16, "italic"))

    def create_add_button(self):
        self.add_button = tk.Button(self.root, bg="#fff",relief= "flat", activebackground= "white", borderwidth=0, text="Tambah", command=self.add_schedule,
                                    font=("Helvetica", 14, "bold"))
        self.add_button.place(x=1030, y=630, width=130, height=40)

    def create_back_button(self):
        self.back_button = tk.Button(self.root, bg="#fff",relief= "flat", activebackground= "white", borderwidth=0, text="Kembali", command=self.go_back,
                                     font=("Helvetica", 14, "bold"))
        self.back_button.place(x=68, y=630, width=130, height=40)

    def add_schedule(self):
        project = self.project_entry.get()
        machine = self.machine_entry.get()
        masalah = self.masalah_box.get()
        mekanik = self.mekanik_box.get()
        date = self.date_entry.get()
        data = [project, machine, mekanik, masalah, date]

        if not project or not machine or not masalah or not mekanik or not date:
            messagebox.showwarning("Warning", "Mohon lengkapi semua data sebelum menambahkan penjadwalan.")
            return
        with open('penjadwalanmekanik.csv', 'a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(data)

        messagebox.showinfo("Penjadwalan Ditambahkan", f"Penjadwalan untuk Projek: {project}, Mesin: {machine}, Tanggal: {date}, Masalah: {masalah}, Mekanik: {mekanik} telah ditambahkan.")

    def go_back(self):
        self.root.destroy()
        admin_root = tk.Tk()
        admin_page = AdminPage(admin_root)
        admin_root.mainloop()
```



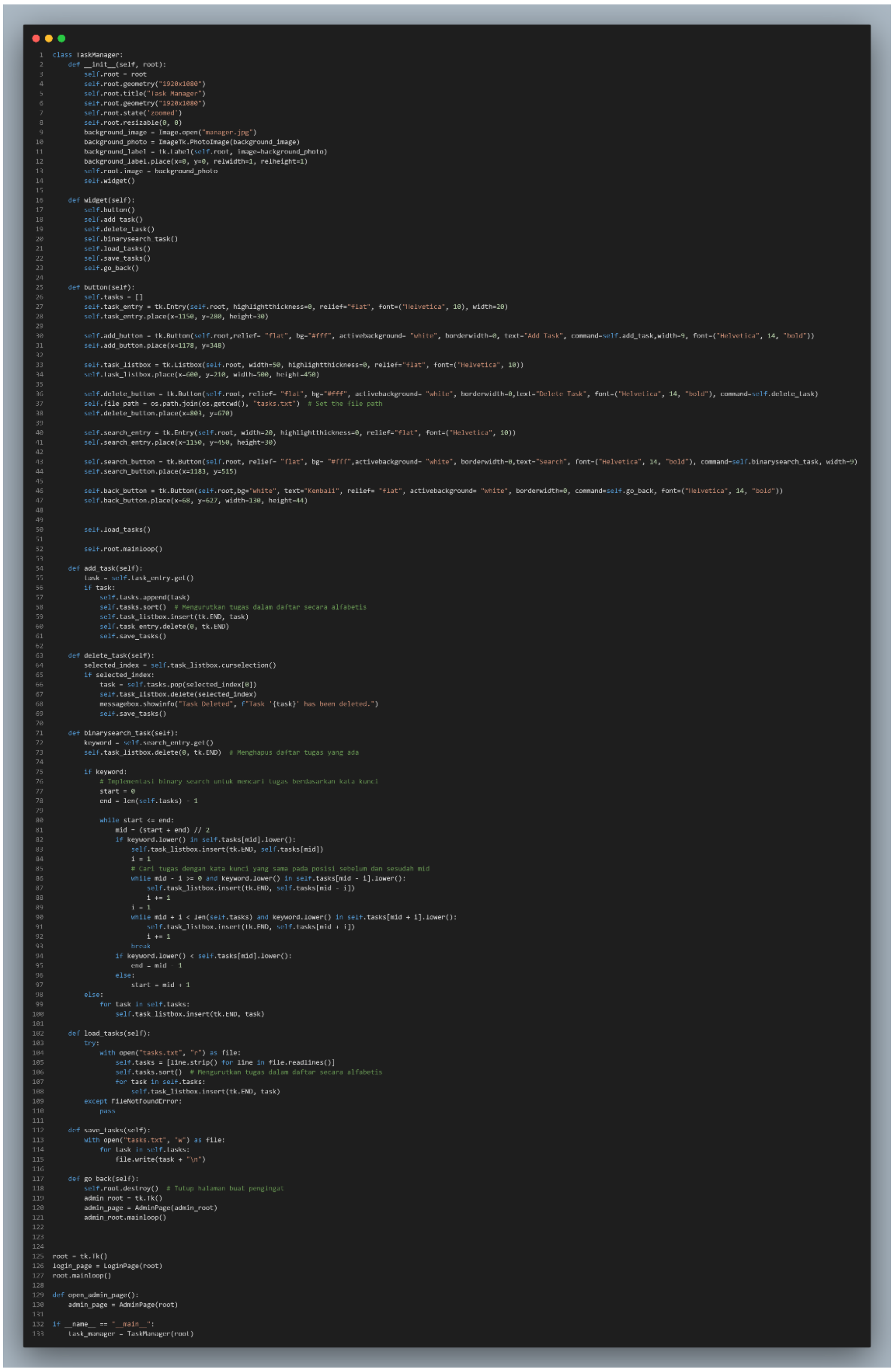
```python
class ViewSchedulePage:
    def __init__(self, root):
        self.root = root
        self.root.title("Lihat Jadwal")
        self.root.geometry("1920x1080")
        self.root.state('zoomed')
        self.root.resizable(0, 0)
        self.create_widgets()

    def create_widgets(self):
        self.create_background()
        self.create_filter_form()
        self.create_view_button()
        self.create_back_button()

    def create_background(self):
        self.background_image = ImageTk.PhotoImage(Image.open("jadwal2.jpg"))
        self.background_label = tk.Label(self.root, image=self.background_image)
        self.background_label.place(x=0, y=0, relwidth=1, relheight=1)

    def create_filter_form(self):
        self.project_label = tk.Label(self.root, bg="#fff", text="Projek", font=("Helvetica", 16, "bold"))
        self.project_label.place(x=607, y=203, width=170, height=40)
        self.project_entry = tk.Entry(self.root,bg="#fff", highlightthickness=0, relief="flat", font=("Helvetica", 16))
        self.project_entry.place(x=937, y=198, width=230, height=50)

        self.machine_label = tk.Label(self.root, bg="#fff", text="Mesin", font=("Helvetica", 16, "bold"))
        self.machine_label.place(x=607, y=290, width=170, height=40)
        self.machine_entry = tk.Entry(self.root,bg="#fff", highlightthickness=0, relief="flat", font=("Helvetica", 16))
        self.machine_entry.place(x=937, y=284, width=230, height=53)

        self.date_label = tk.Label(self.root, bg="#fff", text="Tanggal", font=("Helvetica", 16, "bold"))
        self.date_label.place(x=607, y=378, width=170, height=40)
        self.date_entry = tk.Entry(self.root, bg="#fff", highlightthickness=0, relief="flat", font=("Helvetica", 16))
        self.date_entry.place(x=937, y=373, width=230, height=50)

    def create_view_button(self):
        self.view_button = tk.Button(self.root,bg="#fff",relief= "flat", activebackground= "white", borderwidth=0, text="Lihat", command=self.view_schedule, font=("Helvetica", 14, "bold"))
        self.view_button.place(x=1030, y=629, width=130, height=44)

    def create_back_button(self):
        self.back_button = tk.Button(self.root,bg="#fff",relief= "flat", activebackground= "white", borderwidth=0, text="Kembali", command=self.go_back, font=("Helvetica", 14, "bold"))
        self.back_button.place(x=68, y=629, width=130, height=44)

    def view_schedule(self):
        admin_root = tk.Tk()
        admin_page = Tampilan(admin_root)
        admin_root.mainloop()

    def go_back(self):
        self.root.destroy()
        admin_root = tk.Tk()
        admin_page = AdminPage(admin_root)
        admin_root.mainloop()
```



```python
class Tampilan:
    def __init__(self, root):
        self.root = root
        self.csv_file = "penjadwalanmekanik.csv"
        self.root.title("Tampilan Penjadwalan")
        self.root.geometry("1920x1080")
        self.root.state('zoomed')
        self.root.resizable(0, 0)
        self.create_widgets()
        self.load_data()
        self.binary_search_sentence()

    def create_background(self):
        self.background_image = ImageTk.PhotoImage(Image.open("bg.jpg"))
        self.background_label = tk.Label(self.root, image=self.background_image)
        self.background_label.place(x=0, y=0, relwidth=1, relheight=1)

    def create_widgets(self):
        self.create_background()
        self.frame = ttk.Frame(self.root)
        self.frame.pack()

        self.treeFrame = ttk.Frame(self.frame)
        self.treeFrame.grid(row=0, column=1, pady=50)
        self.treeScroll = ttk.Scrollbar(self.treeFrame)
        self.treeScroll.pack(side="right", fill="y")

        self.cols = ("Project", "Mesin", "Mekanik", "Masalah", "Date")
        self.treeview = ttk.Treeview(self.treeFrame, show="headings", yscrollcommand=self.treeScroll.set, columns=self.cols, height=20)
        self.treeview.column("Project", width=150)
        self.treeview.column("Mesin", width=150)
        self.treeview.column("Mekanik", width=150)
        self.treeview.column("Masalah", width=150)
        self.treeview.column("Date", width=150)
        self.treeview.pack()
        self.treeScroll.config(command=self.treeview.yview)

        self.search_label = tk.Label(self.treeFrame, text="Search Value:")
        self.search_label.pack()
        self.search_entry = tk.Entry(self.treeFrame)
        self.search_entry.pack()

        self.search_button = tk.Button(self.treeFrame,relief= "flat", activebackground= "white", borderwidth=0, text="Search", command=self.search_and_display,bg="white")
        self.search_button.pack()

        self.result_label = tk.Label(self.treeFrame, text="")
        self.result_label.pack()

        self.earliest_button = tk.Button(self.treeFrame,bg="white",relief= "flat", activebackground= "white", borderwidth=0, text="sorting dekat ke jauh", command=self._bubble_sort_near_to_far, font=("Helvetica", 8, "bold"))
        self.earliest_button.place(x=145, y=500, anchor=tk.CENTER, width=250, height=20)

        self.lastest_button = tk.Button(self.treeFrame,bg="white",relief= "flat", activebackground= "white", borderwidth=0, text="Sorting jauh ke dekat", command=self._bubble_sort_far_to_near, font=("Helvetica", 8, "bold"))
        self.lastest_button.place(x=607, y=500, anchor=tk.CENTER, width=250, height=20)

        self.close_button = tk.Button(self.treeFrame, bg="#fff", relief="flat", activebackground="white", borderwidth=0, text="delete", command=self.delete_selected, font=("Helvetica", 8, "bold"))
        self.close_button.place(x=250, y=525, anchor=tk.CENTER, width=50, height=20)

        self.back_button = tk.Button(self.root,bg="#fff",relief= "flat", activebackground= "white", borderwidth=0, text="Kembali", command=self.go_back, font=("Helvetica", 14, "bold"))
        self.back_button.place(x=68, y=627, width=130, height=44)


    def load_data(self):
        self.path = "penjadwalanmekanik.csv"
        with open(self.path, 'r') as file:
            self.csv_reader = csv.reader(file)
            self.list_values = list(self.csv_reader)

        # Menghapus data yang ada sebelumnya pada treeview
        self.treeview.delete(*self.treeview.get_children())

        for col_name in self.list_values[0]:
            self.treeview.heading(col_name, text=col_name)

        for value_list in self.list_values[1:]:
            date = datetime.strptime(value_list[4], "%Y-%m-%d").strftime("%Y-%m-%d")
            value_list[4] = date
            self.treeview.insert("", tk.END, values=value_list)

    def bubble_sort_near_to_far(self):
        sorted_rows = sorted(self.list_values[1:], key=lambda row: datetime.strptime(row[4], "%Y-%m-%d"))

        self.treeview.delete(*self.treeview.get_children())
        for row in sorted_rows:
            self.treeview.insert("", "end", values=row)

    def bubble_sort_far_to_near(self):
        sorted_rows = sorted(self.list_values[1:], key=lambda row: datetime.strptime(row[4], "%Y-%m-%d"), reverse= True)

        self.treeview.delete(*self.treeview.get_children())
        for row in sorted_rows:
            self.treeview.insert("", "end", values=row)

    def delete_selected(self):
        selected_item = self.treeview.focus()  # Dapatkan item yang dipilih
        if selected_item:
            values = self.treeview.item(selected_item)['values']
            name = values[0]  # Kolom nama
            self.delete_from_csv(name)
            self.load_data()

    def delete_from_csv(self,name):
        rows_to_keep = []

        with open('penjadwalanmekanik.csv', 'r') as file:
            reader = csv.reader(file)
            header = next(reader)  # Baca header
            rows_to_keep.append(header)

            for row in reader:
                if row[0] != name:  # Periksa apakah kolom nama tidak sama dengan yang ingin dihapus
                    rows_to_keep.append(row)

        with open('penjadwalanmekanik.csv', 'w', newline='') as file:
            writer = csv.writer(file)
            writer.writerows(rows_to_keep)

    # def binary_search_sentence(self):
    #     import csv

    def binary_search_sentence(self):
        self.search_sentence = self.search_entry.get()
        with open(self.csv_file, 'r') as file:
            reader = csv.reader(file)
            row = list(reader)

        for column_index in range(len(row[0])):
            left = 1
            right = len(row) - 1

            while left <= right:
                mid = (left + right) // 2
                sentence = row[mid][column_index]

                if sentence == self.search_sentence:
                    return row[mid]

                if sentence < self.search_sentence:
                    left = mid + 1
                else:
                    right = mid - 1

        return None

    def display_search_result(self, result):
        if result is None:
            self.result_label.config(text="Pencarian tidak ditemukan")
        else:
            self.treeview.insert("", "end", values=result)
            print("progress",result)

    def search_and_display(self):
        search_value = self.search_entry.get()
        if search_value:

            result = self.binary_search_sentence()
            self.display_search_result(result)

            # Clear previous search results

            results = []
            for child in self.treeview.get_children():
                values = self.treeview.item(child, 'values')
                if search_value.lower() in [str(value).lower() for value in values]:
                    results.append(child)

            self.result_label.config(text=f"Found {len(results)} result(s).")
            self.treeview.selection_set(results)
            if len(results) > 0:
                self.treeview.focus(results[0])
        else:
            self.result_label.config(text="Please enter a search value.")

    def go_back(self):
        self.root.destroy()
        admin_root = tk.Tk()
        admin_page = AdminPage(admin_root)
        admin_root.mainloop()
```



```python
class CreateReminderPage:
    def __init__(self, root):
        self.root = root
        self.root.title("Halaman Emergency")
        self.root.geometry("1920x1080")
        self.root.state('zoomed')
        self.root.resizable(0, 0)
        self.create_widgets()
        self.mechanics = {
            "Suryadi": "+62 819-5794-7981",
            "Maichel": "+62 092-7583-9807",
            "Liyadi": "+62 763-0982-8640",
            "Manayan": "+62 0914 9072 8743",
            "Satrio": "+62 0120-9172-0924",
            "Wijaya": "+62 827-7114-0434",
            "Nugroho": "+62 855-773-702",
            "Susilo": "+62 893-886-988",
            "Hadi": "+62 821-296-588",
            "Santoso": "+62 839-5523-825",
            "Suryanto": "+62 884-692-020",
            "Sutomo": "+62 878-6878-908",
            "Raharjo": "+62 837-0209-788",
            "Wibowo": "+62 858-7151-703"
        }
        self.problems = ["Kelistrikan", "Mesin Krusial", "Kerusakan Komponen", "Telat Perawatan", "Kontaminasi", "Overheat",
                         "Human Error", "Kontaminasi"]

        self.label_mechanic = tk.Label(self.root, text="Pilih Mekanik",bg="white", font=("Helvetica", 16, "bold"))
        self.label_mechanic.place(x=600, y=265,width=200, height=40)

        self.mechanic_var = tk.StringVar()
        self.mechanic_var.set("Mekanik")  # Default selection

        self.option_menu_mechanic = tk.OptionMenu(self.root, self.mechanic_var, *self.mechanics.keys())
        self.option_menu_mechanic.config(bg="white", font=("Helvetica", 16, "bold"))
        self.option_menu_mechanic.place(x=920, y=264, width=265, height=40)

        self.label_problem = tk.Label(self.root, text="Pilih Masalah",bg="white", font=("Helvetica", 16, "bold"))
        self.label_problem.place(x=600, y=355,width=200, height=40)

        self.problem_var = tk.StringVar()
        self.problem_var.set("Masalah")  # Default selection

        self.option_menu_problem = tk.OptionMenu(self.root, self.problem_var, *self.problems)
        self.option_menu_problem.config(bg="white", font=("Helvetica", 16, "bold"))
        self.option_menu_problem.place(x=920, y=354, width=265, height=40)

        self.location_label = tk.Label(self.root, text="Lokasi Anda",bg="white", font=("Helvetica", 16, "bold"))
        self.location_label.place(x=600, y=445, width=200, height=40)

        self.location_info = tk.Label(self.root, text="")
        self.location_info.place(x=920, y=500,width=250, height=45)

        self.get_location_button = tk.Button(self.root, relief= "flat", activebackground= "white", borderwidth=0,text="Ambil Lokasi", command=self.get_location,bg="white", font=("Helvetica", 16, "bold"))
        self.get_location_button.place(x=920, y=448, width=265, height=35)

        self.contact_button = tk.Button(self.root, relief= "flat", activebackground= "white", borderwidth=0,text="Kontak", command=self.contact_mechanic,bg="#fff", font=("Helvetica", 14, "bold"))
        self.contact_button.place(x=1030, y=630, width=130, height=40)

        self.root.mainloop()

    def create_widgets(self):
        self.create_background()
        self.create_back_button()

    def create_background(self):
        self.background_image = ImageTk.PhotoImage(Image.open("emergency.jpg"))
        self.background_label = tk.Label(self.root, image=self.background_image)
        self.background_label.place(x=0, y=0, relwidth=1, relheight=1)

    def get_location(self):
        try:
            g = geocoder.ip('me')

            if g.ok:
                latitude = g.latlng[0]
                longitude = g.latlng[1]
                address = g.address

                self.location_info.config(text=f"Latitude: {latitude}\nLongitude: {longitude}\nAddress: {address}")
            else:
                messagebox.showwarning("Lokasi error", "Gagal mengambil informasi lokasi.")
        except Exception as e:
            messagebox.showerror("Lokasi error", "Gagal mengambil informasi lokasi.")

    def contact_mechanic(self):
        selected_mechanic = self.mechanic_var.get()
        contact_number = self.mechanics[selected_mechanic]
        location_info = self.location_info.cget("text")

        selected_problem = self.problem_var.get()

        if not location_info:
            messagebox.showwarning("Lokasi Hilang",
                                   "Harap dapatkan informasi lokasi Anda sebelum menghubungi mekanik.")
            return

        messagebox.showinfo(
            "Mechanik sudah dikontak",
            f"Kamu telah Menghubungi Mekanik yang bernama {selected_mechanic}.\ndengan nomor kontak: {contact_number}\n\nLokasi:\n{location_info}\n\nMasalah:\n{selected_problem}"
        )

        self.send_notification(selected_mechanic)

    def send_notification(self, mechanic):
        messagebox.showinfo("Notifikasi",
                            f"Mekanik {mechanic} Telah mendapatkan notifikasi tentang kedatangan kedaruratanmu. Harap bersabar.")

    def create_back_button(self):
        self.back_button = tk.Button(self.root, relief= "flat", activebackground= "white", borderwidth=0,bg="#fff", text="Kembali",command= self.go_back,
                                     font=("Helvetica", 14, "bold"))
        self.back_button.place(x=67, y=630, width=130, height=40)


    def go_back(self):
        self.root.destroy()  # Tutup halaman buat pengingat
        admin_root = tk.Tk()
        admin_page = AdminPage(admin_root)
        admin_root.mainloop()

```



```python
class TaskManager:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1920x1080")
        self.root.title("Task Manager")
        self.root.geometry("1920x1080")
        self.root.state('zoomed')
        self.root.resizable(0, 0)
        background_image = Image.open("manager.jpg")
        background_photo = ImageTk.PhotoImage(background_image)
        background_label = tk.Label(self.root, image=background_photo)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        self.root.image = background_photo
        self.widget()

    def widget(self):
        self.button()
        self.add task()
        self.delete_task()
        self.binarysearch task()
        self.load_tasks()
        self.save_tasks()
        self.go_back()

    def button(self):
        self.tasks = []
        self.task_entry = tk.Entry(self.root, highlightthickness=0, relief="flat", font=("Helvetica", 10), width=20)
        self.task_entry.place(x=1150, y=280, height=30)

        self.add_button = tk.Button(self.root,relief= "flat", bg="#fff", activebackground= "white", borderwidth=0, text="Add Task", command=self.add_task,width=9, font=("Helvetica", 14, "bold"))
        self.add_button.place(x=1178, y=348)

        self.task_listbox = tk.Listbox(self.root, width=50, highlightthickness=0, relief="flat", font=("Helvetica", 10))
        self.task_listbox.place(x=600, y=210, width=500, height=450)

        self.delete_button = tk.Button(self.root, relief= "flat", bg="#fff", activebackground= "white", borderwidth=0,text="Delete Task", font=("Helvetica", 14, "bold"), command=self.delete_task)
        self.file path = os.path.join(os.getcwd(), "tasks.txt")  # Set the file path
        self.delete_button.place(x=803, y=670)

        self.search_entry = tk.Entry(self.root, width=20, highlightthickness=0, relief="flat", font=("Helvetica", 10))
        self.search_entry.place(x=1150, y=450, height=30)

        self.search_button = tk.Button(self.root, relief= "flat", bg= "#fff",activebackground= "white", borderwidth=0,text="Search", font=("Helvetica", 14, "bold"), command=self.binarysearch_task, width=9)
        self.search_button.place(x=1183, y=515)

        self.back_button = tk.Button(self.root,bg="white", text="Kembali", relief= "flat", activebackground= "white", borderwidth=0, command=self.go_back, font=("Helvetica", 14, "bold"))
        self.back_button.place(x=68, y=627, width=130, height=44)


        self.load_tasks()

        self.root.mainloop()

    def add_task(self):
        task = self.task_entry.get()
        if task:
            self.tasks.append(task)
            self.tasks.sort()  # Mengurutkan tugas dalam daftar secara alfabetis
            self.task_listbox.insert(tk.END, task)
            self.task entry.delete(0, tk.END)
            self.save_tasks()

    def delete_task(self):
        selected_index = self.task_listbox.curselection()
        if selected_index:
            task = self.tasks.pop(selected_index[0])
            self.task_listbox.delete(selected_index)
            messagebox.showinfo("Task Deleted", f"Task '{task}' has been deleted.")
            self.save_tasks()

    def binarysearch_task(self):
        keyword = self.search_entry.get()
        self.task_listbox.delete(0, tk.END)  # Menghapus daftar tugas yang ada

        if keyword:
            # Implementasi binary search untuk mencari tugas berdasarkan kata kunci
            start = 0
            end = len(self.tasks) - 1

            while start <= end:
                mid = (start + end) // 2
                if keyword.lower() in self.tasks[mid].lower():
                    self.task_listbox.insert(tk.END, self.tasks[mid])
                    i = 1
                    # Cari tugas dengan kata kunci yang sama pada posisi sebelum dan sesudah mid
                    while mid - i >= 0 and keyword.lower() in self.tasks[mid - i].lower():
                        self.task_listbox.insert(tk.END, self.tasks[mid - i])
                        i += 1
                    i = 1
                    while mid + i < len(self.tasks) and keyword.lower() in self.tasks[mid + i].lower():
                        self.task_listbox.insert(tk.END, self.tasks[mid + i])
                        i += 1
                    break
                if keyword.lower() < self.tasks[mid].lower():
                    end = mid - 1
                else:
                    start = mid + 1
        else:
            for task in self.tasks:
                self.task listbox.insert(tk.END, task)

    def load_tasks(self):
        try:
            with open("tasks.txt", "r") as file:
                self.tasks = [line.strip() for line in file.readlines()]
                self.tasks.sort()  # Mengurutkan tugas dalam daftar secara alfabetis
                for task in self.tasks:
                    self.task_listbox.insert(tk.END, task)
        except FileNotFoundError:
            pass

    def save_tasks(self):
        with open("tasks.txt", "w") as file:
            for task in self.tasks:
                file.write(task + "\n")

    def go back(self):
        self.root.destroy()  # Tutup halaman buat pengingat
        admin root = tk.Tk()
        admin_page = AdminPage(admin_root)
        admin_root.mainloop()



root = tk.Tk()
login_page = LoginPage(root)
root.mainloop()

def open_admin_page():
    admin_page = AdminPage(root)

if __name__ == "__main__":
    task_manager = TaskManager(root)
```