LAPORAN PROYEK AKHIR PEMROGRAMAN DASAR
PERANCANGAN DAN IMPLEMENTASI APLIKASI “MYRE” SEBAGAI MANAJEMEN JADWAL DAN PENGINGAT AKADEMIK MAHASISWA

Disusun oleh :

1. Maydina Aulia Putri 25031554143 / 2025G
2. Julia Nur Faizah 25031554146 / 2025G
3. Ahdila Putri Tahta R. 25031554269 / 2025G

Dosen Pengampu:
Hasanuddin Al-Habib, S.Si., M.Si.
Dr. Heri Purnawan, S.Si., M.Si.

PROGRAM STUDI SAINS DATA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SURABAYA

2025

DAFTAR ISI

DAFTAR ISI **2**
BAB I **3**
PENDAHULUAN **3**
1.1 Latar Belakang 3
1.2 Rumusan Masalah 3
1.3 Tujuan 4
BAB II **5**
ANALISIS DAN PERANCANGAN **5**
2.1 Analisis Kebutuhan Aplikasi 5
2.2 Diagram Alur 6
2.3 Sketsa Desain Antarmuka 7
BAB III **14**
IMPLEMENTASI **14**
3.1 Penjelasan Kode Program 14
3.2 Manual Penggunaan & Screenshot Aplikasi 24
LAMPIRAN **31**
DAFTAR PUSTAKA **31**

BAB I

PENDAHULUAN

1.1 Latar Belakang

Mahasiswa merupakan tokoh utama dalam proses pembelajaran di perguruan tinggi. Mereka dituntut untuk mampu beradaptasi dengan berbagai perkembangan teknologi dalam mendukung kegiatan akademik agar lebih efektif dan efisien. Salah satu tantangan yang sering dialami oleh mahasiswa adalah kesulitan dalam mengatur dan mengelola jadwal kuliah secara terstruktur. Ketidakaturan dalam mencatat jadwal, lupa mengikuti perkuliahan, hingga terjadinya tumpang tindih antar jadwal merupakan permasalahan umum yang berdampak langsung pada disiplin, pengelolaan waktu, serta produktivitas belajar secara keseluruhan (Zahwaa et al., 2025).

Catriwati (2022) menyatakan bahwa terdapat mahasiswa yang tidak mengikuti ujian dengan alasan lupa dalam jadwal ujian mereka masing-masing. Hal ini menunjukkan bahwa ketidakteraturan dalam pencatatan jadwal tidak hanya berdampak pada kegiatan perkuliahan harian, tetapi juga pada aktivitas akademik yang bersifat krusial, seperti ujian. Kondisi tersebut menegaskan pentingnya sebuah sistem yang mampu membantu mahasiswa dalam mencatat, mengingatkan, dan mengelola jadwal secara otomatis dan terorganisir, sehingga dapat meminimalisir risiko kelalaian. Oleh karena itu, aplikasi *"Schedule"* dirancang sebagai inovasi untuk membantu mahasiswa dalam menyelenggarakan kegiatan sebagai pengingat saat perkuliah1iannya dan jam sibuk mahasiswa.

1.2 Rumusan Masalah

1. Bagaimana merancang aplikasi jadwal kalender kegiatan dengan menggunakan Python dengan baik?
2. Bagaimana merancang aplikasi "Schedule" yang dapat menyimpan dan mengelola jadwal akademik secara terorganisir?
3. Bagaimana aplikasi dapat menyediakan pengingat otomatis agar mahasiswa tidak terlambat atau melewatkan aktivitas penting?
4. Fitur apa saja yang perlu dikembangkan untuk memenuhi kebutuhan dasar manajemen waktu mahasiswa?

1.3 Tujuan

a. Merancang dan mengembangkan aplikasi sebagai alat bantu pengingat jadwal akademik mahasiswa dengan bahasa pemrograman Python.
b. Memudahkan mahasiswa dalam mengatur jadwal akademik dengan penggunaan aplikasi Schedule Planner berbasis Python.
c. Mengimplementasi kode program yang efisien dan dapat diandalkan untuk memastikan semua fitur utama berfungsi dengan baik.

BAB II

ANALISIS DAN PERANCANGAN

2.1 Analisis Kebutuhan Aplikasi

Aplikasi "MYRE" merupakan aplikasi berupa Graphical User Interface (GUI) atau bisa disebut antarmuka grafis berbasis Python yang dirancang menggunakan pustaka Kivy. Platform ini dirancang dengan tujuan memudahkan mahasiswa mengelola jadwal secara otomatis dan terorganisir sehingga dapat meminimalisir risiko kelalaian. Adapun analisis kebutuhan aplikasi dibagi menjadi tiga, yaitu kebutuhan pengguna, kebutuhan fungsional, dan kebutuhan non-fungsional dengan penjelasan sebagai berikut.

2.1.1 Kebutuhan Pengguna (User Requirement)

a) Mahasiswa membutuhkan aplikasi yang dapat membantu mengingat tugas dan jadwal perkuliahan.
b) Mahasiswa membutuhkan penyimpanan data jadwal dan tugas dalam satu aplikasi terpusat.

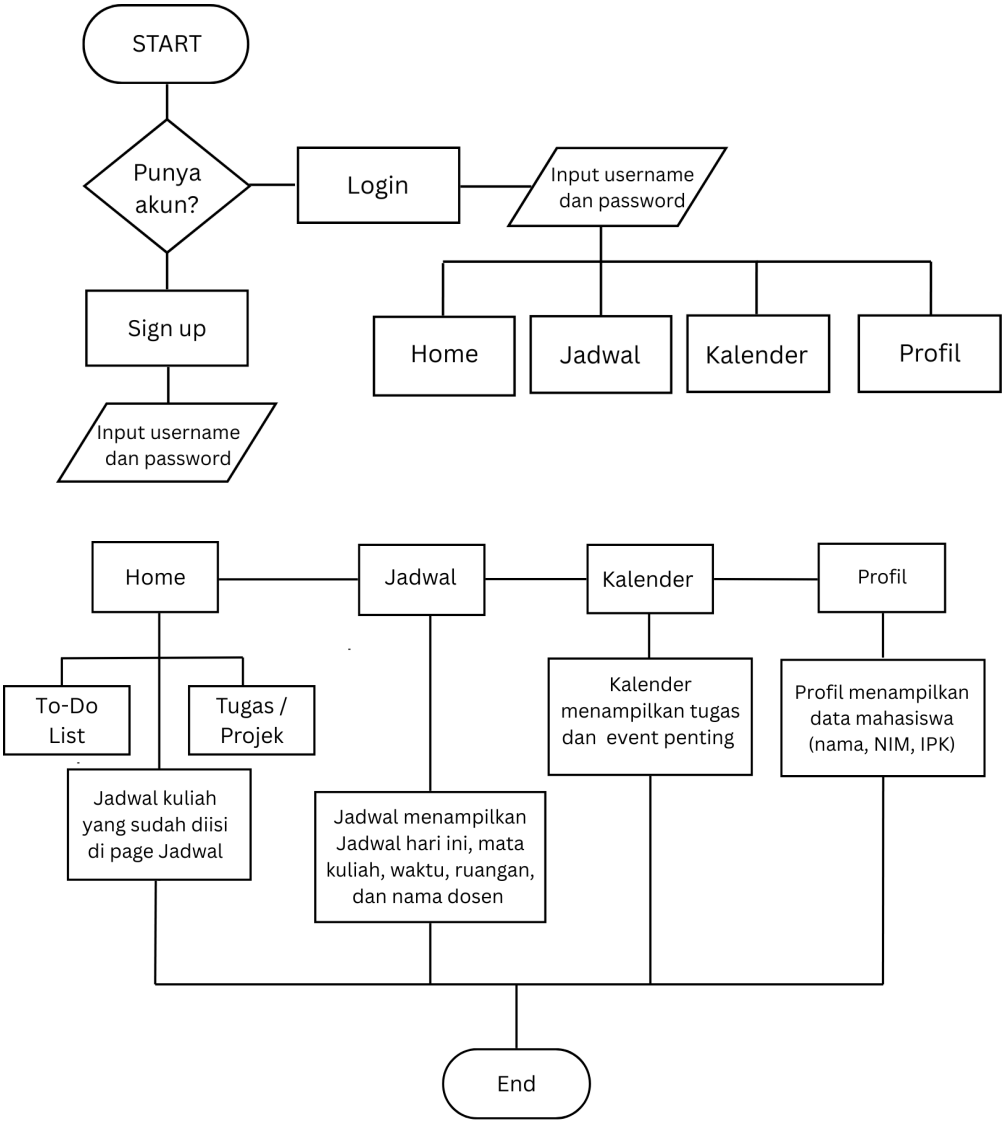
2.1.2 Kebutuhan Fungsional

a) Aplikasi menyediakan antarmuka grafis yang memudahkan pengguna dalam mengakses seluruh fitur MYRE.
b) Aplikasi menyediakan fitur to do list untuk menambah, mengubah, dan menghapus tugas.
c) Aplikasi menampilkan today's schedule yang berisi jadwal dan tugas pada hari berjalan.
d) Aplikasi menyediakan halaman task/projects untuk menambahkan tugas mata kuliah.
e) Aplikasi menyediakan fitur kalender yang memungkinkan pengguna menambahkan event pada tanggal tertentu.

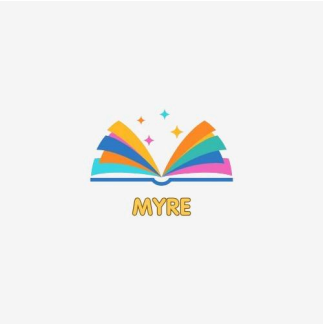
2.1.3 Kebutuhan Non-Fungsional

a) Aplikasi memiliki performa yang baik sehingga mampu menampilkan data dengan cepat.
b) Aplikasi mudah digunakan (user friendly) oleh mahasiswa dari berbagai latar belakang.
c) Aplikasi dapat diakses secara fleksibel kapan saja.
d) Aplikasi memungkinkan pengembangan fitur di masa mendatang.

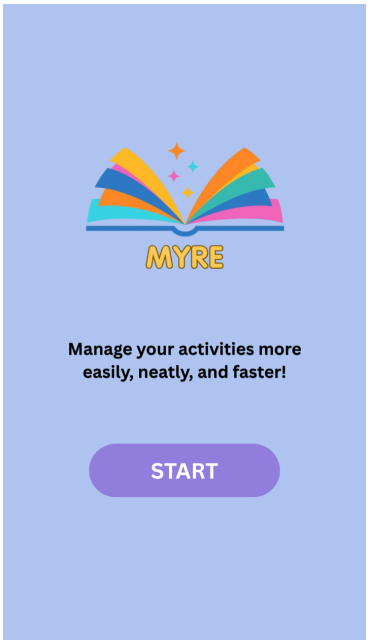
2.2 Diagram Alur



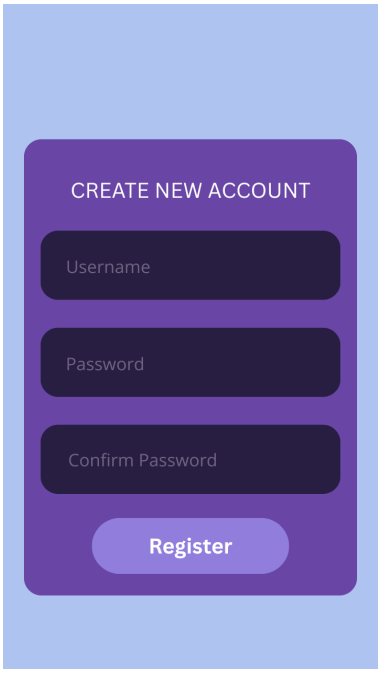
2.3 Sketsa Desain Antarmuka



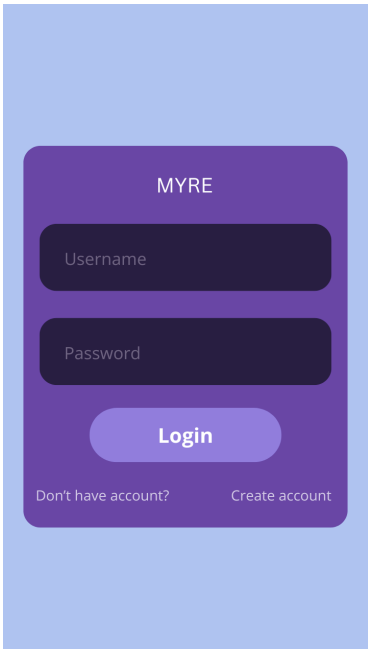
Gambar 1. Logo Platform



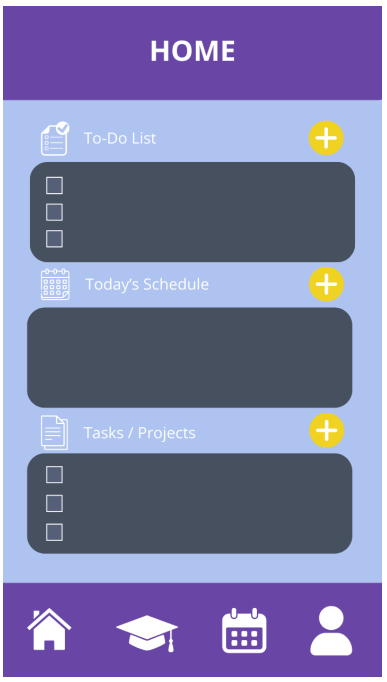
Gambar 2. Tampilan awal



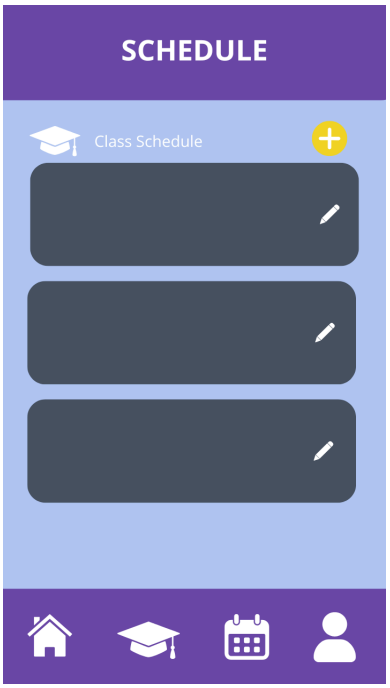
Gambar 3. Tampilan Register



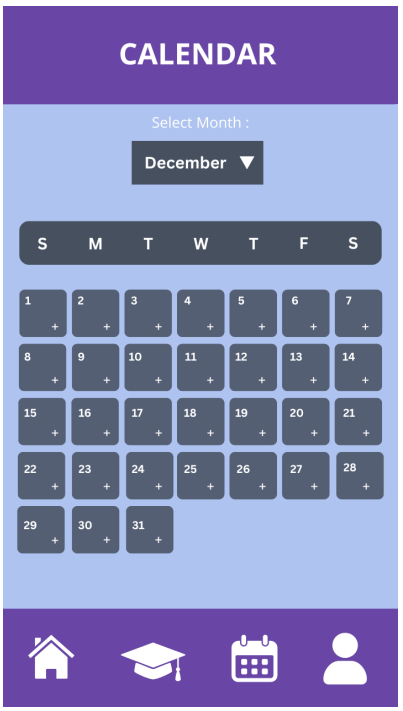
Gambar 4. Tampilan Login



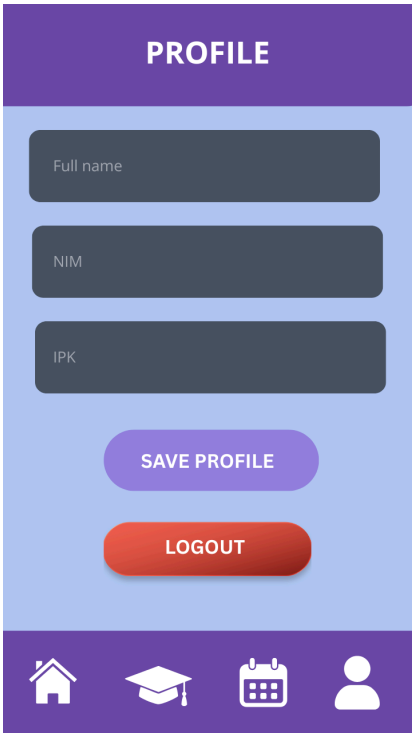
Gambar 5. Tampilan pada halaman Home



Gambar 6. Tampilan pada halaman Jadwal



Gambar 7. Tampilan pada halaman Kalender



Gambar 8. Tampilan pada halaman Profile

BAB III

IMPLEMENTASI

3.1 Penjelasan Kode Program

Implementasi kode pada aplikasi MYRE dibagi ke dalam tiga file utama, yaitu main.py, database.py, dan myre.kv. Pembagian ini dilakukan agar struktur program lebih rapi, mudah dipahami, serta memisahkan antara logika program, pengelolaan data, dan tampilan antarmuka.

3.1.1 Konfigurasi Awal dan Backend Grafis

```
import os
os.environ['KIVY_GL_BACKEND'] = 'angle_sdl2'
from kivy.config import Config
Config.set('graphics', 'width', '360')
Config.set('graphics', 'height', '640')
Config.set('graphics', 'resizable', False)
```

Bagian ini terdapat di file main.py yang digunakan untuk mengatur konfigurasi awal aplikasi sebelum Kivy dijalankan. Pengaturan backend grafis diperlukan agar aplikasi berjalan stabil pada sistem operasi Windows. Selain itu, ukuran jendela diatur agar menyerupai tampilan aplikasi mobile dan tidak dapat diubah ukurannya.

3.1.2 Library dan Framework yang Digunakan

```
#:import dp kivy.metrics.dp
```

Pada file myre.kv terdapat baris kode seperti pada gambar yang digunakan untuk mengimpor fungsi dp (density-independent pixel) dari modul kivy.metrics. Fungsi dp berperan penting dalam pengaturan ukuran komponen antarmuka agar tetap konsisten pada berbagai resolusi layar dan kepadatan pixel perangkat yang berbeda.

```
from datetime import datetime, date
from typing import List, Dict, Any
import openpyxl
from openpyxl import Workbook, load_workbook
import os
```

Pada file database.py terdapat beberapa baris kode import yang digunakan. Impor pertama yakni, modul datetime dan date dari library bawaan python Digunakan untuk mengambil dan mengelola informasi waktu secara lengkap dan mengelola data tanggal. Penggunaan modul ini membantu aplikasi dalam mengelola fitur kalender dan jadwal

secara akurat. Kedua, kami mengimpor tipe data dari modul typing yang berfungsi sebagai type hinting atau penanda tipe data.

Selain itu terdapat Library openpyxl digunakan untuk membaca, menulis, dan memodifikasi file Excel dengan format .xlsx. Dua komponen penting dari library openpyxl yang kami gunakan yakni workbook untuk membuat file excel baru dan load_workbook untuk membuka file Excel yang sudah ada. Terakhir, terdapat Library os yang merupakan library bawaan Python yang digunakan untuk berinteraksi dengan sistem operasi.

```
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen, NoTransition
from kivy.uix.boxlayout import BoxLayout as KBoxLayout
from kivy.uix.scrollview import ScrollView as KScrollView
from kivy.metrics import dp
from kivy.uix.anchorlayout import AnchorLayout
from kivy.animation import Animation

from kivymd.app import MDApp
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.boxlayout import MDBoxLayout
from kivymd.uix.card import MDCard
from kivymd.uix.label import MDLabel
from kivymd.uix.selectioncontrol import MDCheckbox
from kivymd.uix.textfield import MDTextField
from kivymd.uix.button import MDIconButton

from datetime import datetime
import calendar

from database import DatabaseManager
```

Pada file main.py, digunakan beberapa library utama seperti Kivy dan KivyMD yang berfungsi untuk membangun tampilan dan alur aplikasi MYRE. Kivy digunakan sebagai kerangka dasar aplikasi, sedangkan KivyMD digunakan untuk memberikan tampilan berbasis Material Design agar aplikasi terlihat lebih modern dan nyaman digunakan.

Beberapa komponen dari Kivy seperti ScreenManager dan Screen digunakan untuk mengatur perpindahan antar halaman aplikasi, seperti halaman login, register, dan halaman utama. Perpindahan halaman dibuat tanpa animasi dengan menggunakan NoTransition agar aplikasi terasa lebih cepat dan sederhana. Selain itu, BoxLayout, ScrollView, dan AnchorLayout digunakan untuk mengatur tata letak komponen antarmuka, sedangkan Animation digunakan untuk menambahkan efek visual sederhana pada halaman tertentu.

Komponen dari KivyMD seperti MDDialog, MDButton, MDCard, MDLabel, MDCheckbox, dan MDTextField digunakan untuk membuat elemen antarmuka seperti dialog notifikasi, tombol, kartu jadwal, teks, checkbox, serta kolom input. Komponen-komponen ini membantu

menyusun tampilan aplikasi agar lebih rapi dan mudah digunakan oleh pengguna.

Aplikasi ini juga memanfaatkan modul bawaan Python seperti datetime dan calendar untuk mengelola tanggal dan waktu, khususnya pada fitur jadwal dan kalender akademik. Seluruh data aplikasi dikelola melalui kelas DatabaseManager yang berada pada file database.py, sehingga proses penyimpanan dan pengambilan data terpusat dan terorganisir dengan baik.

3.1.3 Screen Manager

```
class MyreScreenManager(ScreenManager):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.current_user = ""
        self.transition = NoTransition()
```

Class berada di file main.py yang memiliki fungsi untuk mengatur perpindahan antar halaman (screen). Variabel current_user digunakan untuk menyimpan username pengguna yang sedang login sehingga dapat digunakan pada screen lain.

3.1.4 Database Manager

```
class DatabaseManager:
    def __init__(self, filename='database.xlsx'):
        self.filename = filename
        self.init_database()

    def init_database(self):
        if not os.path.exists(self.filename):
            wb = Workbook()

            ws_users = wb.active
            ws_users.title = "users"
            ws_users.append(['username', 'password', 'nama', 'nim', 'ipk'])

            ws_jadwal = wb.create_sheet("jadwal")
            ws_jadwal.append(['username', 'hari', 'mata_kuliah', 'waktu', 'ruangan', 'dosen'])

            ws_todo = wb.create_sheet("todo")
            ws_todo.append(['username', 'aktivitas', 'status', 'created_date'])

            ws_tugas = wb.create_sheet("tugas")
            ws_tugas.append(['username', 'nama_tugas', 'mata_kuliah', 'deadline', 'status'])

            ws_kalender = wb.create_sheet("kalender")
            ws_kalender.append(['username', 'tanggal', 'aktivitas', 'waktu'])

            ws_profile = wb.create_sheet("profile")
            ws_profile.append(['username', 'nama', 'nim', 'ipk'])

            wb.save(self.filename)
```

Pada file database.py, terdapat kelas DatabaseManager yang berfungsi sebagai pengelola seluruh data aplikasi MYRE. Kelas ini bertanggung jawab untuk menyiapkan, menyimpan, dan mengatur data pengguna serta data akademik yang digunakan dalam aplikasi. Fungsi

__init__ digunakan sebagai konstruktor yang akan dijalankan secara otomatis saat objek DatabaseManager dibuat. Pada bagian ini, ditentukan nama file database yaitu database.xlsx, kemudian memanggil method init_database() untuk memastikan bahwa file database sudah tersedia sebelum digunakan.

Fungsi init_database berfungsi untuk melakukan inisialisasi database. Method ini akan mengecek apakah file database.xlsx sudah ada atau belum. Jika file belum ada, maka sistem akan membuat file Excel baru menggunakan library openpyxl. Di dalam proses inisialisasi tersebut, dibuat beberapa sheet yang masing-masing memiliki fungsi berbeda. Sheet users digunakan untuk menyimpan data akun pengguna, sheet jadwal untuk menyimpan jadwal perkuliahan, sheet todo untuk aktivitas harian, sheet tugas untuk data tugas atau proyek, sheet kalender untuk aktivitas berdasarkan tanggal, dan sheet profile untuk menyimpan informasi profil pengguna. Setiap sheet dilengkapi dengan header kolom agar struktur data tersusun rapi dan mudah diolah.

3.1.5 Welcome Screen

```
<WelcomeScreen>:
    name: "welcome"

    canvas.before:
        Color:
            rgba: 0.1, 0.1, 0.25, 1
        Rectangle:
            pos: self.pos
            size: self.size

    FloatLayout:

        MDCard:
            id: box
            orientation: "vertical"
            spacing: "40dp"
            padding: "40dp"
            size_hint: .8, None
            height: self.minimum_height
            pos_hint: {"center_x": 0.5, "center_y": 0.5}
            radius: [20]
            elevation: 10
            md_bg_color: rgba(0.15, 0.15, 0.25, 0.9)
            opacity: 0  # Start invisible

            MDIcon:
                icon: "calendar-check-outline"
                halign: "center"
                theme_text_color: "Primary"
                font_size: "48sp"
                size_hint_y: None
                height: "60dp"
```

```
        MDLabel:
            text: "MYRE"
            halign: "center"
            theme_font_style: "H1"
            font_size: "48sp"
            size_hint_y: None
            height: "60dp"

        MDLabel:
            text: "Manage your activities more easily, more neatly, faster."
            halign: "center"
            theme_text_color: "Secondary"
            font_size: "16sp"
            size_hint_y: None
            height: "50dp"

        MDRaisedButton:
            id: btn_next
            text: "Start"
            size_hint: .6, None
            height: "50dp"
            pos_hint: {"center_x": 0.5}
            md_bg_color: app.theme_cls.primary_color
            on_release:
                app.root.current = "login"
```

Potongan file myre.kv diatas berfungsi untuk mengatur seluruh komponen tampilan, seperti latar belakang, kartu utama, ikon, dan tombol. Tombol Start digunakan untuk berpindah ke halaman login melalui screen manager.

```
class WelcomeScreen(Screen):
    def on_enter(self):
        anim = Animation(opacity=1, duration=0.9, t="out_quad")
        anim.start(self.ids.box)
```

Sedangkan di file main.py terdapat fungsi on_enter() yang dijalankan untuk menampilkan animasi fade-in pada kartu utama.

3.1.6 Register Screen

```
<RegisterScreen>:
    canvas.before:
        Color:
            rgba: 0.1, 0.1, 0.3, 1
        Rectangle:
            pos: self.pos
            size: self.size

    AnchorLayout:
        anchor_x: "center"
        anchor_y: "center"

        MDCard:
            orientation: 'vertical'
            size_hint: None, None
            size: "340dp", self.minimum_height
            padding: "24dp"
            spacing: "24dp"
            radius: [20]
            elevation: 8
            md_bg_color: rgba(0.15, 0.15, 0.25, 0.9)

            MDLabel:
                text: "Create New Account"
                theme_font_style: "H4"
                halign: "center"

            MDTextField:
                id: reg_username
                hint_text: "Username"
                mode: "fill"
                fill_color: rgba(0.2, 0.2, 0.3, 0.8)
                radius: [10]
```

```
            MDTextField:
                id: reg_password
                hint_text: "Password"
                password: True
                mode: "fill"
                fill_color: rgba(0.2, 0.2, 0.3, 0.8)
                radius: [10]

            MDTextField:
                id: reg_confirm
                hint_text: "Confirm Password"
                password: True
                mode: "fill"
                fill_color: rgba(0.2, 0.2, 0.3, 0.8)
                radius: [10]

            MDRaisedButton:
                text: "REGISTER"
                size_hint_x: 0.8
                pos_hint: {"center_x": 0.5}
                md_bg_color: app.theme_cls.primary_color
                on_release: root.try_register()

            MDTextButton:
                text: "Back to Login"
                on_release: app.root.current = "login"
```

Pada file myre.kv, Register Screen dirancang menggunakan beberapa komponen MDTextField untuk menerima input username, password, dan konfirmasi password. Setiap field diberikan id agar nilainya dapat diakses dan diproses pada file main.py. Tombol REGISTER digunakan untuk memicu proses pendaftaran pengguna dengan memanggil fungsi try_register.

```python
class RegisterScreen(Screen):
    def try_register(self):
        username = self.ids.reg_username.text
        password = self.ids.reg_password.text
        confirm = self.ids.reg_confirm.text
        if not username or not password:
            self.show_error_dialog("Error", "Username and password must be filled!")
            return
        if password != confirm:
            self.show_error_dialog("Error", "Password does not match!")
            return
        db = DatabaseManager()
        if db.register_user(username, password):
            self.show_success_dialog("Success", "Registration successful! Please login.")
        else:
            self.show_error_dialog("Error", "Username already used!")

    def show_error_dialog(self, title, text):
        self.dialog = MDDialog(
            title=title,
            text=text,
            buttons=[MDFlatButton(text="OK", on_release=lambda x: self.dialog.dismiss())],
        )
        self.dialog.open()

    def show_success_dialog(self, title, text):
        def go_to_login(_):
            self.dialog.dismiss()
            self.manager.current = 'login'

        self.dialog = MDDialog(
            title=title,
            text=text,
            buttons=[MDFlatButton(text="OK", on_release=go_to_login)],
        )
        self.dialog.open()
```

Di file main.py, pada Class Register Screen terdapat beberapa fungsi yakni try_register, show_error_dialog, dan show_succes_dialog. Fungsi try_register berfungsi untuk mengambil data input dari antarmuka pengguna. Data yang dimasukkan kemudian divalidasi, termasuk pengecekan apakah seluruh field telah diisi dan apakah password sesuai dengan konfirmasi password. Jika validasi gagal, aplikasi akan menampilkan dialog kesalahan. Jika validasi berhasil, data pengguna akan diteruskan ke modul database untuk disimpan.

Keberhasilan atau errornya registrasi ditampilkan oleh fungsi show_error_dialog, dan show_succes_dialog setelah proses registrasi. Aplikasi menampilkan dialog pemberitahuan. Ketika pengguna menekan tombol OK, aplikasi secara otomatis mengarahkan pengguna kembali ke halaman login ketika berhasil agar dapat masuk menggunakan akun yang baru dibuat.

```python
class DatabaseManager:
    def register_user(self, username: str, password: str) -> bool:
        wb = load_workbook(self.filename)
        ws = wb["users"]

        for row in ws.iter_rows(min_row=2, values_only=True):
            if row and row[0] == username:
                wb.close()
                return False

        ws.append([username, password, '', '', ''])

        ws_profile = wb["profile"]
        ws_profile.append([username, '', '', ''])

        wb.save(self.filename)
        wb.close()
        return True
```

Pada file database.py, khususnya di Class DatabaseManager terdapat fungsi register_user yang mengatur proses penyimpanan data registrasi. Fungsi ini bertugas memasukkan data username dan password yang diinput pengguna ke dalam database Excel.

3.1.7 Login Screen

```
<LoginScreen>:
    canvas.before:
        Color:
            rgba: 0.1, 0.1, 0.3, 1
        Rectangle:
            pos: self.pos
            size: self.size

    AnchorLayout:
        anchor_x: "center"
        anchor_y: "center"

        MDCard:
            orientation: 'vertical'
            size_hint: None, None
            size: "340dp", self.minimum_height
            padding: "24dp"
            spacing: "24dp"
            radius: [20]
            elevation: 8
            md_bg_color: rgba(0.15, 0.15, 0.25, 0.9)

            MDLabel:
                text: "MYRE"
                theme_font_style: "H3"
                halign: "center"

            MDTextField:
                id: username
                hint_text: "Username"
                mode: "fill"
                fill_color: rgba(0.2, 0.2, 0.3, 0.8)
                radius: [10]
```

```
        MDTextField:
            id: password
            hint_text: "Password"
            password: True
            mode: "fill"
            fill_color: rgba(0.2, 0.2, 0.3, 0.8)
            radius: [10]

        MDRaisedButton:
            text: "LOGIN"
            size_hint_x: 0.8
            pos_hint: {"center_x": 0.5}
            md_bg_color: app.theme_cls.primary_color
            on_release: root.try_login()

        MDBoxLayout:
            orientation: 'horizontal'
            spacing: "5dp"
            size_hint_y: None
            height: self.minimum_height

            MDLabel:
                text: "Don't have an account?"
                theme_font_style: "Body2"

            MDTextButton:
                text: "Create account"
                on_release: app.root.current = "register"
```

Pada file myre.kv, Login Screen dirancang menggunakan beberapa komponen MDTextField untuk menerima input username dan password. Setiap field diberikan id agar nilainya dapat diakses dan diproses pada file main.py. Tombol Login digunakan untuk memicu proses pendaftaran pengguna dengan memanggil fungsi try_login serta tombol Create account digunakan untuk membuka halaman register.

```
class LoginScreen(Screen):
    def try_login(self):
        username = self.ids.username.text
        password = self.ids.password.text
        if not username or not password:
            self.show_error_dialog("Error", "Username and password must be filled!")
            return
        db = DatabaseManager()
        if db.login_user(username, password):
            self.manager.current_user = username
            self.manager.current = 'home'
            self.ids.username.text = ""
            self.ids.password.text = ""
        else:
            self.show_error_dialog("Login Failed", "Username or password is wrong!")

    def show_error_dialog(self, title, text):
        self.dialog = MDDialog(
            title=title,
            text=text,
            buttons=[
                MDFlatButton(text="OK", on_release=lambda x: self.dialog.dismiss())
            ],
        )
        self.dialog.open()
```

Di file main.py pada Class Login Screen terdapat beberapa fungsi yakni try_login dan show_error_dialog. Fungsi try_login digunakan untuk mengambil nilai username dan password yang dimasukkan oleh

pengguna. Data ini kemudian divalidasi untuk memastikan bahwa field tidak kosong sebelum dilakukan pengecekan ke database.

Jika data tidak lengkap atau tidak valid, aplikasi akan menampilkan dialog kesalahan. Jika data valid, proses dilanjutkan ke pengecekan kecocokan data dengan database.

```
class DatabaseManager:
    def login_user(self, username: str, password: str) -> bool:
        wb = load_workbook(self.filename)
        ws = wb["users"]

        for row in ws.iter_rows(min_row=2, values_only=True):
            if row and row[0] == username and row[1] == password:
                wb.close()
                return True
        wb.close()
        return False
```

Pada file database.py, fungsi login digunakan untuk mencocokkan data yang dimasukkan pengguna dengan data yang tersimpan di database. Jika data ditemukan, fungsi mengembalikan hasil yang menandakan bahwa login berhasil. Jika data pengguna ditemukan di database, aplikasi akan mengarahkan pengguna ke halaman Home. Sebaliknya, jika data tidak cocok, aplikasi menampilkan dialog kesalahan untuk memberi informasi bahwa login gagal.

3.1.8 Button Navbar

```
<NavBar@MDBoxLayout>:
    orientation: 'horizontal'
    size_hint_y: None
    height: "60dp"
    padding: "15dp", "5dp", "5dp", "5dp"
    spacing: "16dp"
    md_bg_color: app.theme_cls.primary_color

    Widget:
        size_hint_x: 30

    MDFillRoundFlatIconButton:
        icon: "home"
        size_hint_x: None
        width: "80dp"
        icon_size: "36dp"
        on_release: app.root.current = 'home'

    MDFillRoundFlatIconButton:
        icon: "school"
        size_hint_x: None
        width: "80dp"
        icon_size: "36dp"
        on_release: app.root.current = 'schedule'

    MDFillRoundFlatIconButton:
        icon: "calendar"
        size_hint_x: None
        width: "80dp"
        icon_size: "36dp"
        on_release: app.root.current = 'calendar'

    MDFillRoundFlatIconButton:
        icon: "account"
        size_hint_x: None
        width: "80dp"
        icon_size: "36dp"
        on_release: app.root.current = 'profile'
```

Navbar didefinisikan menggunakan template widget <NavBar@MDBoxLayout>. Artinya, navbar ini bukan screen, tetapi komponen yang bisa dipakai ulang di banyak halaman. Navbar berisi empat tombol navigasi, yaitu icon home yang berfungsi untuk berpindah ke home screen, icon school yang berfungsi untuk berpindah ke schedule screen, icon calendar yang berfungsi untuk berpindah ke calendar screen, dan icon account yang berfungsi untuk berpindah ke profile screen. Pada setiap halaman, Navbar dipanggil langsung sehingga tampilannya akan konsisten di seluruh halaman.

3.1.9 Home Screen

```
<HomeScreen>:
    BoxLayout:
        orientation: 'vertical'

        MDTopAppBar:
            title: "Home"
            elevation: 10
            md_bg_color: app.theme_cls.primary_color
            size_hint_y: None
            height: "32dp"

        ScrollView:
            size_hint_y: 1
            do_scroll_x: False
            do_scroll_y: True

            MDBoxLayout:
                orientation: 'vertical'
                padding: "20dp"
                spacing: "30dp"
                size_hint_y: None
                height: self.minimum_height
                canvas.before:
                    Color:
                        rgba: 0.05, 0.05, 0.1, 1
                    Rectangle:
                        pos: self.pos
                        size: self.size
```

Tampilan dasar homescreen didefinisikan pada file myre.kv. Home Screen menggunakan BoxLayout dengan orientasi vertikal agar setiap komponen ditampilkan secara berurutan dari atas ke bawah. Terdapat fungsi untuk menampilkan header halaman atau judul halaman Home. Hal ini ditampilkan untuk memberikan identitas halaman yang sedang aktif kepada pengguna.

```
class HomeScreen(Screen):
    def on_enter(self, *args):
        self.load_todos()
        self.load_jadwal_hari_ini()
        self.load_tugas()
```

Pada main.py, Home Screen direpresentasikan oleh class HomeScreen. Class ini berfungsi untuk mengatur logika pemuatan dan pengelolaan data yang ditampilkan pada halaman utama aplikasi. Fungsi on_enter akan dieksekusi setiap kali pengguna memasuki Home Screen. Fungsi ini digunakan sebagai pemicu utama untuk memuat ulang seluruh data yang ditampilkan, sehingga informasi yang muncul selalu sesuai dengan kondisi terbaru pada database. Pada tahap ini, aplikasi memanggil tiga fungsi utama, yaitu load_todos, load_jadwal_hari_ini, dan load_tugas.

Tiga fungsi utama yang dipanggil merupakan struktur bagian dari Home Screen itu sendiri, masing-masing sebagai berikut.

3.1.9.1 To-Do List

```
MDBoxLayout:
    orientation: 'horizontal'
    size_hint_y: None
    height: "56dp"
    spacing: "10dp"

    MDIcon:
        icon: "clipboard-list"
        theme_text_color: "Primary"
        size_hint: None, None
        size: "40dp", "56dp"
        pos_hint: {"center_y": 0.5}

    MDLabel:
        text: "To-Do List"
        theme_font_style: "H4"
        theme_text_color: "Primary"
        size_hint: 1, None
        height: "56dp"
        pos_hint: {"center_y": 0.5}
        text_size: 9999, None
        max_lines: 1

    MDIconButton:
        icon: "plus"
        size_hint: None, None
        size: "32dp", "32dp"
        md_bg_color: app.theme_cls.accent_color
        pos_hint: {"center_y": 0.5}
        on_release: root.add_todo()
```

```
MDCard:
    orientation: "vertical"
    padding: "20dp"
    radius: 20
    elevation: 6
    size_hint_y: None
    height: min(todo_container.height + dp(60), dp(320))
    md_bg_color: [0, 0, 0, 0]

    ScrollView:
        bar_width: 0
        do_scroll_x: False

        MDList:
            id: todo_container
            size_hint_y: None
            height: self.minimum_height
```

Bagian To-Do List digunakan untuk menampilkan daftar aktivitas yang harus dilakukan pengguna. Daftar ini ditampilkan menggunakan MDList agar dapat memuat banyak item dan mendukung scroll. Tombol tambah (+) disediakan untuk menambahkan aktivitas baru.

```python
def load_todos(self):
    todo_container = self.ids.todo_container
    todo_container.clear_widgets()

    db = DatabaseManager()
    todos = db.get_todos(self.manager.current_user)

    for todo in todos:
        box = MDBoxLayout(orientation='horizontal', adaptive_height=True, spacing="10dp", padding=("8dp","6dp"))
        checkbox = MDCheckbox(
            size_hint=(None, None),
            size=("40dp", "40dp"),
            active=True if todo.get('status') == 'completed' else False
        )
        checkbox.bind(active=lambda instance, value, t=todo: self.toggle_todo(t, value))

        label = MDLabel(
            text=todo.get('aktivitas', ''),
            halign="left",
            theme_text_color="Primary",
            size_hint_x=0.8
        )
        if todo.get('status') == 'completed':
            label.theme_text_color = "Secondary"

        edit_icon = MDIconButton(
            icon="pencil",
            size_hint=(None, None),
            size=("32dp", "32dp"),
            icon_size="16dp",
            on_release=lambda x, t=todo: self.edit_todo(t)
        )

        box.add_widget(checkbox)
        box.add_widget(label)
        box.add_widget(edit_icon)
```

```python
def load_todos(self):
        card = MDCard(orientation='horizontal', padding="6dp", size_hint_y=None, height="56dp", elevation=0, md_bg_color=[0, 0, 0, 0])
        card.add_widget(box)
        todo_container.add_widget(card)

def toggle_todo(self, todo, value):
    db = DatabaseManager()
    new_status = 'completed' if value else 'pending'
    db.update_todo_status(self.manager.current_user, todo['aktivitas'], new_status)
    self.load_todos()

def add_todo(self):
    content = MDBoxLayout(orientation="vertical", spacing="12dp", size_hint_y=None, height="50dp")
    aktivitas_field = MDTextField(hint_text="Activity", mode="fill")
    content.add_widget(aktivitas_field)

    def save_todo(_):
        if aktivitas_field.text:
            db = DatabaseManager()
            db.add_todo(self.manager.current_user, aktivitas_field.text)
            self.dialog.dismiss()
            self.load_todos()

    self.dialog = MDDialog(
        title="Add Activity",
        type="custom",
        content_cls=content,
        buttons=[
            MDFlatButton(text="CANCEL", on_release=lambda x: self.dialog.dismiss()),
            MDFlatButton(text="ADD", on_release=save_todo),
        ],
    )
    self.dialog.open()
```

```python
class HomeScreen(Screen):
    def edit_todo(self, todo):
        content = MDBoxLayout(orientation="vertical", spacing="12dp", size_hint_y=None, height="50dp")
        aktivitas_field = MDTextField(hint_text="Activity", mode="fill", text=todo.get('aktivitas', ''))
        content.add_widget(aktivitas_field)

        def save_todo(_):
            if aktivitas_field.text:
                db = DatabaseManager()
                db.update_todo_aktivitas(self.manager.current_user, todo['aktivitas'], aktivitas_field.text)
                self.dialog.dismiss()
                self.load_todos()

        def delete_todo(_):
            db = DatabaseManager()
            db.delete_todo(self.manager.current_user, todo['aktivitas'])
            self.dialog.dismiss()
            self.load_todos()

        self.dialog = MDDialog(
            title="Edit Activity",
            type="custom",
            content_cls=content,
            buttons=[
                MDFlatButton(text="DELETE", on_release=delete_todo, text_color=(1, 0, 0, 1)),
                MDFlatButton(text="CANCEL", on_release=lambda x: self.dialog.dismiss()),
                MDFlatButton(text="SAVE", on_release=save_todo),
            ],
        )
        self.dialog.open()
```

Pada file main.py terdapat beberapa fungsi yang mengatur to-do-list. Fungsi load_todos digunakan untuk menampilkan daftar aktivitas harian pengguna pada Home Screen. Pada awal fungsi, aplikasi mengambil komponen antarmuka todo_container yang didefinisikan pada file myre.kv. Seluruh widget yang ada di dalam container tersebut dihapus terlebih dahulu menggunakan clear_widgets untuk mencegah terjadinya duplikasi data ketika proses pemuatan ulang dilakukan.

Selanjutnya, aplikasi membuat objek DatabaseManager untuk mengakses database. Fungsi get_todos dipanggil dengan parameter pengguna yang sedang login, sehingga data yang diambil hanya milik pengguna tersebut. Setiap data to-do yang diperoleh ditampilkan dalam bentuk kartu yang berisi checkbox, teks aktivitas, dan tombol edit. Status checkbox disesuaikan dengan nilai status pada database.

Selain itu, terdapat fungsi add_todo yang digunakan untuk menampilkan dialog input yang memungkinkan pengguna memasukkan aktivitas baru. Fungsi edit_todo digunakan untuk mengubah isi aktivitas yang telah ada. Dialog edit akan menampilkan teks aktivitas sebelumnya agar dapat diperbarui oleh pengguna. Penghapusan to-do juga dilakukan melalui dialog edit dengan fungsi delete_todo. Setiap perubahan akan langsung disimpan ke database.

```python
class DatabaseManager:
    def add_todo(self, username: str, aktivitas: str):
        wb = load_workbook(self.filename)
        ws = wb["todo"]

        ws.append([username, aktivitas, 'pending', datetime.now().strftime('%Y-%m-%d %H:%M:%S')])
        wb.save(self.filename)
        wb.close()

    def get_todos(self, username: str) -> List[Dict]:
        wb = load_workbook(self.filename)
        ws = wb["todo"]

        todos = []
        for row in ws.iter_rows(min_row=2, values_only=True):
            if row and row[0] == username:
                todos.append({
                    'aktivitas': row[1],
                    'status': row[2],
                    'created_date': row[3]
                })

        wb.close()
        return todos

    def update_todo_status(self, username: str, aktivitas: str, status: str):
        wb = load_workbook(self.filename)
        ws = wb["todo"]

        for row in ws.iter_rows(min_row=2):
            if row[0].value == username and row[1].value == aktivitas:
                row[2].value = status
                break

        wb.save(self.filename)
        wb.close()
```

```python
class DatabaseManager:
    def update_todo_aktivitas(self, username: str, old_aktivitas: str, new_aktivitas: str):
        wb = load_workbook(self.filename)
        ws = wb["todo"]

        for row in ws.iter_rows(min_row=2):
            if row[0].value == username and row[1].value == old_aktivitas:
                row[1].value = new_aktivitas
                break

        wb.save(self.filename)
        wb.close()

    def delete_todo(self, username: str, aktivitas: str):
        wb = load_workbook(self.filename)
        ws = wb["todo"]

        rows_to_delete = []
        for row_idx, row in enumerate(ws.iter_rows(min_row=2), start=2):
            if row[0].value == username and row[1].value == aktivitas:
                rows_to_delete.append(row_idx)

        for row_idx in reversed(rows_to_delete):
            ws.delete_rows(row_idx)

        wb.save(self.filename)
        wb.close()
```

Perubahan di database terjadi dalam beberapa kondisi, yaitu penambahan aktivitas, perubahan status aktivitas, pengeditan aktivitas, dan penghapusan aktivitas. Fungsi add_todo digunakan untuk menambahkan data aktivitas baru ke dalam database, fungsi get_todos berfungsi untuk mengambil seluruh data to-do milik pengguna tertentu dari database, fungsi update_todo_status digunakan untuk mengubah status aktivitas pada to-do list, fungsi update_todo_aktivitas berfungsi untuk mengedit atau memperbarui nama aktivitas yang sudah ada, dan fungsi delete_todo digunakan untuk menghapus data aktivitas secara permanen dari database.

3.1.9.2 Jadwal Hari Ini (Today's Schedule)

```
MDBoxLayout:
    orientation: 'horizontal'
    size_hint_y: None
    height: "56dp"
    spacing: "10dp"

    MDIcon:
        icon: "calendar-today"
        theme_text_color: "Primary"
        size_hint: None, None
        size: "40dp", "56dp"
        pos_hint: {"center_y": 0.5}

    MDLabel:
        text: "Today's Schedule"
        theme_font_style: "H4"
        theme_text_color: "Primary"
        size_hint: 1, None
        height: "56dp"
        pos_hint: {"center_y": 0.5}
        text_size: self.width, None
        max_lines: 1

MDCard:
    orientation: 'vertical'
    padding: "20dp"
    radius: 20
    elevation: 6
    size_hint_y: None
    height: min(jadwal_container.height + dp(60), dp(320))
    md_bg_color: [0, 0, 0, 0]
```

```
MDBoxLayout:
    MDCard:
        ScrollView:
            bar_width: 0
            do_scroll_x: False

            MDList:
                id: jadwal_container
                size_hint_y: None
                height: self.minimum_height
```

Tampilan layout Today's Schedule diatur dalam file myre.kv yang disusun menggunakan kombinasi MDBoxLayout, MDCard, ScrollView, dan MDList. Mulai dari header judul dengan penggunaan orientation 'horizontal' yang berisi judul fitur serta icon serta terdapat juga wadah utama untuk menampilkan daftar jadwal.

```
def load_jadwal_hari_ini(self):
    jadwal_container = self.ids.jadwal_container
    jadwal_container.clear_widgets()

    days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
    today = datetime.today().weekday()
    hari_ini = days[today]

    db = DatabaseManager()
    jadwal_list = db.get_jadwal_hari_ini(self.manager.current_user, hari_ini)

    if not jadwal_list:
        card = MDCard(orientation='vertical', padding="8dp", size_hint_y=None, height="56dp", elevation=0, md_bg_color=[0, 0, 0, 0])
        card.add_widget(MDLabel(text="No schedule today", halign="left"))
        jadwal_container.add_widget(card)
    else:
        for jadwal in jadwal_list:
            card = MDCard(orientation='vertical', padding="8dp", size_hint_y=None, height="80dp", spacing="2dp", elevation=0, md_bg_color=[0, 0, 0, 0])
            title_box = MDBoxLayout(orientation='horizontal', adaptive_height=True, spacing="10dp")
            title_label = MDLabel(text=jadwal.get('mata_kuliah', ''), theme_font_style="H6", halign="left", size_hint_x=1)
            title_box.add_widget(title_label)
            card.add_widget(title_box)
            card.add_widget(MDLabel(text=f"Time: {jadwal.get('waktu', '')} | Room: {jadwal.get('ruangan', '')} | Lecturer: {jadwal.get('dosen', '')}", hal
            jadwal_container.add_widget(card)
```

Pada file main.py terdapat fungsi sebagai penghubung antara tampilan (KV) dan database untuk fitur jadwal hari ini. Fungsi load_jadwal_hari_ini digunakan untuk menampilkan jadwal perkuliahan pengguna pada hari berjalan. Pertama, container jadwal dikosongkan agar tidak terjadi duplikasi data. Selanjutnya, aplikasi menentukan hari saat ini menggunakan modul datetime, lalu mengambil jadwal sesuai hari dan pengguna yang sedang login melalui DatabaseManager.

Jika tidak ada jadwal, aplikasi menampilkan pesan "No schedule today". Namun jika jadwal tersedia, setiap mata kuliah akan ditampilkan dalam bentuk kartu yang berisi nama mata kuliah, waktu, ruangan, dan dosen. Fungsi ini membantu pengguna melihat jadwal harian secara cepat dan terstruktur.

```python
class DatabaseManager:
    def get_jadwal_hari_ini(self, username: str, hari: str) -> List[Dict]:
        wb = load_workbook(self.filename)
        ws = wb["jadwal"]

        jadwal_list = []
        for row in ws.iter_rows(min_row=2, values_only=True):
            if row and row[0] == username and row[1].lower() == hari.lower():
                jadwal_list.append({
                    'hari': row[1],
                    'mata_kuliah': row[2],
                    'waktu': row[3],
                    'ruangan': row[4],
                    'dosen': row[5]
                })

        wb.close()
        return jadwal_list
```

Pada file database.py juga terdapat fungsi yang digunakan untuk mengambil data jadwal perkuliahan pengguna berdasarkan hari saat ini. Fungsi ini membaca data dari file Excel, melakukan penyaringan berdasarkan username dan hari, kemudian mengembalikan data dalam bentuk list dictionary untuk ditampilkan pada Home Screen aplikasi.

3.1.9.3 Tugas/Projek (Tasks/Projects)

```
MDBoxLayout:
    orientation: 'horizontal'
    size_hint_y: None
    height: "56dp"
    spacing: "10dp"

    MDIcon:
        icon: "clipboard-text"
        theme_text_color: "Primary"
        size_hint: None, None
        size: "40dp", "56dp"
        pos_hint: {"center_y": 0.5}

    MDLabel:
        text: "Tasks/Projects"
        theme_font_style: "H4"
        theme_text_color: "Primary"
        size_hint: 1, None
        height: "56dp"
        pos_hint: {"center_y": 0.5}
        text_size: 9999, None
        max_lines: 1

    MDIconButton:
        icon: "plus"
        size_hint: None, None
        size: "32dp", "32dp"
        md_bg_color: app.theme_cls.accent_color
        pos_hint: {"center_y": 0.5}
        on_release: root.add_tugas()
```

File myre.kv pada bagian ini berfungsi sebagai container horizontal yang menyusun ikon, teks judul, dan tombol tambah dalam satu baris. Selain itu juga terdapat tombol yang digunakan untuk menambahkan tugas atau projek baru, serta terdapat wadah utama untuk daftar tugas.

```
class HomeScreen(Screen):
    def load_tugas(self):
        tugas_container = self.ids.tugas_container
        tugas_container.clear_widgets()
        db = DatabaseManager()
        tugas_list = db.get_tugas_by_user(self.manager.current_user)

        for tugas in tugas_list:
            card = MDCard(
                orientation='vertical',
                padding="12dp",
                spacing="10dp",
                size_hint_y=None,
                height="120dp",
                elevation=0,
                md_bg_color=[0, 0, 0, 0]
            )

            title_box = MDBoxLayout(orientation='horizontal', adaptive_height=True, spacing="10dp")
            checkbox = MDCheckbox(
                size_hint=(None, None),
                size=("40dp", "40dp"),
                active=True if tugas.get('status') == 'completed' else False
            )

            checkbox.bind(active=lambda instance, value, t=tugas: self.toggle_tugas(t, value))

            title_label = MDLabel(text=tugas.get('nama_tugas', ''),
                                  theme_font_style="H6",
                                  halign="left", valign="center",
                                  size_hint_x=0.7,
                                  size_hint_y=None,
                                  height="40dp", text_size=(None, None))

            if tugas.get('status') == 'completed':
                title_label.theme_text_color = "Secondary"
```

```python
class HomeScreen(Screen):
    def load_tugas(self):
            edit_icon = MDIconButton(
                icon="pencil",
                size_hint=(None, None),
                size=("32dp", "32dp"),
                icon_size="16dp",
                on_release=lambda x, t=tugas: self.edit_tugas(t)
            )
            title_box.add_widget(checkbox)
            title_box.add_widget(title_label)
            title_box.add_widget(edit_icon)

            card.add_widget(title_box)

            mk_label = MDLabel(
                text=f"Subject: {tugas.get('mata_kuliah', '')}",
                halign="left",
                text_color=(1,1,1,1),
                size_hint_y=None,
                height="20dp",
                text_size=(None, None),
                shorten=True,
                shorten_from="right"
            )
            card.add_widget(mk_label)
```

```python
def load_tugas(self):
        deadline_raw = tugas.get('deadline', '')

        deadline_label = MDLabel(
            text=f"Deadline: {deadline_raw}",
            halign="left",
            text_color=(1,1,1,1),
            size_hint_y=None,
            height="20dp",
            text_size=(None, None),
            shorten=True,
            shorten_from="right"
        )
        card.add_widget(deadline_label)
        tugas_container.add_widget(card)

def toggle_tugas(self, tugas, value):
    db = DatabaseManager()
    new_status = 'completed' if value else 'pending'
    db.update_tugas_status(self.manager.current_user, tugas['nama_tugas'], new_status)
    self.load_tugas()

def add_tugas(self):
    self.selected_time = None
    self.selected_date = None

    content = MDBoxLayout(orientation="vertical", spacing="12dp", size_hint_y=None, height="250dp")
    nama_field = MDTextField(hint_text="Task/Project Name", mode="fill")
    mk_field = MDTextField(hint_text="Subject", mode="fill")
    date_field = MDTextField(hint_text="Deadline (DD-MM-YYYY)", mode="fill")
    time_field = MDTextField(hint_text="Time (HH:MM)", mode="fill")
```

```python
def add_tugas(self):
    content.add_widget(nama_field)
    content.add_widget(mk_field)
    content.add_widget(date_field)
    content.add_widget(time_field)

    def save_tugas(_):
        if not (nama_field.text and mk_field.text and date_field.text and time_field.text):
            self.dialog = MDDialog(
                title="Error",
                text="All fields must be filled.",
                buttons=[MDFlatButton(text="OK", on_release=lambda x: self.dialog.dismiss())]
            )
            self.dialog.open()
            return

        final_deadline = f"{date_field.text}\n{time_field.text} WIB"

        db = DatabaseManager()
        db.add_tugas(
            self.manager.current_user,
            nama_field.text,
            mk_field.text,
            final_deadline
        )

        self.dialog.dismiss()
        self.load_tugas()

    self.dialog = MDDialog(
        title="Add Task/Project",
        type="custom",
        content_cls=content,
        buttons=[
            MDFlatButton(text="CANCEL", on_release=lambda x: self.dialog.dismiss()),
            MDFlatButton(text="ADD", on_release=save_tugas),
        ],
    )
```

```python
def add_tugas(self):
    self.dialog.open()

def edit_tugas(self, tugas):
    self.selected_time = None
    self.selected_date = None

    content = MDBoxLayout(orientation="vertical", spacing="12dp", size_hint_y=None, height="250dp")
    nama_field = MDTextField(hint_text="Task/Project Name", mode="fill", text=tugas.get('nama_tugas', ''))
    mk_field = MDTextField(hint_text="Subject", mode="fill", text=tugas.get('mata_kuliah', ''))
    date_field = MDTextField(hint_text="Deadline (DD-MM-YYYY)", mode="fill")
    time_field = MDTextField(hint_text="Time (HH:MM)", mode="fill")

    deadline_str = tugas.get('deadline', '')
    if deadline_str:
        try:
            lines = deadline_str.split('\n')
            if len(lines) >= 2:
                date_part = lines[0]
                time_part = lines[1].replace('WIB', '')
                date_field.text = date_part
                time_field.text = time_part
        except:
            pass

    content.add_widget(nama_field)
    content.add_widget(mk_field)
    content.add_widget(date_field)
    content.add_widget(time_field)
```

```python
def edit_tugas(self, tugas):
    def save_tugas(_):
        if not (nama_field.text and mk_field.text and date_field.text and time_field.text):
            self.dialog = MDDialog(
                title="Error",
                text="All fields must be filled with the correct format.",
                buttons=[MDFlatButton(text="OK", on_release=lambda x: self.dialog.dismiss())]
            )
            self.dialog.open()
            return

        final_deadline = f"{date_field.text}\n{time_field.text} WIB"

        db = DatabaseManager()
        db.update_tugas(
            self.manager.current_user,
            tugas['nama_tugas'],
            nama_field.text,
            mk_field.text,
            final_deadline
        )

        self.dialog.dismiss()
        self.load_tugas()

    def delete_tugas(_):
        db = DatabaseManager()
        db.delete_tugas(self.manager.current_user, tugas['nama_tugas'])
        self.dialog.dismiss()
        self.load_tugas()
```

```python
def edit_tugas(self, tugas):
    self.dialog = MDDialog(
        title="Edit Task/Project",
        type="custom",
        content_cls=content,
        buttons=[
            MDFlatButton(text="DELETE", on_release=delete_tugas, text_color=(1, 0, 0, 1)),
            MDFlatButton(text="CANCEL", on_release=lambda x: self.dialog.dismiss()),
            MDFlatButton(text="SAVE", on_release=save_tugas),
        ],
    )
    self.dialog.open()
```

Pada file main.py terdapat beberapa fungsi untuk mengatur bagian tugas/projek di halaman Home. Fungsi load_tugas berfungsi untuk menampilkan daftar tugas atau proyek milik pengguna yang sedang login. Data tugas diambil dari database, kemudian ditampilkan dalam bentuk kartu yang berisi nama tugas, mata kuliah, deadline, serta checkbox untuk menandai status tugas. Jika tugas sudah selesai, checkbox akan aktif dan tampilan teks dibuat berbeda untuk membedakannya. Selain itu,

setiap tugas juga dilengkapi tombol edit untuk memperbarui atau menghapus data.

Fungsi toggle_tugas digunakan untuk mengubah status tugas antara pending dan completed ketika checkbox ditekan. Perubahan status ini langsung disimpan ke database, lalu daftar tugas diperbarui agar tampilan selalu sesuai dengan data terbaru. Fungsi add_tugas digunakan untuk menambahkan tugas baru. Aplikasi menampilkan dialog berisi form input nama tugas, mata kuliah, tanggal, dan waktu deadline. Jika ada data yang belum diisi, sistem akan menampilkan pesan kesalahan. Setelah semua data valid, tugas disimpan ke database dan daftar tugas diperbarui. Fungsi edit_tugas berfungsi untuk mengubah atau menghapus tugas yang sudah ada. Data tugas lama ditampilkan kembali pada form agar dapat diedit oleh pengguna. Selain menyimpan perubahan, fungsi ini juga menyediakan opsi untuk menghapus tugas dari database, kemudian memperbarui tampilan daftar tugas.

3.1.10 Jadwal Kuliah Screen (Class Schedule Screen)

```
<JadwalKuliahScreen>:
    BoxLayout:
        orientation: 'vertical'

        MDTopAppBar:
            title: "Class Schedule"
            elevation: 10
            md_bg_color: app.theme_cls.primary_color
            size_hint_y: None
            height: "32dp"

        ScrollView:
            do_scroll_x: False
            do_scroll_y: True
            canvas.before:
                Color:
                    rgba: 0.05, 0.05, 0.1, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

            MDBoxLayout:
                orientation: 'vertical'
                padding: "15dp"
                spacing: "25dp"
                size_hint_y: None
                height: self.minimum_height
```

```
<JadwalKuliahScreen>:
    BoxLayout:
        ScrollView:
            MDBoxLayout:
                MDBoxLayout:
                    orientation: 'horizontal'
                    adaptive_height: True
                    spacing: "15dp"

                    MDIcon:
                        icon: "school"
                        theme_text_color: "Primary"
                        size_hint: None, None
                        size: "40dp", "40dp"
                        pos_hint: {"center_y": 0.5}

                    MDLabel:
                        text: "Class Schedule"
                        theme_font_style: "H5"
                        theme_text_color: "Primary"
                        size_hint_x: 0.8
                        pos_hint: {"center_y": 0.5}

                    MDIconButton:
                        icon: "plus"
                        size_hint: None, None
                        size: "32dp", "32dp"
                        md_bg_color: app.theme_cls.accent_color
                        on_release: root.show_add_jadwal_dialog()
```

```
<JadwalKuliahScreen>:
    BoxLayout:
        ScrollView:
            MDBoxLayout:
                MDCard:
                    orientation: "vertical"
                    padding: "15dp"
                    radius: 20
                    elevation: 6
                    size_hint_y: None
                    size_hint_x: 1
                    height: jadwal_list_container.height + dp(80)
                    md_bg_color: [0, 0, 0, 0]

                    ScrollView:
                        bar_width: 0
                        do_scroll_x: False

                        MDList:
                            id: jadwal_list_container
                            size_hint_y: None
                            height: self.minimum_height

        NavBar:
```

Pada file myre,kv bagian jadwal kuliah screen (class schedule screen) terdapat BoxLayout yang digunakan sebagai kontainer utama dengan orientasi vertikal, sehingga seluruh komponen ditampilkan dari

atas ke bawah secara berurutan. Top App Bar berfungsi sebagai header aplikasi pada halaman jadwal kuliah. Judul Class Schedule menandakan bahwa halaman ini khusus untuk pengelolaan jadwal perkuliahan.

Selain itu terdapat bagian yang mengatur fungsi scroll dan latar belakang halaman. MDBoxLayout ini berfungsi sebagai pembungkus seluruh konten jadwal kuliah, termasuk judul, tombol tambah, dan daftar jadwal. MDCard digunakan sebagai wadah visual utama untuk menampilkan daftar jadwal kuliah agar terlihat rapi dan terpisah dari latar belakang.

```
class JadwalKuliahScreen(Screen):
    def on_enter(self, *args):
        self.load_jadwal()

    def load_jadwal(self):
        jadwal_container = self.ids.jadwal_list_container
        jadwal_container.clear_widgets()
        db = DatabaseManager()
        jadwal_list = db.get_jadwal_by_user(self.manager.current_user)
        for jadwal in jadwal_list:
            card = MDCard(orientation='vertical',
                          padding="8dp", size_hint_y=None, height="110dp", spacing="6dp", elevation=0, md_bg_color=[0, 0, 0, 0])

            title_box = MDBoxLayout(orientation='horizontal', adaptive_height=True, spacing="10dp")
            title_label = MDLabel(text=jadwal.get('mata_kuliah', ''), theme_font_style="H6", halign="left", size_hint_x=0.8)
            edit_icon = MDIconButton(
                icon="pencil",
                size_hint=(None, None),
                size=("32dp", "32dp"),
                icon_size="16dp",
                on_release=lambda x, j=jadwal: self.edit_jadwal(j)
            )
            title_box.add_widget(title_label)
            title_box.add_widget(edit_icon)

            card.add_widget(title_box)
            card.add_widget(MDLabel(text=f"Day: {jadwal.get('hari','')} | Time: {jadwal.get('waktu','')}", halign="left"))
            card.add_widget(MDLabel(text=f"Room: {jadwal.get('ruangan','')} | Lecturer: {jadwal.get('dosen','')}",
                                    theme_text_color="Secondary", halign="left"))
            jadwal_container.add_widget(card)
```

```
class JadwalKuliahScreen(Screen):
    def show_add_jadwal_dialog(self):
        content = MDBoxLayout(orientation="vertical", spacing="12dp", size_hint_y=None, height="320dp")
        hari_field = MDTextField(hint_text="Day (e.g., Monday)", mode="fill")
        mk_field = MDTextField(hint_text="Subject", mode="fill")
        waktu_field = MDTextField(hint_text="Time (e.g., 08:00-10:00)", mode="fill")
        ruangan_field = MDTextField(hint_text="Room", mode="fill")
        dosen_field = MDTextField(hint_text="Lecturer", mode="fill")
        content.add_widget(hari_field)
        content.add_widget(mk_field)
        content.add_widget(waktu_field)
        content.add_widget(ruangan_field)
        content.add_widget(dosen_field)

        def save_jadwal(_):
            if all([hari_field.text, mk_field.text, waktu_field.text]):
                db = DatabaseManager()
                db.add_jadwal(
                    self.manager.current_user,
                    hari_field.text,
                    mk_field.text,
                    waktu_field.text,
                    ruangan_field.text,
                    dosen_field.text
                )
                self.dialog.dismiss()
                self.load_jadwal()
                home_screen = self.manager.get_screen('home')
                home_screen.load_jadwal_hari_ini()
```

```python
class JadwalKuliahScreen(Screen):
    def show_add_jadwal_dialog(self):
        self.dialog = MDDialog(
            title="Add Class Schedule",
            type="custom",
            content_cls=content,
            buttons=[
                MDFlatButton(text="CANCEL", on_release=lambda x: self.dialog.dismiss()),
                MDFlatButton(text="ADD", on_release=save_jadwal),
            ],
        )
        self.dialog.open()

    def edit_jadwal(self, jadwal):
        content = MDBoxLayout(orientation="vertical", spacing="12dp", size_hint_y=None, height="320dp")
        hari_field = MDTextField(hint_text="Day (e.g., Monday)", mode="fill", text=jadwal.get('hari', ''))
        mk_field = MDTextField(hint_text="Subject", mode="fill", text=jadwal.get('mata_kuliah', ''))
        waktu_field = MDTextField(hint_text="Time (e.g., 08:00-10:00)", mode="fill", text=jadwal.get('waktu', ''))
        ruangan_field = MDTextField(hint_text="Room", mode="fill", text=jadwal.get('ruangan', ''))
        dosen_field = MDTextField(hint_text="Lecturer", mode="fill", text=jadwal.get('dosen', ''))
        content.add_widget(hari_field)
        content.add_widget(mk_field)
        content.add_widget(waktu_field)
        content.add_widget(ruangan_field)
        content.add_widget(dosen_field)

        def save_jadwal(_):
            if all([hari_field.text, mk_field.text, waktu_field.text]):
                db = DatabaseManager()
                db.update_jadwal(
                    self.manager.current_user,
                    jadwal['mata_kuliah'],
                    hari_field.text,
                    mk_field.text,
                    waktu_field.text,
                    ruangan_field.text,
                    dosen_field.text
                )
```

```python
class JadwalKuliahScreen(Screen):
    def edit_jadwal(self, jadwal):
        def save_jadwal(_):
                    self.dialog.dismiss()
                    self.load_jadwal()
                    home_screen = self.manager.get_screen('home')
                    home_screen.load_jadwal_hari_ini()

        def delete_jadwal(_):
            db = DatabaseManager()
            db.delete_jadwal(self.manager.current_user, jadwal['mata_kuliah'])
            self.dialog.dismiss()
            self.load_jadwal()
            home_screen = self.manager.get_screen('home')
            home_screen.load_jadwal_hari_ini()

        self.dialog = MDDialog(
            title="Edit Class Schedule",
            type="custom",
            content_cls=content,
            buttons=[
                MDFlatButton(text="DELETE", on_release=delete_jadwal, text_color=(1, 0, 0, 1)),
                MDFlatButton(text="CANCEL", on_release=lambda x: self.dialog.dismiss()),
                MDFlatButton(text="SAVE", on_release=save_jadwal),
            ],
        )
        self.dialog.open()
```

Pada file main.py class JadwalKuliahScreen merepresentasikan halaman Class Schedule pada aplikasi. Saat halaman dibuka (on_enter), aplikasi secara otomatis memanggil fungsi load_jadwal untuk menampilkan seluruh jadwal kuliah yang tersimpan di database sesuai dengan akun yang sedang login. Fungsi load_jadwal bertugas mengambil data jadwal dari database dan menampilkannya dalam bentuk kartu. Setiap kartu berisi informasi mata kuliah, hari, waktu, ruangan, dan dosen, serta dilengkapi tombol edit untuk mengubah atau menghapus jadwal tertentu. Tampilan akan selalu diperbarui setiap kali terjadi perubahan data.

Fungsi show_add_jadwal_dialog digunakan untuk menambahkan jadwal kuliah baru. Aplikasi menampilkan dialog berisi form input hari, mata kuliah, waktu, ruangan, dan dosen. Setelah data disimpan ke database, daftar jadwal pada halaman ini serta jadwal hari ini di halaman beranda akan langsung diperbarui. Fungsi edit_jadwal berfungsi untuk mengubah atau menghapus jadwal yang sudah ada. Data lama ditampilkan kembali dalam form agar dapat diedit oleh pengguna. Perubahan maupun penghapusan jadwal akan disimpan ke database dan langsung diperbarui pada tampilan jadwal kuliah dan halaman beranda.

```
class DatabaseManager:
    def add_jadwal(self, username: str, hari: str, mata_kuliah: str,
                   waktu: str, ruangan: str, dosen: str):
        wb = load_workbook(self.filename)
        ws = wb["jadwal"]

        ws.append([username, hari, mata_kuliah, waktu, ruangan, dosen])
        wb.save(self.filename)
        wb.close()

    def get_jadwal_by_user(self, username: str) -> List[Dict]:
        wb = load_workbook(self.filename)
        ws = wb["jadwal"]

        jadwal_list = []
        for row in ws.iter_rows(min_row=2, values_only=True):
            if row and row[0] == username:
                jadwal_list.append({
                    'hari': row[1],
                    'mata_kuliah': row[2],
                    'waktu': row[3],
                    'ruangan': row[4],
                    'dosen': row[5]
                })

        wb.close()
        return jadwal_list

    def update_jadwal(self, username: str, old_mata_kuliah: str,
                      new_hari: str, new_mata_kuliah: str,
                      new_waktu: str, new_ruangan: str, new_dosen: str):

        wb = load_workbook(self.filename)
        ws = wb["jadwal"]
```

```python
class DatabaseManager:
    def update_jadwal(self, username: str, old_mata_kuliah: str,
        for row in ws.iter_rows(min_row=2):
            if row[0].value == username and row[2].value == old_mata_kuliah:
                row[1].value = new_hari
                row[2].value = new_mata_kuliah
                row[3].value = new_waktu
                row[4].value = new_ruangan
                row[5].value = new_dosen
                break

        wb.save(self.filename)
        wb.close()

    def delete_jadwal(self, username: str, mata_kuliah: str):
        wb = load_workbook(self.filename)
        ws = wb["jadwal"]

        rows_to_delete = []
        for row_idx, row in enumerate(ws.iter_rows(min_row=2), start=2):
            if row[0].value == username and row[2].value == mata_kuliah:
                rows_to_delete.append(row_idx)

        for row_idx in reversed(rows_to_delete):
            ws.delete_rows(row_idx)

        wb.save(self.filename)
        wb.close()
```

Pada file database.py data jadwal disimpan pada sheet bernama jadwal, di mana setiap baris merepresentasikan satu jadwal kuliah milik pengguna. Fungsi add_jadwal digunakan untuk menambahkan data jadwal kuliah baru ke dalam database. Fungsi get_jadwal_by_user digunakan untuk mengambil seluruh jadwal kuliah milik pengguna tertentu. Data hanya diambil jika kolom username sesuai dengan pengguna yang sedang login.

Selain itu terdapat fungsi update_jadwal yang digunakan untuk memperbarui data jadwal kuliah yang sudah ada serta fungsi delete_jadwal untuk menghapus jadwal kuliah dari database.

3.1.11 Calendar Screen

```
<CalendarScreen>:
    BoxLayout:
        orientation: 'vertical'

        MDTopAppBar:
            title: "Calendar"
            elevation: 10
            md_bg_color: app.theme_cls.primary_color
            size_hint_y: None
            height: "32dp"

        MDBoxLayout:
            orientation: 'vertical'
            padding: "15dp"
            spacing: "15dp"
            adaptive_height: True
            canvas.before:
                Color:
                    rgba: 0.05, 0.05, 0.1, 1
                Rectangle:
                    pos: self.pos
                    size: self.size

            MDLabel:
                text: "Select Month:"
                halign: "center"
                theme_font_style: "H6"
                theme_text_color: "Primary"

            Spinner:
                id: month_dropdown
                values: ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]
                text: "January"
                size_hint: None, None
                size: "220dp", "50dp"
                pos_hint: {"center_x": 0.5}
                md_bg_color: rgba(0.15, 0.15, 0.25, 0.8)
                on_text: root.on_month_select(self, self.text)
```

```
<CalendarScreen>:
    BoxLayout:
        ScrollView:
            do_scroll_x: True
            do_scroll_y: True
            bar_width: "5dp"
            canvas.before:
                Color:
                    rgba: 0.05, 0.05, 0.1, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            GridLayout:
                id: calendar_grid
                cols: 7
                spacing: "5dp"
                padding: "10dp"
                size_hint_x: None
                width: dp(672)
                size_hint_y: None
                height: self.minimum_height

        NavBar:
```

Pada file myre.kv halaman kalender menggunakan BoxLayout dengan orientasi vertikal sebagai kontainer utama, sehingga setiap elemen ditampilkan dari atas ke bawah secara berurutan. Komponen MDTopAppBar digunakan sebagai header halaman yang menampilkan judul Calendar. Penggunaan md_bg_color memastikan warna mengikuti tema aplikasi, sedangkan elevation memberi efek bayangan agar tampilan lebih modern.

MDBoxLayout lain digunakan untuk kotak pemilihan bulan. Spinner digunakan sebagai dropdown bulan, berisi nama bulan dari Januari

hingga Desember dengan nilai awal diset dari January. Setiap kali bulan dipilih, fungsi on_month_select pada CalendarScreen akan dipanggil. Fungsi ini digunakan untuk mengubah bulan aktif dan memuat ulang data kalender sesuai bulan yang dipilih.

```python
class CalendarScreen(Screen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.current_month = datetime.now().month
        self.current_year = datetime.now().year
        self.dialog_open = False

    def on_enter(self, *args):
        self.load_month_dropdown()
        self.load_calendar()

    def load_month_dropdown(self):
        months = [
            "January", "February", "March", "April", "May", "June",
            "July", "August", "September", "October", "November", "December"
        ]
        dropdown = self.ids.month_dropdown
        dropdown.items = months
        dropdown.text = months[self.current_month - 1]

    def on_month_select(self, instance, value):
        months = [
            "January", "February", "March", "April", "May", "June",
            "July", "August", "September", "October", "November", "December"
        ]
        if value in months:
            self.current_month = months.index(value) + 1
            self.load_calendar()

    def load_calendar(self):
        calendar_grid = self.ids.calendar_grid
        calendar_grid.clear_widgets()
        db = DatabaseManager()
        activities = db.get_kalender_by_month(
            self.manager.current_user,
            self.current_month,
            self.current_year
        )
```

```python
class CalendarScreen(Screen):
    def load_calendar(self):
        cal = calendar.monthcalendar(self.current_year, self.current_month)
        days = ['M', 'T', 'W', 'T', 'F', 'S', 'S']
        for day in days:
            label = MDLabel(text=day, halign="center", theme_text_color="Secondary", size_hint=(None, None), size=("96dp", "20dp"))
            calendar_grid.add_widget(label)
        for week in cal:
            for day in week:
                if day == 0:
                    calendar_grid.add_widget(MDLabel(text=""))
                else:
                    date_str = f"{self.current_year}-{self.current_month:02d}-{day:02d}"
                    box = MDCard(
                        orientation="vertical",
                        size_hint=(None, None),
                        size=("96dp", "96dp"),
                        padding=("4dp", "4dp", "4dp", "4dp"),
                        radius=8,
                        elevation=2,
                        md_bg_color=[0, 0, 0, 0]
            )

                    date_label = MDLabel(text=str(day), halign="center", size_hint_y=None, height="24dp",
                                         theme_font_style="H6", padding=("0dp", "0dp", "0dp", "-3dp"))
                    box.add_widget(date_label)
                    scroll_view = KScrollView(size_hint=(None, None), size=("88dp", "56dp"), do_scroll_x=True, do_scroll_y=True, bar_width="5dp")
                    scroll_view.background_normal = ''
                    scroll_view.background_color = [0,0,0,0]
                    activities_box = KBoxLayout(orientation='vertical', size_hint=(None, None), size=("200dp", 0), spacing="2dp")

                    if date_str in activities:
                        num_activities = len(activities[date_str])
                        total_height = num_activities * dp(20) + (num_activities - 1) * dp(2) if num_activities > 0 else 0
                        activities_box.height = total_height
```

```python
class CalendarScreen(Screen):
    def load_calendar(self):
                    for act in activities[date_str]:
                        act_label = MDLabel(text=f"• {act['aktivitas']}", halign="left", theme_font_style="Caption", size_hint_y=None,
                                            height="20dp", size_hint_x=None, width="200dp")
                        activities_box.add_widget(act_label)
                scroll_view.add_widget(activities_box)
                box.add_widget(scroll_view)
                plus_layer = AnchorLayout(
                    anchor_x="right",
                    anchor_y="top",
                    size_hint_y=None,
                    height="16dp",
                    padding=("0dp", "-10dp", "-12dp", "0dp"),
                )
                add_btn = MDIconButton(icon="plus", icon_size="14dp", on_release=lambda x, d=date_str: self.show_add_activity_dialog(d))
                plus_layer.add_widget(add_btn)
                box.add_widget(plus_layer)
                calendar_grid.add_widget(box)

    def refresh_activities(self):
        self.activities_box.clear_widgets()
        db = DatabaseManager()
        activities = db.get_kalender_by_month(self.manager.current_user, self.current_month, self.current_year)
        if self.tanggal in activities and activities[self.tanggal]:
            for act in activities[self.tanggal]:
                card = MDCard(orientation='horizontal', padding="6dp", size_hint_y=None, height="48dp", radius=[8], elevation=0, md_bg_color=[0, 0, 0, 0])
                act_label = MDLabel(text=f"{act['aktivitas']} {act['waktu'] or ''}", halign="left", size_hint_x=0.8, theme_text_color="Primary")
                delete_btn = MDIconButton(icon="delete", size_hint=(None, None), size=("32dp", "32dp"),
                                          icon_size="20dp", on_release=lambda x, t=self.tanggal, a=act['aktivitas']: self.delete_activity(t, a))
                card.add_widget(act_label)
                card.add_widget(delete_btn)
                self.activities_box.add_widget(card)
        else:
            no_act_label = MDLabel(text="No activities", halign="center", theme_text_color="Secondary")
            self.activities_box.add_widget(no_act_label)
```

```python
class CalendarScreen(Screen):
    def show_add_activity_dialog(self, tanggal):
        try:
            if self.dialog_open:
                return

            self.tanggal = tanggal

            content = MDBoxLayout(orientation="vertical", spacing="12dp", size_hint_y=None, height="450dp")

            aktivitas_field = MDTextField(hint_text="Activity", mode="fill")
            waktu_field = MDTextField(hint_text="Time (optional)", mode="fill")
            content.add_widget(aktivitas_field)
            content.add_widget(waktu_field)

            def add_activity_inline(_):
                if not aktivitas_field.text:
                    error_dialog = MDDialog(
                        title="Error",
                        text="Activity cannot be empty.",
                        buttons=[MDFlatButton(text="OK", on_release=lambda x: error_dialog.dismiss())],
                    )
                    error_dialog.open()
                    return

                try:
                    db = DatabaseManager()
                    db.add_kalender_aktivitas(self.manager.current_user, tanggal, aktivitas_field.text, waktu_field.text)
                    aktivitas_field.text = ""
                    waktu_field.text = ""
                    self.refresh_activities()
                    self.load_calendar()
                except Exception as e:
                    error_dialog = MDDialog(
                        title="Error",
                        text=f"Failed to add activity: {str(e)}",
                        buttons=[MDFlatButton(text="OK", on_release=lambda x: error_dialog.dismiss())],
                    )
```

```python
class CalendarScreen(Screen):
    def show_add_activity_dialog(self, tanggal):
                error_dialog.open()

            content.add_widget(MDFlatButton(text="ADD", on_release=add_activity_inline))

            db = DatabaseManager()
            activities = db.get_kalender_by_month(self.manager.current_user, self.current_month, self.current_year)
            content.add_widget(MDLabel(text="Existing activities:", theme_font_style="H6"))
            activities_scroll = KScrollView(size_hint=(1, None), height="250dp")
            self.activities_box = MDBoxLayout(orientation='vertical', spacing="5dp", adaptive_height=True)
            if tanggal in activities and activities[tanggal]:
                for act in activities[tanggal]:
                    card = MDCard(orientation='horizontal', padding="6dp", size_hint_y=None, height="48dp", radius=[8],
                                  elevation=0, md_bg_color=[0, 0, 0, 0])
                    act_label = MDLabel(text=f"{act['aktivitas']} {act['waktu'] or ''}", halign="left", size_hint_x=0.8, theme_text_color="Primary")
                    delete_btn = MDIconButton(icon="delete", size_hint=(None, None), size=("32dp", "32dp"), icon_size="20dp",
                                              on_release=lambda x, t=tanggal, a=act['aktivitas']: self.delete_activity(t, a))
                    card.add_widget(act_label)
                    card.add_widget(delete_btn)
                    self.activities_box.add_widget(card)
            else:
                no_act_label = MDLabel(text="No activities", halign="center", theme_text_color="Secondary")
                self.activities_box.add_widget(no_act_label)
            activities_scroll.add_widget(self.activities_box)
            content.add_widget(activities_scroll)

            def dismiss_dialog(_):
                self.dialog.dismiss()
                self.dialog_open = False

            def save_and_close(_):
                dismiss_dialog(None)
                self.load_calendar()
```

```python
class CalendarScreen(Screen):
    def show_add_activity_dialog(self, tanggal):
            self.dialog = MDDialog(
                title=f"Activities for {tanggal}",
                type="custom",
                content_cls=content,
                buttons=[
                    MDFlatButton(text="CANCEL", on_release=dismiss_dialog),
                    MDFlatButton(text="SAVE", on_release=save_and_close),
                ],
            )
            self.dialog.open()
            self.dialog_open = True
        except Exception as e:
            print(f"Error in show_add_activity_dialog: {e}")
            error_dialog = MDDialog(
                title="Error",
                text=f"An error occurred: {str(e)}",
                buttons=[MDFlatButton(text="OK", on_release=lambda x: error_dialog.dismiss())],
            )
            error_dialog.open()

    def delete_activity(self, tanggal, aktivitas):
        try:
            db = DatabaseManager()
            db.delete_kalender_aktivitas(self.manager.current_user, tanggal, aktivitas)
            self.refresh_activities()
            self.load_calendar()
        except Exception as e:
            print(f"Error deleting activity: {e}")
            error_dialog = MDDialog(
                title="Error",
                text=f"Failed to delete activity: {str(e)}",
                buttons=[MDFlatButton(text="OK", on_release=lambda x: error_dialog.dismiss())],
            )
            error_dialog.open()
```

Pada file main.py terdapat CalendarScreen yang merupakan kelas untuk menampilkan kalender bulanan. Di dalamnya terdapat beberapa fungsi. Fungsi __init__ untuk menyimpan bulan dan tahun saat ini dan mengambil nilai awal dari tanggal saat ini. Fungsi on_enter dipanggil otomatis ketika halaman kalender dibuka. Fungsi ini memuat dropdown bulan dan isi kalender sesuai bulan aktif. Fungsi load_month_dropdown digunakan untuk menyediakan daftar nama bulan dari Januari sampai Desember, mengisi pilihan pada spinner bulan dan menyesuaikan dengan bulan aktif agar sinkron dengan kalender.

Selain itu, terdapat fungsi on_month_select yang akan dipanggil saat pengguna memilih bulan, mengubah nilai current_month, dan memuat kalender sesuai dengan bulan yang dipilih oleh user. Segala struktur kalender didefinisikan pada fungsi load_calendar. Fungsi refresh_activities digunakan untuk memperbarui isi dialog aktivitas tanpa menutup dialog. Fungsi lain yang ada, yakni fungsi show_add_activity_dialog dan delete_activity untuk menambahkan dan menghapus aktivitas.

```python
class DatabaseManager:
    def add_kalender_aktivitas(self, username: str, tanggal: str, aktivitas: str, waktu: str = ''):
        wb = load_workbook(self.filename)
        ws = wb["kalender"]

        ws.append([username, tanggal, aktivitas, waktu])
        wb.save(self.filename)
        wb.close()

    def get_kalender_by_month(self, username: str, bulan: int, tahun: int = None) -> Dict[str, List]:
        if tahun is None:
            tahun = datetime.now().year

        wb = load_workbook(self.filename)
        ws = wb["kalender"]

        aktivitas_per_tanggal = {}

        for row in ws.iter_rows(min_row=2, values_only=True):
            if row and row[0] == username:
                try:
                    tanggal_obj = datetime.strptime(str(row[1]), '%Y-%m-%d')
                    if tanggal_obj.month == bulan and tanggal_obj.year == tahun:
                        tanggal_str = tanggal_obj.strftime('%Y-%m-%d')
                        if tanggal_str not in aktivitas_per_tanggal:
                            aktivitas_per_tanggal[tanggal_str] = []

                        aktivitas_per_tanggal[tanggal_str].append({
                            'aktivitas': row[2],
                            'waktu': row[3] or ''
                        })
                except:
                    continue

        wb.close()
        return aktivitas_per_tanggal
```

```python
class DatabaseManager:
    def update_kalender_aktivitas(self, username: str, tanggal: str, old_aktivitas: str, new_aktivitas: str, waktu: str = ''):
        wb = load_workbook(self.filename)
        ws = wb["kalender"]

        for row in ws.iter_rows(min_row=2):
            if (row[0].value == username and row[1].value == tanggal and row[2].value == old_aktivitas):
                row[2].value = new_aktivitas
                row[3].value = waktu
                break

        wb.save(self.filename)
        wb.close()

    def delete_kalender_aktivitas(self, username: str, tanggal: str, aktivitas: str):
        wb = load_workbook(self.filename)
        ws = wb["kalender"]

        rows_to_delete = []
        for row_idx, row in enumerate(ws.iter_rows(min_row=2), start=2):
            if (row[0].value == username and row[1].value == tanggal and row[2].value == aktivitas):
                rows_to_delete.append(row_idx)

        for row_idx in reversed(rows_to_delete):
            ws.delete_rows(row_idx)

        wb.save(self.filename)
        wb.close()
```

Pada file database.py terdapat beberapa fungsi yang digunakan untuk menyimpan, mengambil, memperbarui, dan menghapus aktivitas

kalender pengguna. Data kalender disimpan dalam sheet kalender pada file Excel dan dikelola menggunakan library openpyxl. Fungsi add_kalender_aktivitas digunakan untuk menambahkan aktivitas baru ke kalender pengguna. Sistem akan membuka file database excel dan mengakses sheet kalender, serta menambahkan baris yang berisi data baru. Fungsi get_kalender_by_month digunakan untuk mengambil seluruh aktivitas kalender pengguna berdasarkan bulan tertentu. Selain itu, terdapat fungsi update_kalender_aktivitas untuk mengedit aktivitas kalender yang sudah ada serta fungsi delete_kalender_aktivitas untuk menghapus aktivitas kalender tertentu.

3.1.12 Profile Screen

```
<ProfileScreen>:
    BoxLayout:
        orientation: 'vertical'

        MDTopAppBar:
            title: "Profile"
            elevation: 10
            md_bg_color: app.theme_cls.primary_color
            size_hint_y: None
            height: "32dp"

        ScrollView:
            canvas.before:
                Color:
                    rgba: 0.05, 0.05, 0.1, 1
                Rectangle:
                    pos: self.pos
                    size: self.size
            MDBoxLayout:
                orientation: 'vertical'
                padding: "30dp"
                spacing: "25dp"
                adaptive_height: True

                MDTextField:
                    id: nama_field
                    hint_text: "Full Name"
                    leading_icon: "account"
                    mode: "fill"
                    radius: [10]
                    fill_color: rgba(0.2, 0.2, 0.3, 0.8)
```

```
            MDTextField:
                id: nim_field
                hint_text: "NIM"
                leading_icon: "card-account-details"
                mode: "fill"
                radius: [10]
                fill_color: rgba(0.2, 0.2, 0.3, 0.8)

            MDTextField:
                id: ipk_field
                hint_text: "IPK"
                leading_icon: "school"
                mode: "fill"
                radius: [10]
                fill_color: rgba(0.2, 0.2, 0.3, 0.8)

            MDRaisedButton:
                text: "SAVE PROFILE"
                size_hint_x: 0.8
                pos_hint: {"center_x": 0.5}
                md_bg_color: app.theme_cls.primary_color
                on_release: root.save_profile()

            MDRaisedButton:
                text: "LOGOUT"
                size_hint_x: 0.8
                pos_hint: {"center_x": 0.5}
                md_bg_color: app.theme_cls.error_color
                on_release: root.logout()

    NavBar:
```

Pada file myre.kv, Profile Screen menggunakan BoxLayout vertikal sebagai kontainer utama. MDTopAppBar digunakan untuk menampilkan judul halaman di bagian atas layar. Terdapat scroll view yang memungkinkan halaman tetap bisa digulir jika konten melebihi tinggi layar. MDBoxLayout digunakan sebagai wadah semua input pada Profile Screen, sedangkan MDTextField digunakan untuk menampilkan data yang diinput pengguna. Selain itu, terdapat tombol untuk simpan profil dan tombol logout untuk keluar dari sesi pengguna.

```
class ProfileScreen(Screen):
    def on_enter(self, *args):
        self.load_profile()

    def load_profile(self):
        db = DatabaseManager()
        profile = db.get_profile(self.manager.current_user)
        if profile:
            self.ids.nama_field.text = profile.get('nama', '')
            self.ids.nim_field.text = profile.get('nim', '')
            self.ids.ipk_field.text = profile.get('ipk', '')

    def save_profile(self):
        nama = self.ids.nama_field.text
        nim = self.ids.nim_field.text
        ipk = self.ids.ipk_field.text
        db = DatabaseManager()
        db.update_profile(self.manager.current_user, nama, nim, ipk)
        self.show_success_dialog("Success", "Profile saved successfully!")

    def show_success_dialog(self, title, text):
        self.dialog = MDDialog(title=title, text=text, buttons=[MDFlatButton(text="OK", on_release=lambda x: self.dialog.dismiss())])
        self.dialog.open()

    def logout(self):
        self.manager.current_user = ""
        self.manager.current = 'login'
```

Pada file main.py terdapat Class ProfileScreen yang terdiri dari beberapa fungsi. Fungsi on_enter merupakan fungsi yang dipanggil

secara otomatis setiap kali halaman Profile dibuka dan digunakan untuk memastikan bahwa data profil yang ditampilkan selalu merupakan data terbaru dari database. Fungsi load_profile untuk mengambil data profil pengguna dari database dan fungsi save_profile untuk mengambil data input terbaru dari pengguna pada halaman Profile. Selain itu terdapat fungsi show_success_dialog untuk menampilkan dialog informasi serta terdapat fungsi logout untuk mengakhiri sesi pengguna.

3.1.13 Class MyreApp

```
class MyreApp(MDApp):
    def build(self):
        self.theme_cls.primary_palette = "DeepPurple"
        self.theme_cls.theme_style = "Dark"
        self.theme_cls.accent_palette = "Amber"
        Builder.load_file('myre.kv')
        sm = MyreScreenManager()
        sm.add_widget(WelcomeScreen(name='welcome'))
        sm.add_widget(LoginScreen(name='login'))
        sm.add_widget(RegisterScreen(name='register'))
        sm.add_widget(HomeScreen(name='home'))
        sm.add_widget(JadwalKuliahScreen(name='jadwal'))
        sm.add_widget(CalendarScreen(name='calendar'))
        sm.add_widget(ProfileScreen(name='profile'))
        return sm
```

Pada file main.py, terdapat kelas MyreApp yang merupakan turunan dari MDApp, yaitu kelas utama pada KivyMD yang digunakan untuk membangun aplikasi berbasis Material Design. Seluruh konfigurasi awal aplikasi didefinisikan di dalam kelas ini. Fungsi build adalah fungsi utama yang dijalankan pertama kali saat aplikasi dibuka. Fungsi ini bertugas untuk mengatur tema aplikasi, memuat file .kv, dan menginisialisasi serta mengatur screen manager. Pada fungsi ini juga terdapat daftar seluruh halaman aplikasi yang telah diberi nama di setiap halamannya sebagai identitas saat berpindah.

3.1.14 Blok Eksekusi

```
if __name__ == '__main__':
    db = DatabaseManager()
    MyreApp().run()
```

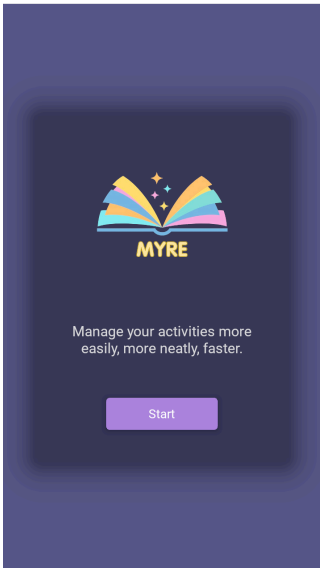
Pada bagian akhir file main.py, terdapat blok eksekusi yang memastikan bahwa kode hanya dijalankan ketika file main.py dijalankan secara langsung, bukan saat diimpor sebagai modul. Terdapat inisialisasi database agar file database dibuat atau dipersiapkan

sebelum aplikasi dijalankan serta adanya bagian MyreApp().run() untuk menjalankan aplikasi MYRE dan menampilkan antarmuka kepada pengguna.

3.2 Manual Penggunaan & Screenshot Aplikasi

- Langkah 1 : Halaman Awal Aplikasi

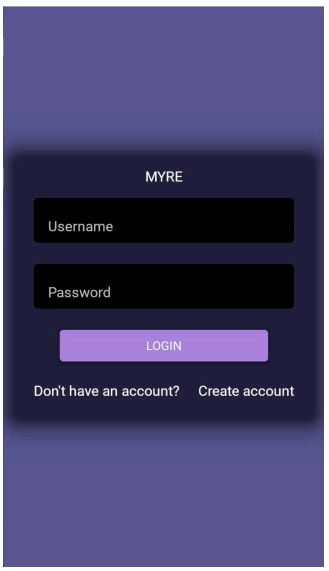
  Pada saat aplikasi dijalankan, pengguna akan melihat halaman utama seperti pada gambar 1. Pengguna dapat menekan tombol "Start" untuk lanjut ke halaman berikutnya.



Gambar 1. Tampilan awal

- Langkah 2 : Halaman Login

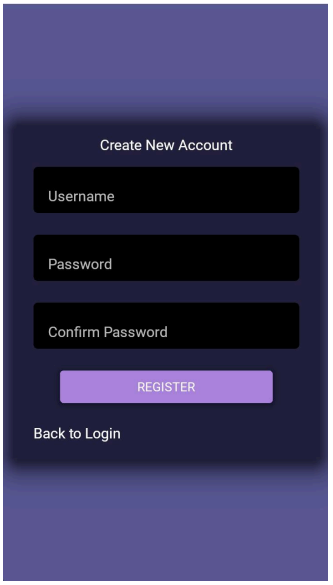
  Pengguna memasuki halaman Login. Apabila sudah pernah melakukan register sebelumnya, pengguna dapat login dengan username dan password yang pernah digunakan sebelumnya. Namun apabila pengguna belum pernah mengunjungi platform ini, pengguna dapat menekan tulisan "Create account" yang akan ditujukan pada halaman register.

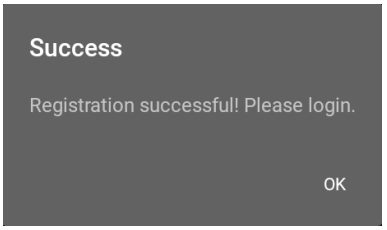


Gambar 2 : Halaman Login

- Langkah 3 : Masuk pada halaman Register

  Pengguna yang belum pernah login sebelumnya akan ditujukan pada halaman "Register". Pengguna dapat mengisikan username dan password serta konfirmasi password kembali. Setelah melakukan register, pengguna akan mendapatkan informasi bahwa pengguna berhasil melakukan register. pengguna dapat menekan "OK" lalu akan ditujukan pada halaman Login untuk melakukan login.

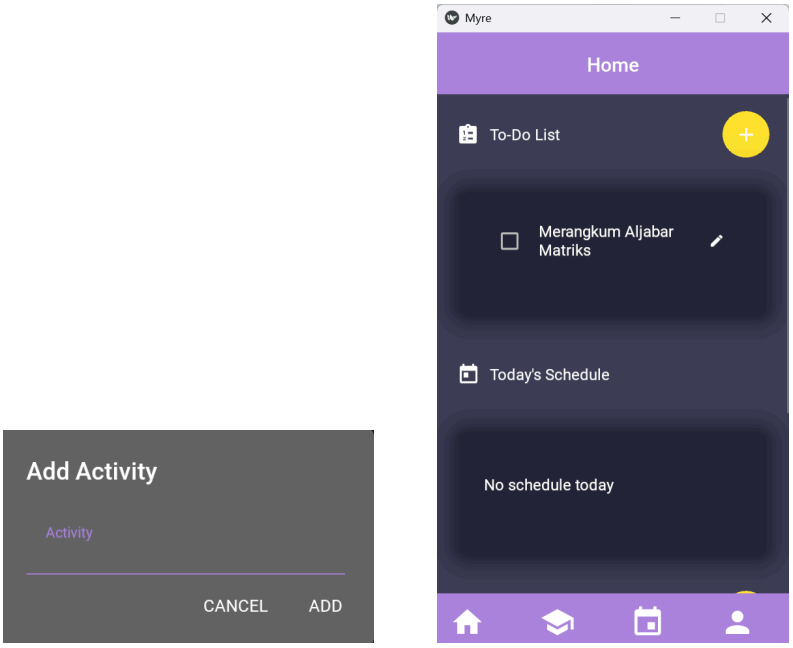




Gambar 3 & 4 : Tampilan Registrasi dan ketika registrasi berhasil

- Langkah 4 : Halaman Home pada bagian To-Do List

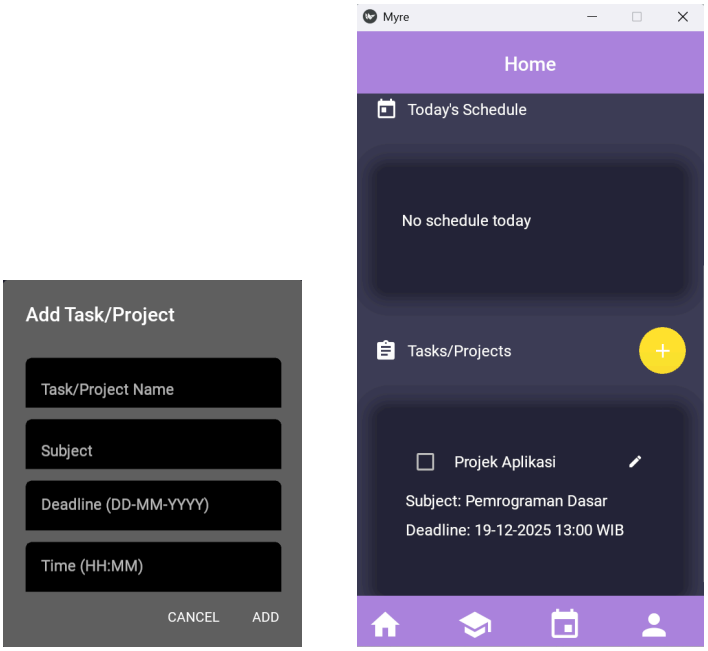
Pada halaman Home, pengguna dapat mengisi To-Do List dengan menekan tombol "+" disamping kanan. Lalu akan muncul data yang harus diisikan pengguna. Pengguna menekan tombol "ADD" agar To-Do List muncul di halaman Home. Pengguna dapat mencentang kotak disamping kiri Activity apabila sudah selesai mengerjakan To-Do List, dan dapat menekan icon pensil untuk mengedit atau menghapus.



Gambar 5 & 6. Tampilan pada Add Activity & halaman Home setelah To-Do List ditambahkan

- Langkah 5 : Halaman Home bagian Today's Schedule dan Task/Projects

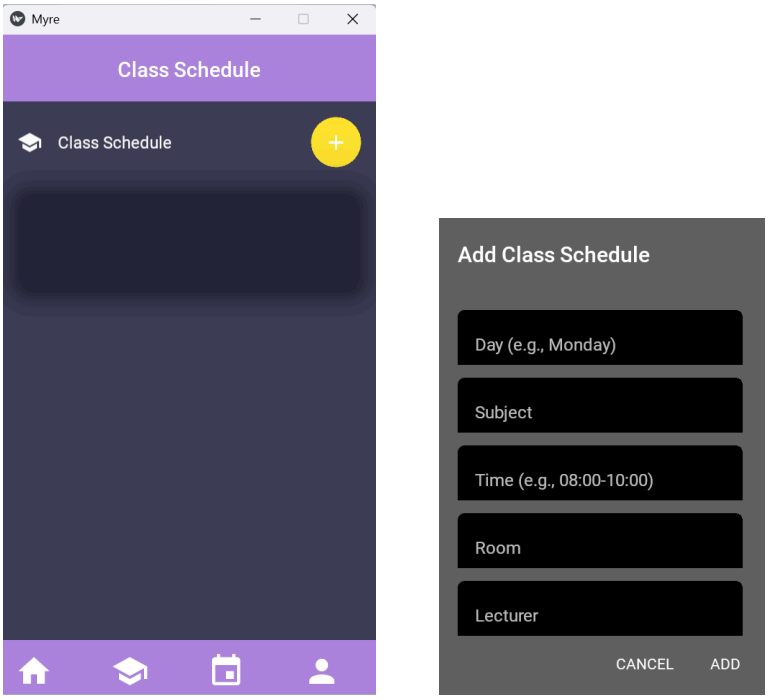
  Pengguna dapat melihat Jadwal Hari ini jika sudah mengisi data Jadwal pada halaman Schedule. Untuk mengisi Task/Projects, pengguna dapat menekan tombol "+" dan akan diarahkan untuk mengisi data. Pengguna menekan tombol "ADD" agar Task/Projects muncul di halaman Home. Pengguna dapat mencentang kotak disamping kiri Task/Projects apabila sudah selesai mengerjakan Task/Projects, dan dapat menekan icon pensil untuk mengedit atau menghapus.



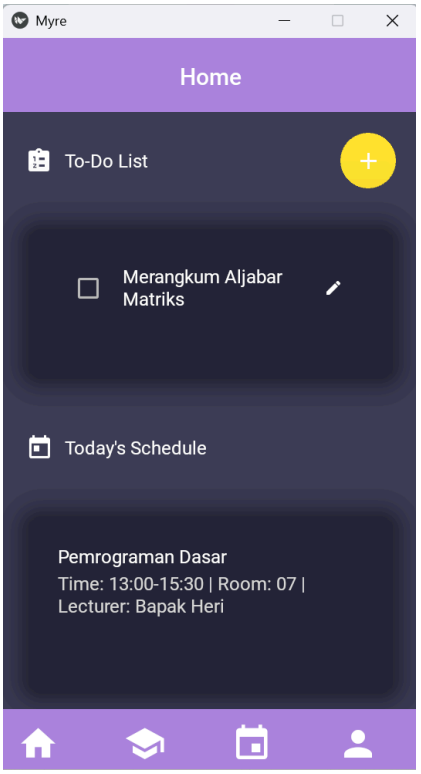
Gambar 7 & 8. Tampilan Add Task/Projects & Home apabila Task/Projects ditambahkan

- Langkah 6 : Halaman Schedule

  Pada Halaman Schedule, pengguna dapat menambahkan jadwal dengan menekan tombol "+" berwarna kuning, maka akan muncul data yang harus diisi. Pengguna menekan tombol "ADD" agar Schedule muncul di halaman Home dan halaman Schedule.



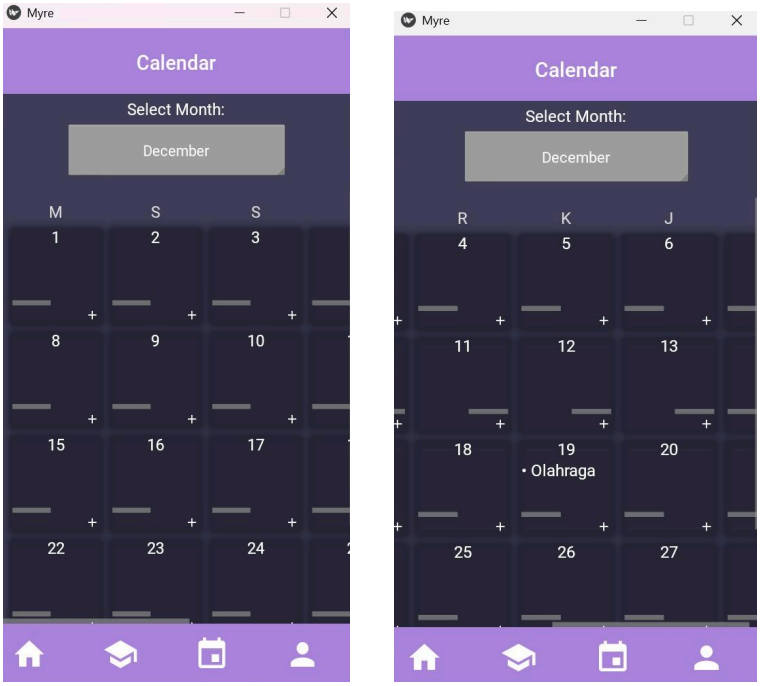
Gambar 9 & 10 Tampilan Schedule awal dan tampilan Add Class Schedule



Gambar 11. Tampilan apabila Jadwal muncul di halaman Home

- Langkah 7 : Halaman Calendar

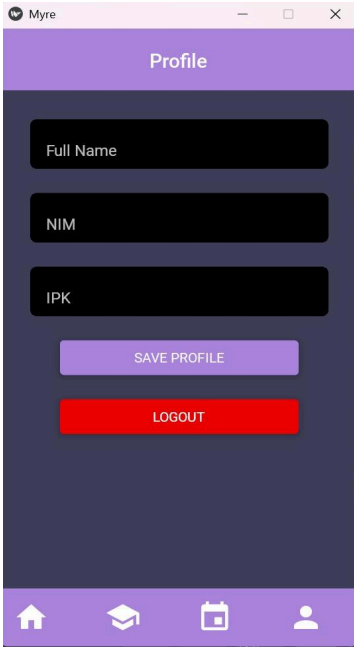
  Pada halaman Calendar, pengguna dapat menambahkan events yang diadakan pada tanggal tersebut dengan menekan tombol '+' di kanan bawah setiap tanggal.



Gambar 12 & 13. Tampilan Halaman Calender awal & apabila events ditambahkan

- Langkah 8 : Tampilan Profile

  Pada halaman Profile, pengguna dapat memasukkan Nama, NIM, dan IPK. Setelah itu, tekan tombol "SAVE PROFILE" maka Nama, NIM, dan IP pengguna akan ditampilkan. Apabila pengguna ingin keluar dari account, pengguna dapat menekan tombol "LOGOUT".



Gambar 14. Tampilan pada halaman Profile

LAMPIRAN

Berikut adalah lampiran link GitHub yang berisikan kode program aplikasi "MYRE" : https://github.com/Juliaaa1234/Kode-Program-Aplikasi-MYRE

DAFTAR PUSTAKA

Catriwati, S. (2022). APLIKASI PENGINGAT JADWAL DAN TUGAS KULIAH BERBASIS ANDROID: AMIK Tri Dharma Pekanbaru dan AMIK Mahaputra Riau. *Journal Intra Tech, 6*(1), 01–07. https://doi.org/10.37030/jit.v6i1.106

Zahwaa, S. A., Aulia, T. N., Afridan, A., Hadi, D. S., & Gunawan, I. (2025). Analisis Dan Perancangan Aplikasi Pengingat Jadwal Kuliah Mahasiswa STIKOM Tunas Bangsa. *Jurnal Inovasi Artificial Intelligence & Komputasional Nusantara*, *3*(1), 17–23. https://doi.org/10.260396/r3jwnt59