

SAVINGS REMINDER
LAPORAN PROJECT SEMESTER GANJIL



Disusun Oleh:

1. Revaldo Meisya Nadin Rifai (25031554064)
2. Pandu Sakti Weldiansah (25031554191)
3. Tarra Afifah Zahra Ghaida (25031554270)

Dosen Pengampu:

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

Mata Kuliah:

Pemrograman Dasar

PROGRAM STUDI SAINS DATA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SURABAYA
2025

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1.....	3
PENDAHULUAN	3
1.1. LATAR BELAKANG.....	3
1.2. RUMUSAN MASALAH.....	4
1.3. TUJUAN	4
BAB 2.....	5
ANALISIS DAN PERANCANGAN	5
2.1. ANALISIS KEBUTUHAN APLIKASI	5
2.2. DIAGRAM ALUR (FLOWCHART)	6
2.3. SKETSA DESAIN ANTARMUKA PENGGUNA	7
BAB 3.....	10
IMPLEMENTASI.....	10
3.1. PENJELASAN KODE	10
3.2. SCREENSHOT APLIKASI.....	45
BAB 4.....	51
LAMPIRAN	51
4.1. LAMPIRAN FILE.....	51
DAFTAR PUSTAKA.....	52

BAB 1

PENDAHULUAN

1.1. LATAR BELAKANG

Aplikasi *Savings Reminder* merupakan aplikasi yang dirancang untuk membantu pengguna dalam merencanakan dan mencapai target tabungan secara lebih teratur. Aplikasi ini memungkinkan pengguna untuk menentukan target tabungan, memilih periode menabung (harian, mingguan, atau bulanan), serta menghitung estimasi waktu yang dibutuhkan untuk mencapai target tersebut. Selain itu, aplikasi ini dilengkapi dengan fitur pengingat otomatis berupa notifikasi agar pengguna tidak lupa menabung sesuai jadwal yang telah ditentukan.

Di era saat ini, pengelolaan keuangan pribadi menjadi hal yang sangat penting terutama dalam menghadapi kebutuhan hidup yang semakin meningkat. Menabung menjadi salah satu cara yang efektif untuk mempersiapkan kebutuhan di masa depan, baik untuk pendidikan, kebutuhan darurat, maupun tujuan finansial lainnya. Namun, pada kenyataannya masih banyak orang yang mengalami kesulitan dalam menerapkan kebiasaan menabung secara rutin dan disiplin.

Permasalahan utama yang sering dihadapi adalah kurangnya perencanaan tabungan yang jelas, seperti tidak adanya target yang terukur, ketidakpastian nominal yang harus disisihkan, serta ketidaktahuan mengenai jangka waktu pencapaian target tabungan. Selain itu, faktor kesibukan dan kurangnya pengingat juga menyebabkan seseorang sering lupa atau menunda kegiatan menabung. Hal ini mengakibatkan rencana keuangan yang telah dibuat tidak berjalan sesuai harapan dan sulit untuk dievaluasi.

Dengan adanya aplikasi *Savings Reminder*, diharapkan pengguna dapat lebih mudah mengatur tabungan, mengetahui perkiraan waktu pencapaian target, serta terbantu oleh sistem pengingat yang mendorong kebiasaan menabung secara konsisten.

1.2. RUMUSAN MASALAH

Dalam kehidupan sehari-hari, kegiatan menabung masih sering menghadapi berbagai kendala. Permasalahan-permasalahan tersebut menjadi alasan utama perlunya pengembangan aplikasi *Savings Reminder*, di antaranya:

- 1.2.1. Banyak orang yang mengalami kesulitan dalam memperkirakan berapa lama waktu yang dibutuhkan untuk mencapai target tabungan apabila menabung secara rutin.
- 1.2.2. Kurangnya media atau aplikasi sederhana yang dapat membantu menghitung estimasi waktu pencapaian target tabungan secara otomatis dan mudah dipahami.
- 1.2.3. Tidak adanya sistem pengingat yang efektif menyebabkan pengguna sering lupa atau menunda kegiatan menabung, sehingga kebiasaan menabung menjadi tidak konsisten.

Berdasarkan permasalahan tersebut, diperlukan sebuah aplikasi yang mampu membantu pengguna dalam merencanakan tabungan secara terstruktur, menghitung estimasi pencapaian target, serta memberikan pengingat agar pengguna dapat menabung secara konsisten.

1.3. TUJUAN

Tujuan dari pembuatan aplikasi *Savings Reminder* adalah:

- 1.3.1. Menyediakan aplikasi yang dapat membantu pengguna menghitung estimasi waktu untuk mencapai target tabungan.
- 1.3.2. Membantu pengguna merencanakan tabungan berdasarkan periode tertentu (harian, mingguan, atau bulanan).
- 1.3.3. Menyediakan fitur pengingat otomatis agar pengguna lebih disiplin dan konsisten dalam menabung

BAB 2

ANALISIS DAN PERANCANGAN

2.1. ANALISIS KEBUTUHAN APLIKASI

Untuk mengembangkan aplikasi *Savings Reminder*, dilakukan analisis kebutuhan aplikasi untuk memahami alasan mengapa aplikasi ini diperlukan serta fitur apa saja yang dibutuhkan oleh pengguna agar aplikasi yang dikembangkan dapat membantu pengguna dalam merencanakan tabungan secara efektif, konsisten, dan mudah digunakan. Analisis ini dibagi menjadi dua bagian utama:

2.3.1. Kebutuhan Fungsional

Kebutuhan fungsional menjelaskan fungsi utama yang harus dimiliki oleh aplikasi *Savings Reminder* agar dapat berjalan sesuai dengan tujuan pengembangannya. Kebutuhan fungsional tersebut antara lain:

- Aplikasi dapat menerima input target tabungan yang ingin dicapai oleh pengguna.
- Aplikasi memungkinkan pengguna menentukan nominal tabungan secara rutin, baik harian, mingguan, maupun bulanan.
- Sistem dapat menghitung estimasi waktu yang dibutuhkan untuk mencapai target tabungan berdasarkan nominal dan periode yang dipilih.
- Aplikasi dapat menampilkan hasil perhitungan estimasi waktu dan informasi progres tabungan pengguna.
- Aplikasi menyediakan fitur pengingat otomatis berupa notifikasi sesuai jadwal menabung yang telah ditentukan pengguna.

2.3.1. Kebutuhan Non-Fungsional

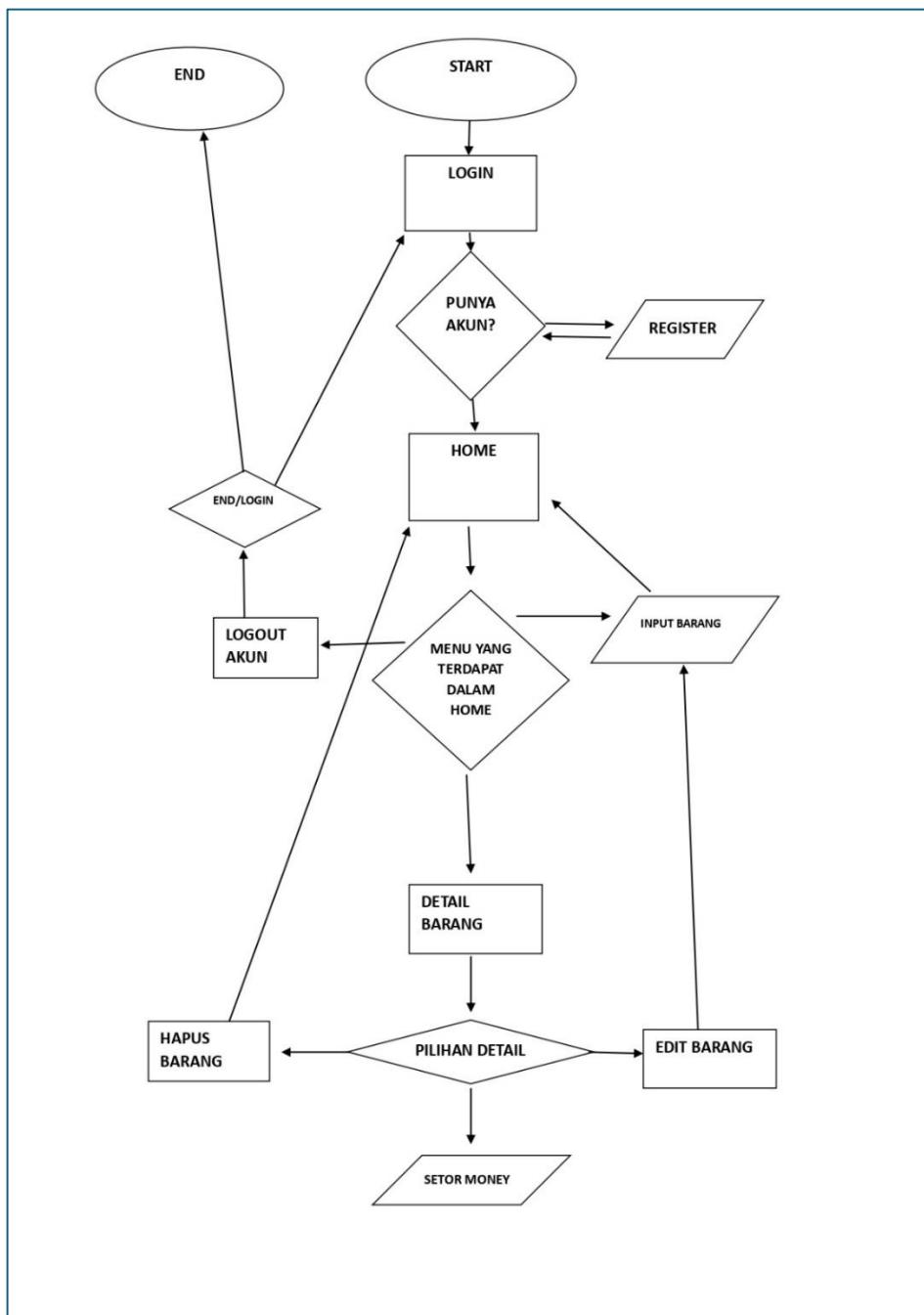
Kebutuhan non-fungsional berkaitan dengan kualitas dan kinerja aplikasi *Savings Reminder*. Kebutuhan ini meliputi:

- Aplikasi memiliki tampilan antarmuka yang sederhana, menarik, dan mudah dipahami oleh pengguna.
- Sistem dapat berjalan secara stabil dan responsif saat digunakan.
- Proses perhitungan dilakukan secara akurat dan cepat.

- Aplikasi dapat digunakan secara praktis sebagai alat bantu perencanaan tabungan sehari-hari.

2.2. DIAGRAM ALUR (FLOWCHART)

Flowchart digunakan untuk menggambarkan alur kerja dan logika program pada aplikasi *Savings Reminder*. *Flowchart* ini menunjukkan urutan proses yang dilakukan oleh sistem mulai dari pengguna membuka aplikasi hingga mendapatkan hasil perhitungan dan pengaturan pengingat.



2.3. SKETSA DESAIN ANTARMUKA PENGGUNA

2.3.1. Komponen Halaman Login

Judul: “LOGIN”

Masukan Pengguna (Input):

Kotak Masukan Username

Kotak Masukan Password

Tombol:

Login

“Belum punya akun?” (pindah ke halaman Register)

Pesan Pop-up (Dialog):

Error: “Isi semua field”

Error: “Username tidak ditemukan / Password salah”

2.3.2. Komponen Halaman Register

Judul: “REGISTER”

Masukan Pengguna (Input):

Kotak Masukan Username

Kotak Masukan Password

Kotak Masukan Konfirmasi Password

Tombol:

Register

“Udah punya akun?” (pindah ke halaman Login)

Pesan Pop-up:

Success: “Registrasi berhasil”

Error: “Password tidak cocok”

Error: “Username sudah dipakai”

Error: “Isi semua field”

2.3.3. Komponen Halaman Utama

Header / Judul Aplikasi:

Judul: “Savings Reminder - {username}”

Tombol:

- Logout
- Berlangsung
- Tercapai
- “+ Tambah Celengan” (membuka halaman Input/Tambah)

2.3.4. Komponen Halaman Tambah / Edit Celengan

Judul: “Tambah / Edit Celengan”

Komponen Gambar: Area tambah gambar (pilih file)

Masukan Pengguna (Input):

- Nama Tabungan
- Target Tabungan (Rp)
- Nominal Pengisian

Pemilihan Rencana: Harian, Mingguan, Bulanan

Komponen Notifikasi:

- Tampilan Jam
- Tombol edit waktu (pilih jam & menit)
- Switch Notifikasi (ON/OFF)

Tombol:

- Simpan
- Kembali

Pop-up Error:

“Semua field harus diisi!”

“Target & Nominal harus angka!”

2.3.5. Komponen Halaman Detail Tabungan

Komponen Tampilan Gambar: Gambar (jika ada)

Bar Informasi 1:

- Nama
- Harga (target)

Bar Informasi 2:

- Dibuat: (tanggal)
- Estimasi: Hari/Minggu/Bulan

Bar Informasi 3:

- Waktu
- Notifikasi

Bar Informasi 4:

- Terkumpul
- Kekurangan

Komponen Setor Tabungan: Input ‘Nominal (Rp)’

Tombol Aksi:

- Tambah (menambahkan setor ke terkumpul)
- Edit Data
- Hapus
- Kembali

2.3.6. Komponen Notifikasi Sistem

Judul Notifikasi: “Pengingat Tabungan”

Isi: “Setor Tabungan”: nama dan nominal

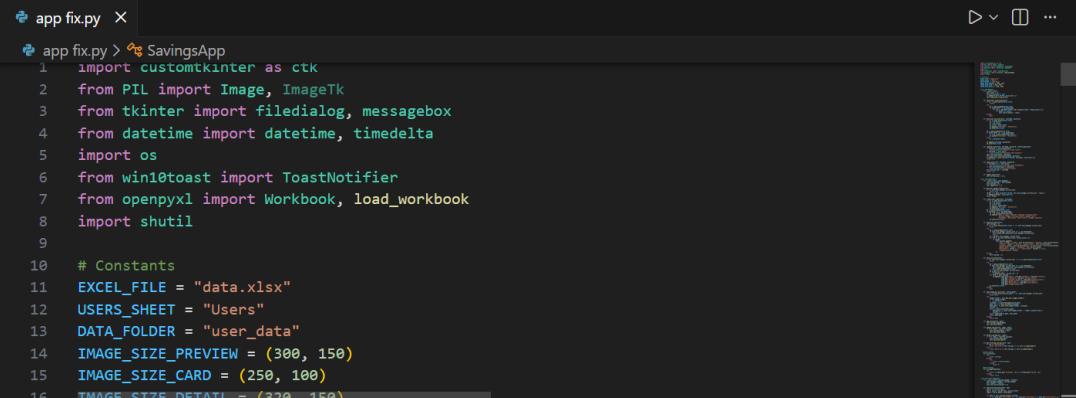
Kondisi Muncul: Jika Notif aktif (switch ON) dan mengikuti jam sistem

BAB 3

IMPLEMENTASI

3.1. PENJELASAN KODE

3.1.1. Library & Setting Global



```
app fix.py > SavingsApp
  1 import customtkinter as ctk
  2 from PIL import Image, ImageTk
  3 from tkinter import filedialog, messagebox
  4 from datetime import datetime, timedelta
  5 import os
  6 from win10toast import ToastNotifier
  7 from openpyxl import Workbook, load_workbook
  8 import shutil
  9
 10 # Constants
 11 EXCEL_FILE = "data.xlsx"
 12 USERS_SHEET = "Users"
 13 DATA_FOLDER = "user_data"
 14 IMAGE_SIZE_PREVIEW = (300, 150)
 15 IMAGE_SIZE_CARD = (250, 100)
 16 IMAGE_SIZE_DETAIL = (320, 150)
```

LIBRARY

1. Customtkinter

Library ini digunakan untuk membuat Graphical User Interface (GUI)

2. Tkinter

Digunakan untuk mengakses filedialog dan messagebox yang nantinya

3. Datetime

Digunakan untuk mengakses tanggal hari ini dan timedelta digunakan untuk menghitung selisih waktu

4. OS

Digunakan untuk mengecek dan membuat path

5. Win10toast

Digunakan untuk membuat pop-up di pojok kanan monitor

6. Openpyxl

Digunakan untuk excel yang nantinya akan jadi tempat untuk mengekspor dan menyimpan data dari program. Workbook untuk membuat file excelnya dan load_workbook yang digunakan untuk memuat file excel yang sudah ada

7. Shutil

Digunakan untuk menyalin dan memindahkan file

EXPORT DATA

EXCEL_FILE = “data.xlsx” nantinya akan digunakan untuk membuat nama file excelnnya

USERS_SHEET = “Users” nantinya digunakan untuk membuat sheet di excel yang diberi nama Users

DATA_FOLDER = “user_data” membuat sheet baru yang nantinya dijadikan folder untuk menyimpan data data yang disimpan oleh user
IMAGE_SIZE_PREVIEW = “(300, 150)” Digunakan untuk membuat ukuran preview gambar yang nantinya akan dimasukkan dan ditampilkan ke tampilan utama menjadi 300 x 150

IMAGE_SIZE_CARD = “(250, 100)” Digunakan untuk membuat ukuran gambar di tampilan utamanya menjadi 250 x 100

IMAGE_SIZE_DETAIL = “(320, 150)” Digunakan untuk nantinya merubah ukuran gambar di tampilan detail menjadi 320 x 150

3.1.2. CLASS UserManager:

```
18  class UserManager:
19      def __init__(self):
20          self.users = {}
21          self.current_user = None
22          os.makedirs(DATA_FOLDER, exist_ok=True)
23          self.load_users_from_excel()
24
25      def load_users_from_excel(self):
26          if not os.path.exists(EXCEL_FILE):
27              return
28          try:
29              wb = load_workbook(EXCEL_FILE)
30              if USERS_SHEET in wb.sheetnames:
31                  for row in wb[USERS_SHEET].iter_rows(min_row=2, values_only=True):
32                      if row[0] and row[1]:
33                          self.users[row[0]] = row[1]
34          except:
35              pass
```

```

18   class UserManager:
19       def save_user_to_excel(self, username, password):
20           if not os.path.exists(EXCEL_FILE):
21               wb = Workbook()
22               ws = wb.active
23               ws.title = USERS_SHEET
24               ws.append(["Username", "Password"])
25               wb.save(EXCEL_FILE)
26
27       wb = load_workbook(EXCEL_FILE)
28       if USERS_SHEET not in wb.sheetnames:
29           ws = wb.create_sheet(USERS_SHEET)
30           ws.append(["Username", "Password"])
31       else:
32           ws = wb[USERS_SHEET]
33
34       ws.append([username, password])
35       wb.save(EXCEL_FILE)
36
37   def register_user(self, username, password, confirm_password):
38       if password != confirm_password:
39           raise ValueError("Password tidak cocok")
40       if username in self.users:
41           raise ValueError("Username sudah dipakai")
42       self.users[username] = password
43       self.save_user_to_excel(username, password)
44       os.makedirs(os.path.join(DATA_FOLDER, username), exist_ok=True)
45       return True
46
47   class UserManager:
48       def login_user(self, username, password):
49           if username not in self.users:
50               raise ValueError("Username tidak ditemukan")
51           if self.users[username] != password:
52               raise ValueError("Password salah")
53           self.current_user = username
54           return True
55
56       def logout_user(self):
57           self.current_user = None

```

Konstruktor `__init__`

Konstruktor ini dijalankan secara otomatis saat objek `UserManager` dibuat.

Fungsinya adalah:

- Menginisialisasi atribut `users` sebagai dictionary untuk menyimpan data pengguna dalam bentuk pasangan `username` dan `password`.
- Menginisialisasi atribut `current_user` dengan nilai `None` untuk menandakan bahwa belum ada pengguna yang sedang login.
- Membuat folder penyimpanan data pengguna (`DATA_FOLDER`) apabila folder tersebut belum ada.
- Memanggil metode `load_users_from_excel()` untuk memuat data pengguna yang telah tersimpan sebelumnya di file Excel ke dalam memori aplikasi.

Metode load_user_from_excel

Langkah kerja metode ini adalah:

1. Mengecek apakah file Excel (EXCEL_FILE) tersedia. Jika file tidak ditemukan, proses dihentikan.
2. Membuka file Excel menggunakan pustaka openpyxl.
3. Memeriksa keberadaan *sheet* pengguna (USERS_SHEET).
4. Membaca data baris demi baris mulai dari baris kedua untuk menghindari header.
5. Setiap data *username* dan *password* yang valid dimasukkan ke dalam dictionary self.users.

Metode ini dibungkus dengan blok try-except untuk mencegah aplikasi berhenti apabila terjadi kesalahan saat proses pembacaan file.

Metode save_user_to_excel

Metode ini digunakan untuk menyimpan data pengguna baru ke dalam file Excel.

Tahapan yang dilakukan:

- Jika file Excel belum tersedia, maka sistem akan membuat file baru beserta *sheet* pengguna dan header kolom.
- Membuka file Excel yang ada.
- Memastikan *sheet* pengguna tersedia, atau membuatnya jika belum ada.
- Menambahkan data *username* dan *password* ke baris baru.
- Menyimpan kembali file Excel agar perubahan tersimpan secara permanen.

Metode register_user

Metode ini menangani proses pendaftaran pengguna baru.

Fungsi ini melakukan beberapa validasi, yaitu:

- Memastikan *password* dan *konfirmasi password* sesuai.
- Memastikan *username* belum terdaftar sebelumnya.

Jika seluruh validasi terpenuhi, maka:

- Data pengguna disimpan ke dalam dictionary self.users.
- Data pengguna disimpan ke dalam file Excel melalui metode `save_user_to_excel`.
- Sistem membuat folder khusus untuk pengguna tersebut di dalam `DATA_FOLDER`.
- Metode mengembalikan nilai True sebagai penanda bahwa proses pendaftaran berhasil.

Metode `login_user`

Metode ini digunakan untuk melakukan proses autentikasi pengguna.

Langkah-langkah yang dilakukan:

- Memeriksa apakah `username` terdaftar dalam sistem.
- Memverifikasi kecocokan `password` dengan data yang tersimpan.
- Jika valid, maka atribut `current_user` diisi dengan `username` yang sedang login.

Metode ini mengembalikan nilai True apabila proses login berhasil.

Metode `logout_user`

Metode ini berfungsi untuk mengakhiri sesi pengguna yang sedang aktif. Atribut `current_user` dikosongkan kembali untuk menandakan bahwa tidak ada pengguna yang sedang login.

3.1.3. CLASS SavingsManager

```

76     class SavingsManager:
77         def __init__(self, user_manager):
78             self.user_manager = user_manager
79             self.savings = []
80             self.image_refs = {}
81
82         def get_user_images_folder(self):
83             if not self.user_manager.current_user:
84                 return None
85             folder = os.path.join(DATA_FOLDER, self.user_manager.current_user, "images")
86             os.makedirs(folder, exist_ok=True)
87             return folder
88
89         def create_user_sheet(self, username):
90             if not os.path.exists(EXCEL_FILE):
91                 wb = Workbook()
92                 ws = wb.active
93                 ws.title = USERS_SHEET
94                 ws.append(["Username", "Password"])
95                 wb.save(EXCEL_FILE)
96             wb = load_workbook(EXCEL_FILE)
97             if username not in wb.sheetnames:
98                 ws = wb.create_sheet(username)
99                 ws.append(["Nama", "Target", "Nominal", "Rencana", "Estimasi Hari",
100                         "Estimasi Minggu", "Estimasi Bulan", "Gambar Path",
101                         "Terkumpul", "Notifikasi", "Notif Aktif", "Tanggal Dibuat"])
102                 wb.save(EXCEL_FILE)
103
104         def load_user_data(self):
105             self.savings = []
106             if not os.path.exists(EXCEL_FILE) or not self.user_manager.current_user:
107                 return
108             try:
109                 wb = load_workbook(EXCEL_FILE)
110                 if self.user_manager.current_user not in wb.sheetnames:
111                     self.create_user_sheet(self.user_manager.current_user)
112                     return
113                 ws = wb[self.user_manager.current_user]
114                 for row in ws.iter_rows(min_row=2, values_only=True):
115                     if row[0]:
116                         self.savings.append({
117                             "nama": row[0], "target": self.to_int(row[1]), "nominal": self.to_int(row[2]),
118                             "rencana": row[3] or "Harian", "estimasi_hari": self.to_int(row[4]),
119                             "estimasi_minggu": self.to_int(row[5]), "estimasi_bulan": self.to_int(row[6]),
120                             "gambar_path": row[7], "terkumpul": self.to_int(row[8]),
121                             "notifikasi": row[9] or "", "notif_aktif": row[10] or False,
122                             "tanggal_dibuat": row[11]
123                         })
124             except:
125                 self.savings = []
126
127         def save_user_data(self):
128             if not self.user_manager.current_user or not os.path.exists(EXCEL_FILE):
129                 return
130             try:
131                 wb = load_workbook(EXCEL_FILE)
132                 if self.user_manager.current_user not in wb.sheetnames:
133                     self.create_user_sheet(self.user_manager.current_user)
134                     wb = load_workbook(EXCEL_FILE)
135                 ws = wb[self.user_manager.current_user]
136                 if ws.max_row > 1:
137                     ws.delete_rows(2, ws.max_row - 1)
138                 for item in self.savings:
139                     ws.append([item.get("nama"), item.get("target"), item.get("nominal"),
140                               item.get("rencana"), item.get("estimasi_hari"),
141                               item.get("estimasi_minggu"), item.get("estimasi_bulan"),
142                               item.get("gambar_path"), item.get("terkumpul"),
143                               item.get("notifikasi"), item.get("notif_aktif"),
144                               item.get("tanggal_dibuat")])
145                 wb.save(EXCEL_FILE)
146             except:
147                 pass

```

```

149     def copy_image_for_user(self, source_path):
150         if not os.path.exists(source_path) or not self.user_manager.current_user:
151             return None
152         try:
153             images_folder = self.get_user_images_folder()
154             if not images_folder:
155                 return None
156             filename = os.path.basename(source_path)
157             name, ext = os.path.splitext(filename)
158             dest_path = os.path.join(images_folder, filename)
159             counter = 1
160             while os.path.exists(dest_path):
161                 dest_path = os.path.join(images_folder, f"{name}_{counter}{ext}")
162                 counter += 1
163             shutil.copy2(source_path, dest_path)
164             return dest_path
165         except:
166             return None
167
168     def add_saving(self, data):
169         self.savings.append(data)
170         self.save_user_data()
171
172     def update_saving(self, index, data):
173         if 0 <= index < len(self.savings):
174             self.savings[index] = data
175             self.save_user_data()
176
177     def delete_saving(self, index):
178         if 0 <= index < len(self.savings):
179             self.savings.pop(index)
180             self.save_user_data()
181
182     def get_filtered_savings(self, tab):
183         if tab == "Berlangsung":
184             return [d for d in self.savings if not self.is_completed(d)]
185         else:
186             return [d for d in self.savings if self.is_completed(d)]
187
188     @staticmethod
189     def to_int(val):
190         try:
191             return int(val)
192         except:
193             try:
194                 return int(float(val))
195             except:
196                 return 0
197
198     @staticmethod
199     def is_completed(data):
200         try:
201             return int(data.get("terkumpul", 0)) >= int(data.get("target", 0))
202         except:
203             return False

```

Konstruktor `__init__`

Konstruktor ini dijalankan saat objek `SavingsManager` dibuat. Fungsinya adalah:

- Menyimpan referensi ke objek `UserManager` untuk mengetahui pengguna yang sedang login.
- Menginisialisasi atribut `savings` sebagai list untuk menampung data tabungan pengguna.

- Menginisialisasi atribut `image_refs` sebagai dictionary untuk menyimpan referensi gambar agar tidak terhapus dari memori saat ditampilkan pada antarmuka.

Metode `get_user_images_folder`

Metode ini digunakan untuk menentukan dan membuat folder penyimpanan gambar milik pengguna yang sedang login.

Alur kerja metode ini:

- Mengecek apakah terdapat pengguna yang sedang login.
- Membuat folder `images` di dalam folder data pengguna jika belum tersedia.
- Mengembalikan path folder tersebut sebagai lokasi penyimpanan gambar.

Jika tidak ada pengguna yang login, metode akan mengembalikan nilai `None`.

Metode `create_user_sheet`

Metode ini berfungsi untuk membuat sheet Excel khusus untuk pengguna tertentu.

Langkah-langkah yang dilakukan:

- Mengecek apakah file Excel sudah ada, jika belum maka file akan dibuat beserta sheet pengguna utama.
- Membuka file Excel.
- Membuat sheet baru dengan nama sesuai username jika belum tersedia.
- Menambahkan header kolom yang berisi informasi tabungan, seperti nama tabungan, target, nominal, estimasi waktu, gambar, notifikasi, dan tanggal pembuatan.

Metode ini memastikan setiap pengguna memiliki sheet tersendiri untuk menyimpan data tabungannya.

Metode load_user_data

Metode ini digunakan untuk memuat data tabungan pengguna dari file Excel ke dalam memori aplikasi.

Proses yang dilakukan:

- Mengosongkan list savings sebelum memuat data baru.
- Mengecek keberadaan file Excel dan pengguna yang sedang login.
- Membuka file Excel dan memeriksa sheet milik pengguna.
- Membaca data tabungan baris demi baris (mulai dari baris kedua).
- Mengonversi setiap baris menjadi dictionary dan menambahkannya ke dalam list savings.

Metode ini menggunakan blok try-except untuk mencegah aplikasi berhenti jika terjadi kesalahan pembacaan data.

Metode save_user_data

Metode ini berfungsi untuk menyimpan seluruh data tabungan pengguna dari memori ke dalam file Excel.

Tahapan kerja metode ini:

- Memastikan pengguna sedang login dan file Excel tersedia.
- Membuka file Excel dan memastikan sheet pengguna tersedia.
- Menghapus data tabungan lama pada sheet (kecuali header).
- Menuliskan kembali seluruh data tabungan yang tersimpan di list savings.
- Menyimpan perubahan ke file Excel.

Metode ini memastikan data di Excel selalu sinkron dengan data di aplikasi.

Metode copy_image_for_user

Metode ini digunakan untuk menyalin file gambar ke folder milik pengguna.

Fungsi utama metode ini:

- Memastikan file sumber dan pengguna yang login tersedia.
- Menentukan folder tujuan penyimpanan gambar pengguna.

- Mencegah konflik nama file dengan menambahkan penomoran jika diperlukan.
- Menyalin file gambar ke folder tujuan.
- Mengembalikan path gambar yang telah disalin.

Jika terjadi kegagalan, metode akan mengembalikan nilai None.

Metode add_saving

Metode ini digunakan untuk menambahkan data tabungan baru ke dalam list savings dan langsung menyimpannya ke file Excel.

Metode update_saving

Metode ini berfungsi untuk memperbarui data tabungan berdasarkan indeks tertentu.

Jika indeks valid, data lama akan diganti dengan data baru dan disimpan kembali ke Excel.

Metode delete_saving

Metode ini digunakan untuk menghapus data tabungan berdasarkan indeks pada list savings. Setelah penghapusan, data akan langsung diperbarui di file Excel.

Metode get_filtered_saving

Metode ini digunakan untuk memfilter data tabungan berdasarkan status penyelesaian.

- Jika tab bernilai "Berlangsung", maka data tabungan yang belum mencapai target akan ditampilkan.
- Jika tab lainnya, maka data tabungan yang sudah mencapai target akan ditampilkan.

Metode Statis to_int

Metode ini digunakan untuk mengonversi nilai ke dalam tipe data integer. Jika nilai tidak dapat dikonversi secara langsung, metode akan mencoba konversi dari tipe float. Jika tetap gagal, metode mengembalikan nilai 0.

Metode Statis is_completed

Metode ini berfungsi untuk menentukan apakah suatu tabungan telah selesai.

Tabungan dianggap selesai apabila nilai terkumpul lebih besar atau sama dengan nilai target.

3.1.4. CLASS Notification_Manager

```
205     class NotificationManager:
206         def __init__(self, savings_manager, toaster):
207             self.savings_manager = savings_manager
208             self.toaster = toaster
209             self.last_notifications = {}
210
211         def check_notifications(self, app):
212             now_dt = datetime.now()
213             now_h, now_m = now_dt.hour, now_dt.minute
214             today = now_dt.date().isoformat()
215
216             for data in self.savings_manager.savings:
217                 if not data.get("notif_aktif") or not data.get("notifikasi") or data.get("notifikasi") == "-":
218                     continue
219
220                 jam = data.get("notifikasi")
221                 if not isinstance(jam, str) or ":" not in jam:
222                     continue
223                 try:
224                     h, m = map(int, jam.split(":"))
225                 except:
226                     continue
227
228                 if h != now_h or m != now_m:
229                     continue
230
231                 key = f"{data.get('nama', '')}|{data.get('tanggal_dibuat', '')}"
232                 last = self.last_notifications.get(key)
233                 allowed = self._is_allowed_to_notify(data, last, today, now_dt)
234
235                 if not allowed:
236                     continue
237
238                 self.toaster.show_toast(
239                     "Pengingat Tabungan",
240                     f"Setor tabungan: {data.get('nama', '')} - Rp {data.get('nominal', 0)}",
241                     duration=10,
242                     threaded=True
243                 )
244
245                 self.last_notifications[key] = today
246
247             app.after(30000, lambda: self.check_notifications(app))
248
```

```

249     def _is_allowed_to_notify(self, data, last, today, now_dt):
250         rencana = data.get('rencana', 'Harian')
251         if rencana == 'Harian':
252             # Untuk harian, izinkan berkali-kali dalam sehari
253             return True
254         elif rencana == 'Mingguan':
255             if not last:
256                 return True
257             try:
258                 last_dt = datetime.fromisoformat(last).date()
259                 return (now_dt.date() - last_dt).days >= 7
260             except:
261                 return True
262         elif rencana == 'Bulanan':
263             if not last:
264                 return True
265             try:
266                 last_dt = datetime.fromisoformat(last).date()
267                 return (now_dt.date() - last_dt).days >= 30
268             except:
269                 return True
270         else:
271             return True

```

Konstruktor __init__

Konstruktor ini dijalankan saat objek NotificationManager dibuat. Fungsinya adalah:

- Menyimpan referensi ke objek SavingsManager untuk mengakses data tabungan pengguna.
- Menyimpan objek toaster yang digunakan untuk menampilkan notifikasi sistem.
- Menginisialisasi dictionary last_notifications untuk mencatat waktu terakhir notifikasi dikirim, sehingga notifikasi tidak muncul berulang kali secara tidak terkendali.

Metode check_notifications

Metode ini melakukan pengecekan berkala terhadap seluruh data tabungan dan menampilkan notifikasi apabila waktu yang ditentukan telah tercapai dan syarat notifikasi terpenuhi.

Alur Kerja

1. Mengambil waktu saat ini

```
now_dt = datetime.now()
```

Digunakan untuk mendapatkan jam, menit, dan tanggal saat ini.

2. Melakukan iterasi pada seluruh data tabungan


```
for data in self.savings_manager.savings:
```
3. Validasi awal data notifikasi
 - Notifikasi harus aktif (notif_aktif)

- Waktu notifikasi harus tersedia dan valid
 - Format waktu harus berupa HH:MM
4. Mencocokkan waktu notifikasi dengan waktu sekarang
- ```
if h != now_h or m != now_m:
 continue
```
- Notifikasi hanya diproses jika jam dan menit sesuai dengan waktu saat ini.
5. Membuat kunci unik notifikasi
- ```
key = f'{data.get('nama')}{data.get('tanggal_dibuat')}
```
- Kunci ini digunakan untuk melacak riwayat notifikasi setiap tabungan.
6. Menentukan apakah notifikasi boleh ditampilkan
- ```
allowed = self._is_allowed_to_notify(...)
```
- Pengecekan ini mempertimbangkan jenis rencana tabungan (harian, mingguan, atau bulanan).
7. Menampilkan notifikasi
- Jika notifikasi diizinkan, sistem akan menampilkan pesan pengingat tabungan menggunakan *toast notification* Windows.
8. Menjadwalkan pengecekan ulang
- ```
9. app.after(30000, ...)
```
- Metode ini akan dipanggil kembali setiap 30 detik agar notifikasi tetap berjalan secara otomatis.

Metode is_allowed_to_notify

Metode ini berfungsi sebagai logika pengendali frekuensi notifikasi, sehingga notifikasi tidak muncul terlalu sering.

Mekanisme Berdasarkan Rencana Tabungan

1. Harian

Notifikasi diizinkan setiap hari tanpa batasan interval.

2. Mingguan

Notifikasi hanya diizinkan jika sudah berlalu minimal 7 hari sejak notifikasi terakhir.

3. BulananNotifikasi

hanya diizinkan jika sudah berlalu minimal 30 hari sejak notifikasi terakhir.

4. Kondisi lainnya

Jika data tidak lengkap atau terjadi kesalahan, notifikasi tetap diizinkan untuk menjaga fleksibilitas sistem.

Metode ini mengembalikan nilai True jika notifikasi boleh ditampilkan dan False jika tidak.

3.1.5. CLASS LoginFrame

```
273     class LoginFrame(ctk.CTkFrame):
274         def __init__(self, master, user_manager, on_login_success, on_show_register):
275             super().__init__(master, width=420, height=460, corner_radius=20)
276             self.user_manager = user_manager
277             self.on_login_success = on_login_success
278             self.on_show_register = on_show_register
279             self.setup_ui()
280
281         def setup_ui(self):
282             ctk.CTkLabel(self, text="LOGIN", font=("Poppins", 26, "bold")).pack(pady=10)
283
284             self.username_entry = ctk.CTkEntry(self, width=300, height=45)
285             self.username_entry.pack(pady=10)
286             self.add_placeholder(self.username_entry, "Username")
287
288             self.password_entry = ctk.CTkEntry(self, width=300, height=45)
289             self.password_entry.pack(pady=10)
290             self.add_placeholder(self.password_entry, "Password", is_password=True)
291
292             ctk.CTkButton(self, text="Login", height=45, command=self.login_action).pack(pady=15)
293             ctk.CTkButton(self, text="Belum punya akun?", fg_color="transparent", text_color="#3b82f6", hover=False,
294                         command=self.on_show_register).pack()
295
296         def add_placeholder(self, entry, text, is_password=False):
297             entry.delete(0, "end")
298             entry.insert(0, text)
299             entry.configure(text_color="#9ca3af")
300
301             def on_in(_):
302                 if entry.get() == text:
303                     entry.delete(0, "end")
304                     entry.configure(text_color="#111827")
305                     if is_password: entry.configure(show="")
306
307             def on_out(_):
308                 if not entry.get().strip():
309                     entry.insert(0, text)
310                     entry.configure(text_color="#9ca3af")
311                     if is_password: entry.configure(show="")
312
313             entry.bind("<FocusIn>", on_in)
314             entry.bind("<FocusOut>", on_out)
315
316         def login_action(self):
317             uname = self.username_entry.get()
318             pw = self.password_entry.get()
319             if uname == "Username" or pw == "Password":
320                 messagebox.showerror("Error", "Isi semua field")
321             return
322
323             try:
324                 self.user_manager.login_user(uname, pw)
325                 self.on_login_success()
326             except ValueError as e:
327                 messagebox.showerror("Error", str(e))
```

Konstruksi __init__

Konstruktor ini dijalankan saat objek LoginFrame dibuat. Parameter yang diterima adalah:

- master: jendela atau frame induk tempat LoginFrame ditempatkan.

- `user_manager`: objek `UserManager` yang digunakan untuk memproses login.
- `on_login_success`: fungsi callback yang akan dipanggil ketika login berhasil.
- `on_show_register`: fungsi callback untuk menampilkan halaman pendaftaran.

Di dalam konstruktor:

- Frame diinisialisasi dengan ukuran dan sudut membulat.
- Objek `UserManager` dan callback disimpan sebagai atribut.
- Metode `setup_ui()` dipanggil untuk membangun tampilan antarmuka.

Metode `setup_ui`

Metode ini bertugas menyusun seluruh komponen antarmuka login.

Komponen yang dibuat antara lain:

- Label judul bertuliskan “LOGIN”.
- Input teks (`CTkEntry`) untuk *username*.
- Input teks (`CTkEntry`) untuk *password*.
- Tombol login.
- Tombol navigasi untuk berpindah ke halaman pendaftaran.

Metode ini menggunakan `pack()` sebagai pengatur tata letak agar elemen tampil terpusat dan rapi secara vertikal.

Metode `add_placeholder`

Metode ini digunakan untuk menambahkan teks placeholder pada komponen input (`CTkEntry`).

Fungsi metode ini meliputi:

- Menampilkan teks petunjuk (placeholder) saat kolom kosong.
- Mengubah warna teks placeholder agar berbeda dengan teks input pengguna.
- Menghapus placeholder saat kolom mendapat fokus.

- Mengembalikan placeholder jika kolom ditinggalkan dalam keadaan kosong.
- Menyembunyikan karakter input dengan simbol * jika kolom merupakan input password.

Metode ini memanfaatkan event FocusIn dan FocusOut untuk mendeteksi interaksi pengguna dengan kolom input.

Metode login_action

Metode ini dijalankan ketika tombol Login ditekan.

Alur kerja metode ini adalah:

1. Mengambil nilai *username* dan *password* dari input pengguna.
2. Memastikan seluruh field telah diisi dan bukan placeholder.
3. Memanggil metode *login_user* dari *UserManager* untuk melakukan autentikasi.
4. Jika login berhasil, memanggil fungsi callback *on_login_success*.
5. Jika terjadi kesalahan (misalnya username tidak ditemukan atau password salah), sistem akan menampilkan pesan kesalahan menggunakan messagebox.

3.1.6. CLASS RegisterFrame

```

325     class RegisterFrame(ctk.CTkFrame):
326         def __init__(self, master, user_manager, on_register_success, on_show_login):
327             super().__init__(master, width=420, height=460, corner_radius=20)
328             self.user_manager = user_manager
329             self.on_register_success = on_register_success
330             self.on_show_login = on_show_login
331             self.setup_ui()
332
333     def setup_ui(self):
334         ctk.CTkLabel(self, text="REGISTER", font=("Poppins", 26, "bold")).pack(pady=10)
335
336         self.reg_user_entry = ctk.CTkEntry(self, width=300, height=45)
337         self.reg_user_entry.pack(pady=10)
338         self.add_placeholder(self.reg_user_entry, "Username")
339
340         self.reg_pass_entry = ctk.CTkEntry(self, width=300, height=45)
341         self.reg_pass_entry.pack(pady=10)
342         self.add_placeholder(self.reg_pass_entry, "Password", is_password=True)
343
344         self.reg_confirm_entry = ctk.CTkEntry(self, width=300, height=45)
345         self.reg_confirm_entry.pack(pady=10)
346         self.add_placeholder(self.reg_confirm_entry, "Konfirmasi Password", is_password=True)
347
348         ctk.CTkButton(self, text="Register", height=45, command=self.register_action).pack(pady=15)
349         ctk.CTkButton(self, text="Udah punya akun?", fg_color="transparent", text_color="#3b82f6", hover=False,
350                           command=self.on_show_login).pack()

```

```

352     def add_placeholder(self, entry, text, is_password=False):
353         entry.delete(0, "end")
354         entry.insert(0, text)
355         entry.configure(text_color="#9ca3af")
356
357     def on_in(_):
358         if entry.get() == text:
359             entry.delete(0, "end")
360             entry.configure(text_color="#111827")
361             if is_password: entry.configure(show="")
362
363     def on_out_():
364         if not entry.get().strip():
365             entry.insert(0, text)
366             entry.configure(text_color="#9ca3af")
367             if is_password: entry.configure(show="")
368
369     entry.bind("<FocusIn>", on_in)
370     entry.bind("<FocusOut>", on_out)
371
372
373     def register_action(self):
374         uname = self.reg_user_entry.get()
375         pw = self.reg_pass_entry.get()
376         cp = self.reg_confirm_entry.get()
377         if uname == "Username" or pw == "Password" or cp == "Konfirmasi Password":
378             messagebox.showerror("Error", "Isi semua field")
379             return
380
381         try:
382             self.user_manager.register_user(uname, pw, cp)
383             messagebox.showinfo("Success", "Registrasi berhasil")
384             self.on_register_success()
385         except ValueError as e:
386             messagebox.showerror("Error", str(e))

```

Konstruktor __init__

Metode ini berfungsi untuk menginisialisasi objek RegisterFrame.

Parameter:

- master : window atau frame induk tempat RegisterFrame ditampilkan.
- user_manager : objek yang menangani logika manajemen pengguna (registrasi dan validasi akun).
- on_register_success : fungsi callback yang dijalankan ketika proses registrasi berhasil.
- on_show_login : fungsi callback untuk berpindah kembali ke halaman login.

Di dalam metode ini, atribut-atribut penting disimpan ke dalam objek, kemudian metode setup_ui() dipanggil untuk membangun tampilan antarmuka.

Metode setup_ui

Metode setup_ui bertanggung jawab untuk membuat dan menyusun seluruh komponen antarmuka pengguna pada halaman registrasi.

Komponen yang dibuat meliputi:

- Label judul bertuliskan REGISTER.
- Field input username.
- Field input password.

- Field input konfirmasi password.
- Tombol Register untuk memproses pendaftaran akun.
- Tombol navigasi “Udah punya akun?” untuk kembali ke halaman login.

Setiap field input menggunakan metode `add_placeholder` agar memiliki teks petunjuk (placeholder) sehingga memudahkan pengguna dalam mengisi data.

Metode `add_placeholder`

Metode ini berfungsi untuk menambahkan placeholder pada komponen input (Entry).

Cara kerjanya:

- Saat field belum diisi, placeholder akan ditampilkan dengan warna abu-abu.
- Ketika field mendapatkan fokus (diklik), placeholder akan dihapus dan warna teks berubah menjadi warna normal.
- Jika field merupakan password (`is_password=True`), karakter input akan disamarkan menggunakan simbol *.
- Ketika field kehilangan fokus dan tidak diisi, placeholder akan ditampilkan kembali.

Metode ini meningkatkan kenyamanan dan kejelasan antarmuka pengguna.

Metode `register_action`

Metode ini berfungsi untuk menangani proses registrasi pengguna ketika tombol *Register* ditekan.

Alur proses:

1. Mengambil nilai username, password, dan konfirmasi password dari input pengguna.
2. Melakukan validasi awal untuk memastikan semua field telah diisi.
3. Memanggil metode `register_user` dari `UserManager` untuk memproses pendaftaran akun.

4. Jika registrasi berhasil:

- Menampilkan pesan keberhasilan.
- Menjalankan fungsi on_register_success untuk berpindah ke halaman login.

5. Jika terjadi kesalahan (misalnya username sudah digunakan atau password tidak cocok):

- Menampilkan pesan error kepada pengguna.

3.1.7. CLASS MainFrame

MainFrame merupakan halaman utama (main dashboard) pada aplikasi Savings Reminder. Kelas ini menampilkan seluruh data tabungan/celengan milik pengguna dalam bentuk kartu (card) yang rapi dan mudah dibaca. Selain itu, MainFrame juga menjadi pusat interaksi pengguna untuk:

- Menampilkan daftar tabungan berdasarkan status
- Mengganti kategori tampilan lewat tab,
- Menambah data tabungan baru melalui tombol “+ Tambah Celengan”
- Menampilkan progres pencapaian Tabungan
- Menyediakan aksi tambah, edit, dan logout pengguna

```
380 class MainFrame(ctk.CTkFrame):  
381     def __init__(self, master, savings_manager, on_logout, on_add_saving, on_edit_saving):  
382         super().__init__(master, fg_color="#f9fafb")  
383         self.savings_manager = savings_manager  
384         self.on_logout = on_logout  
385         self.on_add_saving = on_add_saving  
386         self.on_edit_saving = on_edit_saving  
387         self.current_tab = "Berlangsung"  
388         self.setup_ui()  
389  
390     def setup_ui(self):  
391         # HEADER  
392         header_frame = ctk.CTkFrame(self, fg_color="white", height=60, corner_radius=0)  
393         header_frame.pack(fill="x", pady=(0, 5))  
394  
395         header_content = ctk.CTkFrame(header_frame, fg_color="white")  
396         header_content.pack(fill="both", expand=True, padx=20, pady=10)  
397  
398         self.judul_label = ctk.CTkLabel(header_content, text="Savings Reminder", text_color="black", font=("Poppins", 22, "bold"), anchor="w")  
399         self.judul_label.pack(side="left", fill="x", expand=True)  
400  
401         ctk.CTkButton(header_content, text="Logout", width=80, height=40, fg_color="#ef4444", text_color="white", font=("Poppins", 12, "bold"), command=self.on_logout).pack(side="right", fill="x", expand=True)  
402  
403         # TAB  
404         tab_frame = ctk.CTkFrame(self, fg_color="white", corner_radius=0)  
405         tab_frame.pack(fill="x", pady=(0, 10))  
406  
407         self.tab_berlangsung = ctk.CTkButton(tab_frame, text="Berlangsung", width=300, height=35, corner_radius=10, fg_color="#3b82f6", command=lambda: self.set_tab("Berlangsung"))  
408         self.tab_berlangsung.pack(side="left", padx=(30, 5), pady=5)  
409  
410         self.tab_tercapai = ctk.CTkButton(tab_frame, text="Tercapai", width=300, height=35, corner_radius=10, fg_color="#e5e7eb", text_color="black", command=lambda: self.set_tab("Tercapai"))  
411         self.tab_tercapai.pack(side="left", padx=(5, 100), pady=5)  
412  
413         # LIST CARD  
414         self.content_frame = ctk.CTkScrolledFrame(self, width=300, height=540, fg_color="transparent", scrollbar_fg_color="transparent", scrollbar_button_color="#d1d5db", scrollbar_width=10)  
415         self.content_frame.pack(pady=(5, 10), fill="both", expand=True)  
416  
417         # FLOAT BUTTON  
418         float_frame = ctk.CTkFrame(self, fg_color="transparent")  
419         float_frame.place(relx=0.5, rely=0.92, anchor="center")  
420  
421         ctk.CTkButton(float_frame, text="+ Tambah Celengan", width=220, height=45, corner_radius=25, font=("Poppins", 15, "bold"), command=self.on_add_saving).pack()  
422
```

```

423     def set_tab(self, tab):
424         self.current_tab = tab
425         self.update_tab_header()
426         self.update_cards()
427
428     def update_tab_header(self):
429         if self.current_tab == "Berlangsung":
430             self.tab_berlangsung.configure(fg_color="#3b02f6", text_color="white")
431             self.tab_tercapai.configure(fg_color="#e5e7eb", text_color="black")
432         else:
433             self.tab_tercapai.configure(fg_color="#3b02f6", text_color="white")
434             self.tab_berlangsung.configure(fg_color="#e5e7eb", text_color="black")
435
436     def update_cards(self):
437         for widget in self.content_frame.winfo_children():
438             widget.destroy()
439         list_filtered = self.savings_manager.get_filtered_savings(self.current_tab)
440         if not list_filtered:
441             ctk.CTkLabel(self.content_frame, text="Tidak ada data", font=("Poppins", 14), text_color="#6b7280").pack(pady=20)
442             return
443         for data in list_filtered:
444             actual_index = self.savings_manager.savings.index(data)
445             self.create_card(data, actual_index)
446
447     def create_card(self, data, index):
448         card = ctk.CTkFrame(self.content_frame, corner_radius=10, fg_color="white", border_width=1, border_color="#e5e7eb")
449         card.pack(pady=8, padx=8, fill="x")
450
451         def buka_detail_event():
452             self.on_edit_saving(data, index)
453
454         card.bind("<Button-1>", lambda e: buka_detail_event())
455
456         # Judul
457         ctk.CTkLabel(card, text=data["nama"], font=("Poppins", 16, "bold"), text_color="#111827").pack(pady=(6, 0))
458
459         # Gambar
460         img_frame = ctk.CTkFrame(card, width=250, height=100, fg_color="#f3f4f6", corner_radius=10)
461         img_frame.pack(pady=6)
462
463         if data.get("gambar_path"):
464             try:
465                 img = Image.open(data["gambar_path"]).resize(IMAGE_SIZE_CARD)
466                 img_tk = ctk.CTkImage(light_image=img, size=IMAGE_SIZE_CARD)
467                 img_label = ctk.CTkLabel(img_frame, image=img_tk, text="")
468                 self.savings_manager.image_refs[id(img_label)] = img_tk
469                 img_label.pack()
470             except:
471                 ctk.CTkLabel(img_frame, text="*", font=("Arial", 38)).pack()
472         else:
473             ctk.CTkLabel(img_frame, text="*", font=("Arial", 38)).pack()
474
475         # Target
476         ctk.CTkLabel(card, text=f"Target: Rp {data['target']}").pack(pady=(0, 6))
477
478         # Info nominal & rencana
479         info_frame = ctk.CTkFrame(card, fg_color="white", corner_radius=15)
480         info_frame.pack(pady=(8,12), padx=15, fill="x")
481
482         ctk.CTkLabel(info_frame, text=f"Rp {data.get('nominal',0)} / {data.get('rencana','Harian')}", font=("Poppins", 13, "bold"), text_color="#111827").pack(pady=(6,2))
483
484         # Estimasi
485         if data.get("rencana") == "Harian":
486             estimasi_text = f"({data.get('estimasi_hari',0)} Hari"
487         elif data.get("rencana") == "Minggu":
488             estimasi_text = f"({data.get('estimasi_minggu',0)} Minggu"
489         else:
490             estimasi_text = f"({data.get('estimasi_bulan',0)} Bulan"
491
492         ctk.CTkLabel(info_frame, text=f"Estimasi Tercapai: {estimasi_text}", font=("Poppins", 12), text_color="#2563eb").pack(pady=(0,8))
493
494         # Progress bar
495         terkumpul = data.get("terkumpul", 0)
496         target = max(data.get("target", 1), 1)
497         persen = (terkumpul / target) * 100
498
499         progress_frame = ctk.CTkFrame(card, fg_color="white")
500         progress_frame.pack(pady=(8,10), padx=15, fill="x")
501
502         progress_bar = ctk.CTkProgressBar(progress_frame, height=12, corner_radius=8)
503         progress_bar.pack(fill="x", padx=10, pady=(6,2))
504         progress_bar.set(persen/100)
505
506         ctk.CTkLabel(progress_frame, text=f"{persen} % Terkumpul", font=("Poppins", 11, "bold"), text_color="#16a34a").pack(pady=(0,6))

```

1. Konstruktor Class MainFrame

Konstruktor berfungsi untuk menginisialisasi objek MainFrame sebagai halaman utama aplikasi, serta menyiapkan seluruh kebutuhan awal sebelum antarmuka ditampilkan kepada pengguna. Secara khusus, konstruktor ini:

- Menghubungkan frame utama dengan window induk aplikasi
- Menyimpan objek pengelola data tabungan (savings_manager)
- Menghubungkan event antarmuka dengan fungsi logika aplikasi melalui callback
- Menentukan kondisi awal tampilan aplikasi

- Menjamin bahwa antarmuka langsung terbentuk saat objek dibuat

Proses:

- Menyimpan seluruh parameter ke dalam atribut class
- Mengatur tab awal ke status "Berlangsung"
- Memanggil method setup_ui() untuk membangun antarmuka utama

2. Method setup_ui()

Method ini berfungsi untuk membangun dan mengatur seluruh komponen antarmuka pengguna secara terstruktur pada halaman utama aplikasi. Method ini berfokus pada pembuatan tampilan (view) tanpa menangani logika pengolahan data, sehingga mendukung pemisahan tanggung jawab dalam OOP.

- Menginisialisasi semua elemen GUI
- Mengatur tata letak dan hierarki komponen
- Menghubungkan komponen antarmuka dengan event handler
- Menyediakan navigasi utama aplikasi

Komponen yang Dibuat:

- **Header**
 - Menampilkan judul aplikasi sebagai identitas sistem
 - Menyediakan tombol logout untuk mengakhiri sesi pengguna
- **Tab Navigasi**
 - Tab *Berlangsung* untuk celengan yang masih aktif
 - Tab *Tercapai* untuk celengan yang telah mencapai target
- **Content Frame**
 - Area scroll yang menampilkan daftar card celengan
 - Menyesuaikan jumlah data yang ditampilkan secara dinamis
- **Floating Button**
 - Tombol aksi utama untuk menambah celengan baru
 - Diletakkan pada posisi strategis agar mudah diakses pengguna

3. Method set_tab()

Method ini berfungsi sebagai pengendali perpindahan tab (tab controller) pada halaman utama aplikasi. Fungsi utama method ini adalah:

- Mencatat tab yang sedang aktif
- Menjadi pusat logika perpindahan tab
- Menghindari duplikasi kode pada event klik tab

Proses:

1. Mengubah nilai atribut current_tab
2. Memperbarui tampilan visual tab melalui update_tab_header()
3. Memperbarui daftar celengan melalui update_cards()

4. Method update_tab_header()

Method ini berfungsi untuk memberikan umpan balik visual (visual feedback) kepada pengguna terkait tab yang sedang aktif. Dengan adanya perubahan warna tombol tab, pengguna dapat:

- Mengetahui posisi navigasi saat ini
- Menghindari kesalahan navigasi
- Merasakan pengalaman pengguna (UX) yang lebih baik

Keterangan:

- Tab aktif → warna biru (menandakan fokus)
- Tab tidak aktif → warna abu-abu (menandakan status pasif)

5. Method update_cards()

Method berfungsi untuk mengelola pembaruan daftar card celengan secara dinamis berdasarkan tab yang sedang aktif. Method ini memastikan bahwa data yang ditampilkan selalu sesuai dengan status terbaru, tidak menumpuk di memori, relevan dengan pilihan pengguna.

6. Method create_card()

Method ini berfungsi untuk mengonversi satu data celengan menjadi representasi visual berbentuk card. Method ini dianggap sebagai *factory*

method karena bertugas membentuk komponen GUI berdasarkan data mentah yang diterima. Fungsi method ini:

- Menampilkan informasi tabungan secara ringkas dan informatif
- Mengelola event klik untuk membuka halaman detail/edit
- Menampilkan progres pencapaian tabungan secara visual

Tampilan:

1. Nama celengan
 2. Gambar celengan
 3. Target tabungan
 4. Nominal dan rencana menabung
 5. Estimasi waktu tercapai
 6. Progress bar Tabungan
7. Alur kerja Class MainFrame
- Objek MainFrame diinisialisasi melalui konstruktur
 - Antarmuka utama dibangun menggunakan `setup_ui()`
 - Pengguna berinteraksi dengan tab atau tombol aksi
 - Sistem memfilter dan memperbarui data
 - Card celengan ditampilkan sesuai kondisi data

3.1.8. CLASS InputFrame

InputFrame merupakan halaman formulir pada aplikasi Savings Reminder. Kelas ini berfungsi sebagai media bagi pengguna untuk menambahkan data tabungan baru maupun mengubah data tabungan yang sudah ada. Seluruh informasi penting terkait tabungan—mulai dari nama, target, rencana pengisian, nominal, gambar, hingga pengaturan notifikasi—dikumpulkan dan diproses melalui kelas ini.

Fungsi utama InputFrame meliputi:

- Menyediakan form input data celengan
- Menangani proses tambah dan edit data
- Mengelola input gambar, notifikasi, dan estimasi tabungan

```

588 class InputFrame(ctk.CTkScrollableFrame):
589     def __init__(self, master, savings_manager, on_save, on_back):
590         super().__init__(master, fg_color="#f9fafb", width=700, height=600)
591         self.savings_manager = savings_manager
592         self.on_save = on_save
593         self.on_back = on_back
594         self.edit_index = None
595         self.selected_image = None
596         self.setup_ui()
597
598     def setup_ui(self):
599         self.judul_input = ctk.CTkLabel(self, text="Tambah / Edit Celengan", font=("Poppins", 20, "bold"))
600         self.judul_input.pack(pady=(20, 10))
601
602         # FRAME GAMBAR
603         self.gambar_frame = ctk.CTkFrame(self, width=300, height=150, fg_color="#e5e7eb", corner_radius=10)
604         self.gambar_frame.pack(pady=10)
605
606         self.gambar_label = ctk.CTkLabel(self.gambar_frame, text="Tambah Gambar", font=("Poppins", 14))
607         self.gambar_label.place(relx=0.5, rely=0.5, anchor="center")
608
609         self.gambar_frame.bind("<Button-1>", lambda e: self.select_image())
610         self.gambar_label.bind("<Button-1>", lambda e: self.select_image())
611
612         # INPUT DASAR
613         self.nama_entry = ctk.CTkEntry(self, placeholder_text="Nama Tabungan", width=300, height=40)
614         self.nama_entry.pack(pady=10)
615
616         self.target_entry = ctk.CTkEntry(self, placeholder_text="Target Tabungan (Rp)", width=300, height=40)
617         self.target_entry.pack(pady=10)
618
619         # RENCANA
620         rencana_label = ctk.CTkLabel(self, text="Rencana Pengisian:", font=("Poppins", 14, "bold"))
621         rencana_label.pack(pady=(15, 5))
622
623         self.opsi_var = ctk.StringVar(value="Harian")
624
625         rencana_frame = ctk.CTkFrame(self, fg_color="transparent")
626         rencana_frame.pack()
627
628         for opsi in ["Harian", "Mingguan", "Bulanan"]:
629             ctk.CTRadioButton(rencana_frame, text=opsi, variable=self.opsi_var, value=opsi).pack(side="left",
630
631         # NOMINAL
632         self.nominal_entry = ctk.CTkEntry(self, placeholder_text="Nominal Pengisian", width=300, height=40)
633         self.nominal_entry.pack(pady=(15, 5))
634
635         # NOTIFIKASI
636         notif_label = ctk.CTkLabel(self, text="Notifikasi", font=("Poppins", 14, "bold"))
637         notif_label.pack(anchor="w", padx=15)
638
639         notif_frame = ctk.CTkFrame(self, fg_color="white", corner_radius=10)
640         notif_frame.pack(pady=10, padx=20, fill="x")
641
642         self.time_var = ctk.StringVar(value="12:00")
643
644         def ubah_waktu():
645             jam_window = ctk.CTkToplevel(self)
646             jam_window.title("Pilih Waktu")
647             jam_window.geometry("200x180")
648
649             ctk.CTkLabel(jam_window, text="Pilih Jam", font=("Poppins", 13, "bold")).pack(pady=5)
650
651             jam_spin = ctk.CTkComboBox(jam_window, values=[f"{i:02d}" for i in range(24)], width=60)
652             jam_spin.pack(pady=5)
653
654             menit_spin = ctk.CTkComboBox(jam_window, values=[f"{i:02d}" for i in range(60)], width=60)
655             menit_spin.pack(pady=5)
656
657             def simpan_waktu():
658                 self.time_var.set(f"{jam_spin.get()}:{menit_spin.get()}")
659                 jam_window.destroy()
660
661             ctk.CTkButton(jam_window, text="Pilih", command=simpan_waktu).pack(pady=10)
662
663         time_label = ctk.CTkLabel(notif_frame, textvariable=self.time_var, font=("Poppins", 22, "bold"), text_color="#e5e7eb", width=40, height=40, fg_color="#e5e7eb", bg_color="white")
664         time_label.pack(side="left", padx=10, pady=10)
665
666         edit_waktu = ctk.CTkButton(notif_frame, text="Edit", width=40, height=40, fg_color="#e5e7eb", text_color="white", bg_color="white")
667         edit_waktu.pack(side="left", padx=10)
668
669         self.notif_switch_var = ctk.BooleanVar(value=False)
670         notif_switch = ctk.CTRadioButton(notif_frame, variable=self.notif_switch_var, text="")
671         notif_switch.pack(side="right", padx=10)

```

```

001     def select_image(self):
002         file_path = filedialog.askopenfilename(title="Pilih Gambar", filetypes=[("File Gambar", "*.*")])
003         if file_path:
004             self.selected_image = file_path
005             try:
006                 img = Image.open(file_path).resize(IMAGE_SIZE_PREVIEW)
007                 img_tk = ctk.CTkImage(light_image=img, size=IMAGE_SIZE_PREVIEW)
008                 self.gambar_label.configure(image=img_tk, text="")
009                 self.savings_manager.image_refs[id(self.gambar_label)] = img_tk
010             except Exception as e:
011                 print("Preview gambar gagal!", e)
012                 self.gambar_label.configure(text="preview gagal")
013
014     def load_data(self, data=None, index=None):
015         self.edit_index = index
016         self.selected_image = None
017
018         self.nama_entry.delete(0, "end")
019         self.target_entry.delete(0, "end")
020         self.nominal_entry.delete(0, "end")
021         self.opsi_var.set("Harian")
022         self.gambar_label.configure(image=None, text="Tambah Gambar")
023         self.savings_manager.image_refs.pop(id(self.gambar_label), None)
024
025         if data is not None:
026             self.nama_entry.insert(0, data["nama"])
027             self.target_entry.insert(0, str(data["target"]))
028             self.opsi_var.set(data.get("rencana", "Harian"))
029             self.nominal_entry.insert(0, str(data.get("nominal", "")))
030
031         if data.get("gambar_path"):
032             try:
033                 img = Image.open(data["gambar_path"]).resize(IMAGE_SIZE_PREVIEW)
034                 img_tk = ctk.CTkImage(light_image=img, size=IMAGE_SIZE_PREVIEW)
035                 self.gambar_label.configure(image=img_tk, text="")
036                 self.savings_manager.image_refs[id(self.gambar_label)] = img_tk
037                 self.selected_image = data["gambar_path"]
038             except:
039                 pass
040
041     def save_and_back(self):
042         nama = self.nama_entry.get().strip()
043         target = self.target_entry.get().strip()
044         rencana = self.opsi_var.get()
045         nominal = self.nominal_entry.get().strip()
046
047         if not nama or not target or not nominal:
048             messagebox.showerror("Error", "Semua field harus diisi!")
049             return
050         if not target.isdigit() or not nominal.isdigit():
051             messagebox.showerror("Error", "Target & Nominal harus angka!")
052             return
053
054         nominal = int(nominal)
055         target = int(target)
056
057         notifikasi_waktu = self.time_var.get()
058         notifikasi_status = self.notif_switch_var.get()
059
060         estimasi_hari = target // nominal if nominal > 0 else 0
061         estimasi_minggu = estimasi_hari // 7
062         estimasi_bulan = estimasi_hari // 30
063
064         if self.edit_index is not None and 0 <= self.edit_index < len(self.savings_manager.savings):
065             existing = self.savings_manager.savings[self.edit_index]
066             terkumpul_val = existing.get("terkumpul", 0)
067             tanggal_dibuat = existing.get("tanggal_dibuat")
068         else:
069             terkumpul_val = 0
070             tanggal_dibuat = datetime.now().strftime("%d-%m-%Y %H:%M:%S")
071
072         # Copy gambar ke folder user jika ada gambar baru dipilih
073         final_image_path = None
074         if self.selected_image:
075             final_image_path = self.savings_manager.copy_image_for_user(self.selected_image)
076
077         # Jika edit, pertahankan gambar lama jika tidak ada gambar baru
078         if self.edit_index is not None and 0 <= self.edit_index < len(self.savings_manager.savings) and not final_image_path:
079             final_image_path = self.savings_manager.savings[self.edit_index].get("gambar_path")
080
081         data = {
082             "nama": nama,
083             "target": target,
084             "nominal": nominal,
085             "rencana": rencana,
086             "terkumpul": terkumpul_val,
087             "tanggal_dibuat": tanggal_dibuat,
088             "final_image_path": final_image_path
089         }
090
091         self.savings_manager.add_saving(data)
092
093         self.load_data()
094
095         self.back_button.config(state="normal")
096
097         self.back_button.invoke()
098
099         self.root.destroy()
100
101

```

1. Konstruktor Class InputFrame

Fungsi:

- Menghubungkan frame input dengan window induk aplikasi
- Menyimpan objek pengelola data tabungan
- Menghubungkan tombol aksi dengan callback eksternal
- Menentukan kondisi awal form (tambah atau edit)
- Menjamin form bisa digunakan sejak pertama kali ditampilkan

2. Method setup_ui()

Method ini berfungsi untuk membangun seluruh komponen form input celengan secara lengkap dan terstruktur. Method ini berfokus pada pembuatan tampilan tanpa melakukan pengolahan data, sehingga menjaga pemisahan tanggung jawab dalam OOP.

- Menyusun elemen input dari atas ke bawah
- Menyediakan kontrol input yang jelas dan mudah digunakan
- Menghubungkan event UI dengan method terkait
- Mengatur tata letak agar konsisten dan ramah pengguna

Komponen:

1. Judul Form

Menunjukkan fungsi halaman sebagai form tambah/edit celengan

2. Frame Gambar

Area untuk memilih dan menampilkan gambar celengan

3. Input Dasar

- Nama Tabungan
- Target tabungan

4. Pilihan Rencana Menabung

- Harian
- Mingguan
- Bulanan

5. Input Nominal

6. Pengaturan Notifikasi

- Pemilihan waktu notifikasi
- Saklar aktif/nonaktif notifikasi

7. Tombol Aksi

- Tombol simpan data
- Tombol kembali ke halaman sebelumnya

3. Method select_image()

Method ini berfungsi untuk menyediakan pengguna dalam memilih gambar celengan dari perangkat dan menampilkan pratinjau gambar tersebut pada form.

- Membuka dialog pemilihan file gambar
- Menyimpan path gambar terpilih
- Menampilkan pratinjau gambar pada antarmuka
- Menangani kesalahan pemuatan gambar agar aplikasi tetap stabil

4. Method load_data()

Method ini berfungsi untuk mengisi ulang form input dengan data yang sudah ada, sehingga dapat digunakan pada proses pengeditan celengan.

Method ini juga digunakan untuk mereset form ketika pengguna ingin menambahkan data baru.

- Menentukan apakah form berada dalam mode tambah atau edit
- Mengosongkan seluruh input sebelum diisi ulang
- Mengisi form berdasarkan data yang diterima

5. Method save_and_back()

Method ini berfungsi sebagai pengendali utama proses penyimpanan data celengan, termasuk validasi, perhitungan estimasi, serta penyimpanan data ke sistem.

Fungsi:

- Mengambil seluruh nilai input dari form
- Melakukan validasi kelengkapan dan tipe data
- Mengonversi data input ke tipe yang sesuai
- Menghitung estimasi waktu pencapaian target
- Mengelola status tambah atau edit data
- Menyimpan atau memperbarui data melalui savings_manager
- Memanggil callback setelah data berhasil disimpan

6. Alur kerja Class InputFrame

- Halaman input diinisialisasi
- Antarmuka form dibangun
- Pengguna mengisi atau mengedit data
- Sistem memvalidasi input
- Estimasi tabungan dihitung
- Data disimpan atau diperbarui
- Sistem kembali ke halaman utama

3.1.9. CLASS DetailFrame

```
702 class DetailFrame(ctk.CTkFrame):  
703     def __init__(self, master, savings_manager, on_back, on_edit):  
704         super().__init__(master, fg_color="#F0fafb")  
705         self.savings_manager = savings_manager  
706         self.on_back = on_back  
707         self.on_edit = on_edit  
708         self.detail_content = ctk.CTkScrollableFrame(self, fg_color="transparent", width=700, height=600)  
709         self.detail_content.pack(fill="both", expand=True)  
710  
711     def load_detail(self, data, index):  
712         for widget in self.detail_content.winfo_children():  
713             widget.destroy()  
714  
715         # Gambar  
716         if data.get("gambar_path"):  
717             try:  
718                 img = Image.open(data["gambar_path"]).resize(IMAGE_SIZE_DETAIL)  
719                 img_tk = ctk.CTkImage(light_image=img, size=IMAGE_SIZE_DETAIL)  
720                 img_label = ctk.CTkLabel(self.detail_content, image=img_tk, text="")  
721                 self.savings_manager.image_refs[id(img_label)] = img_tk  
722                 img_label.pack(pady=10)  
723             except:  
724                 ctk.CTkLabel(self.detail_content, text="Gambar tidak tersedia.", font=("Arial", 38)).pack(pady=10)  
725             else:  
726                 ctk.CTkLabel(self.detail_content, text="Gambar berhasil dimuat.", font=("Arial", 38)).pack(pady=10)  
727  
728         # Bar 1: Nama + Target  
729         bar1 = ctk.CTkFrame(self.detail_content, fg_color="white", corner_radius=10)  
730         bar1.pack(pady=5, padx=15, fill="x")  
731  
732         nama_label = ctk.CTkLabel(bar1, text=f"Nama: {data['nama']}", font=("Poppins", 16, "bold"), text_color="#111827")  
733         nama_label.pack(side="left", padx=10, pady=8)  
734  
735         target_label = ctk.CTkLabel(bar1, text=f"Target: Rp {data['target']}.", font=("Poppins", 14), text_color="#2563eb")  
736         target_label.pack(side="right", padx=10)  
737  
738         # Bar 2: Tanggal & Estimasi  
739         bar2 = ctk.CTkFrame(self.detail_content, fg_color="white", corner_radius=10)  
740         bar2.pack(pady=5, padx=15, fill="x")  
741  
742         tanggal = datetime.now().strftime("%d %B %Y")  
743  
744         if data.get("rencana") == "Harian":  
745             estimasi_str = f"(data.get('estimasi_hari', 0)) Hari"  
746         elif data.get("rencana") == "Mingguan":  
747             estimasi_str = f"(data.get('estimasi_minggu', 0)) Minggu"  
748         else:  
749             estimasi_str = f"(data.get('estimasi_bulan', 0)) Bulan"  
750  
751         tanggal_label = ctk.CTkLabel(bar2, text=f"Dibuat: {tanggal}", font=("Poppins", 13))  
752         tanggal_label.pack(side="left", padx=10, pady=8)  
753  
754         estimasi_label = ctk.CTkLabel(bar2, text=f"Estimasi: {estimasi_str}", font=("Poppins", 13))  
755         estimasi_label.pack(side="right", padx=10)
```

Keterangan untuk Class frame, gambar, Bar 1: nama+Target, Tanggal & Estimasi.

- `__init__(self, master, savings_manager, on_back, on_edit)`
 Method ini merupakan konstruktor yang dijalankan saat halaman detail dibuat. Fungsinya untuk menginisialisasi frame utama, menyimpan `savings_manager` sebagai pengelola data, serta fungsi `on_back` dan `on_edit` untuk navigasi. Selain itu, method ini membuat dan menampilkan `CTkScrollableFrame` bernama `detail_content` sebagai wadah komponen detail agar halaman dapat digulir.
- `load_detail(self, data, index)`
 Method ini berfungsi memuat dan menampilkan detail data tabungan yang dipilih. Widget lama dihapus terlebih dahulu agar tampilan tidak menumpuk, kemudian menampilkan gambar produk atau ikon pengganti. Selanjutnya ditampilkan dua bar informasi: bar pertama berisi nama produk dan target harga, dan bar kedua berisi tanggal pembuatan serta estimasi waktu pencapaian berdasarkan rencana tabungan (harian, mingguan, atau bulanan).
- Bagian gambar
 Menampilkan gambar atau ilustrasi produk tabungan. Jika `gambar_path` tersedia, gambar dibuka dan disesuaikan ukurannya. Jika tidak ada atau terjadi kesalahan, ditampilkan ikon pengganti agar tampilan tetap konsisten.
- Bagian nama dan target
 Ditampilkan pada bar pertama dalam satu baris. Nama produk ditampilkan dengan teks lebih tebal, sedangkan target harga ditampilkan di sisi kanan dengan warna berbeda untuk menegaskan informasi utama tabungan.
- Bagian tanggal dan estimasi
 Ditampilkan pada bar kedua sebagai informasi pendukung. Tanggal menunjukkan waktu pembuatan data, sedangkan estimasi

menunjukkan perkiraan waktu pencapaian target berdasarkan rencana tabungan, sehingga membantu pengguna dalam perencanaan dan pemantauan progres.

```

53     # Bar 3: Waktu submit
54     bar3 = ctk.CTkFrame(self.detail_content, fg_color="white", corner_radius=10)
55     bar3.pack(pady=5, padx=15, fill="x")
56
57     waktu = (datetime.now() + timedelta(days=1)).strftime("%H:%M")
58     ctk.CTkLabel(bar3, text=f"Waktu: {waktu}", font=("Poppins", 13)).pack(padx=10, pady=8, anchor="w")
59
60     notif = data.get("notifikasi", "")
61     notif_label = ctk.CTkLabel(bar3, text=f"⚠️ Notifikasi: {notif}", font=("Poppins", 13), text_color="#ea580c")
62     notif_label.pack(side="left", padx=10, pady=(5, 0), anchor="w")
63
64     # Bar 4: Terkumpul & Kekurangan
65     bar4 = ctk.CTkFrame(self.detail_content, fg_color="white", corner_radius=10)
66     bar4.pack(pady=10, padx=15, fill="x")
67
68     terkumpul = data.get("terkumpul", 0)
69     kekurangan = data["target"] - terkumpul
70     if kekurangan < 0:
71         kekurangan = 0
72
73     left = ctk.CTkFrame(bar4, fg_color="white")
74     left.pack(side="left", expand=True, fill="both", padx=(0, 2), pady=5)
75
76     right = ctk.CTkFrame(bar4, fg_color="white")
77     right.pack(side="left", expand=True, fill="both", padx=(2, 0), pady=5)
78
79     ctk.CTkLabel(left, text="Terkumpul", font=("Poppins", 13, "bold"), text_color="#16a34a").pack()
80     terkumpul_label = ctk.CTkLabel(left, text=f"Rp {terkumpul:,}", font=("Poppins", 14))
81     terkumpul_label.pack()
82
83     ctk.CTkLabel(right, text="Kekurangan", font=("Poppins", 13, "bold"), text_color="#dc2626").pack()
84     kekurangan_label = ctk.CTkLabel(right, text=f"Rp {kekurangan:,}", font=("Poppins", 14))
85     kekurangan_label.pack()
86
87     line = ctk.CTkFrame(bar4, width=1, fg_color="#d1d5db")
88     line.place(relx=0.5, rely=0.5, anchor="center", relheight=0.7)

```

Keterangan pada bar 3 waktu submit, bar 4 terkumpul & kekurangan

- Bagian Bar 3 waktu submit.

Menampilkan waktu pengingat dan status notifikasi tabungan. Frame dibuat dengan latar putih. Waktu pengingat diatur dari waktu saat ini ditambah satu hari lalu diformat jam dan menit. Status notifikasi diambil dari data dan ditampilkan dengan ikon lonceng serta warna khusus untuk menunjukkan apakah notifikasi aktif.

- Bagian Bar 4 terkumpul & kekurangan

Menampilkan jumlah tabungan terkumpul dan sisa kekurangan target. Nilai terkumpul diambil dari data, sedangkan kekurangan dihitung dari selisih target dan terkumpul dengan batas minimal nol. Tampilan dibagi kiri (terkumpul) dan kanan (kekurangan) dengan warna teks berbeda, dipisahkan garis vertikal agar informasi lebih jelas dan rapi.

```

796     # Area Input: setor nominal
797     isi_frame = ctk.CTkFrame(self.detail_content, fg_color="transparent")
798     isi_frame.pack(pady=8, padx=15, fill="x")
799
800     ctk.CTkLabel(isi_frame, text="Masukkan Nominal Setor:", font=("Poppins", 13)).pack(anchor="w", padx=5)
801     setor_entry = ctk.CTkEntry(isi_frame, placeholder_text="Nominal (Rp)", width=200, height=35)
802     setor_entry.pack(side="left", padx=(5, 8), pady=8)
803
804     def tambah_setor():
805         nonlocal terkumpul, kekurangan
806         val = setor_entry.get().strip()
807         if not val:
808             messagebox.showerror("Error", "Masukkan nominal yang ingin disetor.")
809             return
810         if not val.isdigit():
811             messagebox.showerror("Error", "Nominal harus berupa angka.")
812             return
813         jumlah = int(val)
814         if jumlah < 0:
815             messagebox.showerror("Error", "Nominal harus lebih besar dari 0.")
816             return
817
818         self.savings_manager.savings[index]["terkumpul"] = self.savings_manager.savings[index].get("terkumpul", 0) + jumlah
819         self.savings_manager.save_user_data()
820
821         terkumpul = self.savings_manager.savings[index]["terkumpul"]
822         kekurangan = self.savings_manager.savings[index]["target"] - terkumpul
823         if kekurangan < 0:
824             kekurangan = 0
825
826         terkumpul_label.config(text=f"Rp {terkumpul:,}")
827         kekurangan_label.config(text=f"Rp {kekurangan:,}")
828
829         setor_entry.delete(0, "end")
830
831     ctk.CTkButton(isi_frame, text="Tambah", width=90, height=35, fg_color="#16a34a", command=tambah_setor).pack(side="left", padx=(8, 5))
832
833     # Tombol
834     tombol_Frame = ctk.CTkFrame(self.detail_content, fg_color="transparent")
835     tombol_Frame.pack(pady=20)
836
837     ctk.CTkButton(tombol_Frame, text="Edit Data", width=120, height=40, fg_color="#388ef5", font=("Poppins", 14, "bold"), command=lambda: self.on_edit(data, index)).pack(side="left", padx=5)
838
839     def hapus_data():
840         ask = messagebox.askyesno("Hapus Data", "Yakin ingin menghapus tabungan '" + data["nama"] + "'?")
841         if ask:
842             self.savings_manager.delete_saving(index)
843             self.on_back()
844
845     ctk.CTkButton(tombol_Frame, text="Hapus", width=120, height=40, fg_color="#ff4444", text_color="white", font=("Poppins", 14, "bold"), command=hapus_data).pack(side="left", padx=5)
846
847     ctk.CTkButton(tombol_Frame, text="Kembali", width=120, height=40, fg_color="#9ca3af", font=("Poppins", 14, "bold"), command=self.on_back).pack(side="left", padx=5)

```

Keterangan area input & setor nominal, tombol.

- Bagian area input setor nominal.

Bagian area input setor nominal digunakan untuk memasukkan jumlah uang yang akan ditambahkan ke tabungan. Frame `isi_frame` dibuat sebagai wadah input dengan latar transparan agar menyatu dengan tampilan detail. Di dalamnya terdapat label sebagai petunjuk pengisian dan sebuah field input (CTkEntry) yang digunakan pengguna untuk memasukkan nominal setor. Input ini dibatasi hanya untuk nilai nominal tabungan dalam bentuk angka.

- Fungsi `tambah_setor()`.

berfungsi untuk memproses penambahan nominal tabungan. Fungsi ini melakukan validasi data yang dimasukkan, yaitu memastikan input tidak kosong, berupa angka, dan bernilai lebih dari nol. Jika validasi berhasil, sistem akan menambahkan nominal tersebut ke jumlah tabungan yang telah terkumpul, kemudian menyimpan perubahan data menggunakan `savings_manager`. Setelah data diperbarui, sistem menghitung ulang nilai terkumpul dan kekurangan target, lalu memperbarui tampilan label agar pengguna dapat langsung melihat

perubahan saldo tabungan. Terakhir, field input dikosongkan kembali untuk pengisian berikutnya.

- Tombol Tambah.

Tombol Tambah dihubungkan dengan fungsi tambah_setor() melalui parameter command. Tombol ini berfungsi untuk menjalankan proses penambahan nominal tabungan sesuai dengan data yang diinput oleh pengguna.

- Pada bagian tombol aksi

Pada bagian tombol aksi, terdapat tombol Edit Data yang digunakan untuk mengubah informasi tabungan. Tombol ini memanggil fungsi on_edit dengan mengirimkan data dan indeks tabungan yang dipilih. Selain itu, terdapat fungsi hapus_data() yang berfungsi untuk menghapus data tabungan.

3.1.10. CLASS SavingsApp

```
848 class SavingsApp:  
849     def __init__(self):  
850         ctk.set_appearance_mode("light")  
851         ctk.set_default_color_theme("blue")  
852  
853         self.app = ctk.CTk()  
854         self.app.title("Savings Reminder")  
855         self.app.geometry("700x600")  
856         self.app.resizable(False, False)  
857  
858         self.toaster = ToastNotifier()  
859  
860         self.user_manager = UserManager()  
861         self.savings_manager = SavingsManager(self.user_manager)  
862         self.notification_manager = NotificationManager(self.savings_manager, self.toaster)  
863  
864         self.login_frame = LoginFrame(self.app, self.user_manager, self.on_login_success, self.show_register)  
865         self.register_frame = RegisterFrame(self.app, self.user_manager, self.show_login, self.show_login)  
866         self.main_frame = MainFrame(self.app, self.savings_manager, self.logout, self.show_input, self.show_detail)  
867         self.input_frame = InputFrame(self.app, self.savings_manager, self.back_to_main, self.back_to_main)  
868         self.detail_frame = DetailFrame(self.app, self.savings_manager, self.back_to_main, self.show_input)  
869  
870     def show_login(self):  
871         self.register_frame.pack_forget()  
872         self.login_frame.pack(pady=40)  
873  
874     def show_register(self):  
875         self.login_frame.pack_forget()  
876         self.register_frame.pack(pady=40)  
877  
878     def on_login_success(self):  
879         self.savings_manager.load_user_data()  
880         self.login_frame.pack_forget()  
881         self.register_frame.pack_forget()  
882         self.main_frame.pack(fill="both", expand=True)  
883         self.main_frame.judul_label.configure(text=f"Savings Reminder - {self.user_manager.current_user}")  
884         self.main_frame.update_tab_header()  
885         self.main_frame.update_cards()  
886
```

Keterangan Class SavingsApp dengan pendefisiannya.

- Class SavingsApp merupakan class utama yang berfungsi sebagai pengendali keseluruhan aplikasi *Savings Reminder*. Pada method `__init__`,
- Fungsi `show_login` digunakan untuk menampilkan halaman login. Fungsi ini menyembunyikan halaman registrasi yang sedang aktif, kemudian menampilkan halaman login agar pengguna dapat masuk ke aplikasi menggunakan akun yang telah terdaftar.
- Fungsi `show_register` berfungsi untuk menampilkan halaman pendaftaran akun (register). Pada fungsi ini, halaman login disembunyikan dan halaman register ditampilkan sehingga pengguna dapat membuat akun baru.
- Fungsi `on_login_success` dijalankan ketika proses login berhasil. Fungsi ini memuat data tabungan milik pengguna yang sedang login, menyembunyikan halaman login dan register, lalu menampilkan halaman utama aplikasi. Selain itu, fungsi ini juga memperbarui judul aplikasi dengan nama pengguna yang aktif serta memperbarui tampilan data tabungan agar sesuai dengan akun yang sedang digunakan.

```

886     def logout(self):
887         self.savings_manager.save_user_data()
888         self.user_manager.logout_user()
889         self.savings_manager.savings.clear()
890         self.main_frame.pack_forget()
891         self.input_frame.pack_forget()
892         self.detail_frame.pack_forget()
893         self.show_login()
894
895
896     def show_input(self, data=None, index=None):
897         self.main_frame.pack_forget()
898         self.detail_frame.pack_forget()
899         self.input_frame.pack(fill="both", expand=True)
900         self.input_frame.load_data(data, index)
901
902     def show_detail(self, data, index):
903         self.main_frame.pack_forget()
904         self.input_frame.pack_forget()
905         self.detail_frame.pack(fill="both", expand=True)
906         self.detail_frame.load_detail(data, index)
907
908     def back_to_main(self):
909         self.input_frame.pack_forget()
910         self.detail_frame.pack_forget()
911         self.main_frame.pack(fill="both", expand=True)
912         self.main_frame.update_cards()
913
914     def run(self):
915         self.show_login()
916         self.notification_manager.check_notifications(self.app)
917         self.app.mainloop()
918
919 if __name__ == "__main__":
920     app = SavingsApp()
921     app.run()
922

```

Keterangan pendefisian dari Class App, If_name.

- Fungsi logout digunakan untuk mengakhiri sesi pengguna yang sedang login. Pada fungsi ini, data tabungan terlebih dahulu disimpan agar tidak hilang, kemudian sistem mengeluarkan pengguna dari akun yang aktif. Setelah itu, data tabungan yang ada di memori dikosongkan dan seluruh halaman utama, input, serta detail disembunyikan. Terakhir, aplikasi menampilkan kembali halaman login sehingga pengguna dapat masuk dengan akun lain.
- Fungsi show_input berfungsi untuk menampilkan halaman input tabungan. Fungsi ini menyembunyikan halaman utama dan halaman detail, kemudian menampilkan halaman input. Parameter data dan index digunakan untuk mengirimkan data tabungan tertentu apabila pengguna ingin menambah atau mengedit data. Setelah halaman input ditampilkan,

fungsi ini memanggil `load_data` untuk memuat data yang akan ditampilkan atau diedit.

- Fungsi `show_detail` digunakan untuk menampilkan halaman detail tabungan. Fungsi ini menyembunyikan halaman utama dan halaman input, kemudian menampilkan halaman detail. Setelah itu, fungsi ini memanggil `load_detail` untuk menampilkan informasi lengkap tabungan berdasarkan data dan indeks yang dipilih oleh pengguna.
- Fungsi `back_to_main` berfungsi untuk kembali ke halaman utama (Home). Fungsi ini menyembunyikan halaman input dan halaman detail, kemudian menampilkan kembali halaman utama. Selain itu, fungsi ini memperbarui tampilan daftar tabungan agar data yang ditampilkan selalu sesuai dengan kondisi terbaru.
- Fungsi `run` merupakan fungsi utama untuk menjalankan aplikasi. Fungsi ini menampilkan halaman login sebagai tampilan awal, mengaktifkan sistem pengecekan notifikasi tabungan, dan menjalankan loop utama aplikasi (mainloop) agar aplikasi dapat berinteraksi dengan pengguna.
- Bagian `if_name_berfungsi` sebagai titik awal eksekusi program. Jika file dijalankan secara langsung, maka objek `SavingsApp` akan dibuat dan fungsi `run` akan dipanggil untuk menjalankan aplikasi *Savings Reminder*.

3.2. BAGIAN REVISI KODE

3.3.1. Revisi Validasi Input Nominal Setoran

Pada bagian ini dilakukan revisi pada fungsi tambah_setor() di kelas DetailFrame. Pada versi sebelum revisi, validasi input nominal menggunakan isdigit() dan pemeriksaan jumlah ≤ 0 . Setelah dilakukan revisi ada penambahan pengecekan isalpha() untuk memastikan input bukan huruf dan pemisahan kondisi untuk nilai negatif (< 0) dan nol ($= 0$).

Sebelum:

```
766  class DetailFrame(ctk.CTkFrame):
775      def load_detail(self, data, index):
766          def tambah_setor():
767              nonlocal terkumpul, kekurangan
768              val = setor_entry.get().strip()
769              if not val:
770                  messagebox.showerror("Error", "Masukkan nominal yang ingin disetor.")
771                  return
772              if not val.isdigit():
773                  messagebox.showerror("Error", "Nominal harus berupa angka.")
774                  return
775              jumlah = int(val)
776              if jumlah <= 0:
777                  messagebox.showerror("Error", "Nominal harus lebih besar dari 0.")
778                  return
```

Sesudah:

```
763  class DetailFrame(ctk.CTkFrame):
772      def load_detail(self, data, index):
763          def tambah_setor():
764              nonlocal terkumpul, kekurangan
765              val = setor_entry.get().strip()
766              if not val:
767                  messagebox.showerror("Error", "Masukkan nominal yang ingin disetor.")
768                  return
769              ✦
770              if val.isalpha():
771                  messagebox.showerror("Error", "Nominal harus berupa angka.")
772                  return
773              jumlah = int(val)
774              if jumlah < 0:
775                  messagebox.showerror("Error", "Nominal harus lebih besar dari 0.")
776                  return
777              if jumlah == 0:
778                  messagebox.showerror("Error", "Nominal harus lebih besar dari 0.")
779                  return
```

3.3.1. Revisi Perhitungan Estimasi Target Tabungan

Revisi dilakukan pada fungsi save_and_back() di kelas InputFrame yang berfungsi menghitung estimasi waktu pencapaian target tabungan. Sebelum revisi, logika estimasi minggu dan bulan kurang akurat karena dihitung berdasarkan hasil pembagian estimasi hari (estimasi_hari // 7 dan estimasi_hari // 30). Setelah revisi, perhitungan estimasi minggu dan bulan dihitung langsung dari target dan nominal setoran.

Sebelum:

```
571     class InputFrame(ctk.CTkScrollableFrame):
704         def save_and_back(self):
723             estimasi_hari = target // nominal if nominal > 0 else 0
724             estimasi_minggu = estimasi_hari // 7
725             estimasi_bulan = estimasi_hari // 30
```

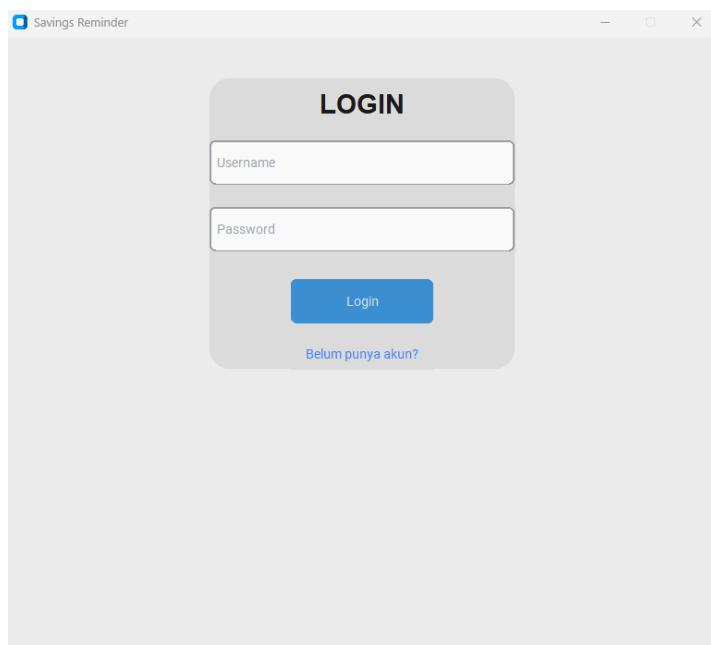
Sesudah:

```
568     class InputFrame(ctk.CTkScrollableFrame):
701         def save_and_back(self):
720             estimasi_hari = target // nominal if nominal > 0 else 0
721             estimasi_minggu = target // nominal
722             estimasi_bulan = target // nominal
```

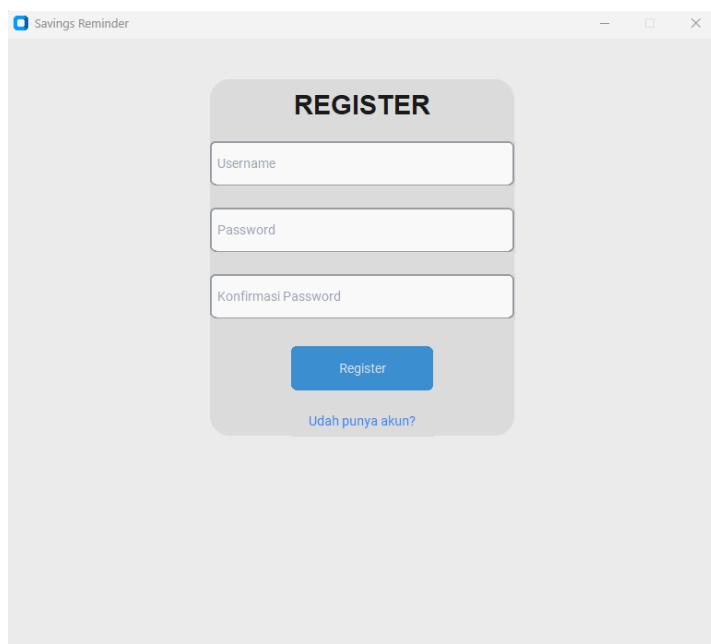
3.3. SCREENSHOT APLIKASI

Untuk membantu pembaca memahami alur kerja dan tampilan antarmuka aplikasi secara lebih jelas, maka pada bagian ini disajikan beberapa screenshot yang menampilkan setiap halaman dan fitur utama pada aplikasi yang telah dikembangkan.

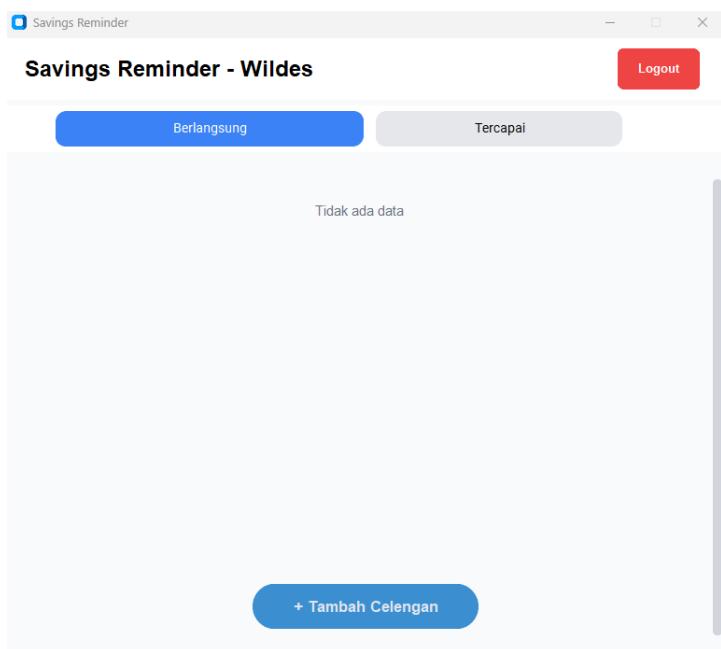
3.3.1. Halaman Login



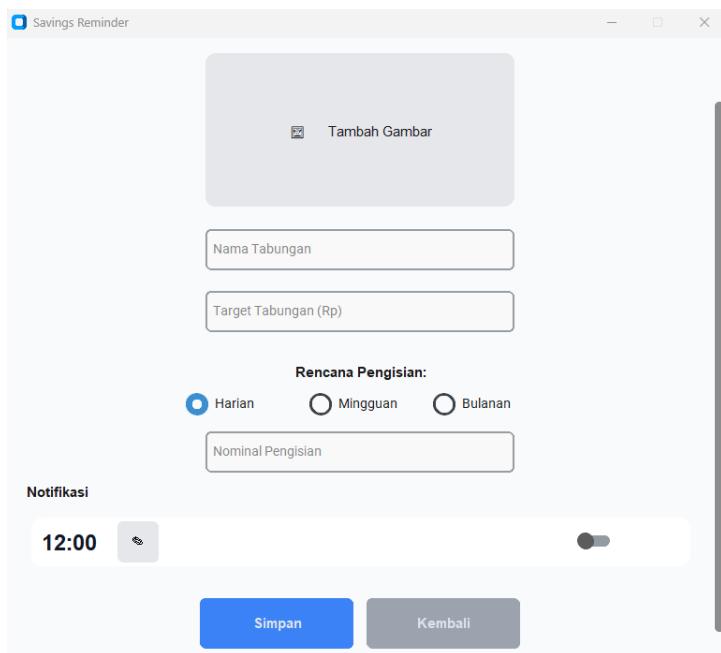
3.3.1. Halaman Register



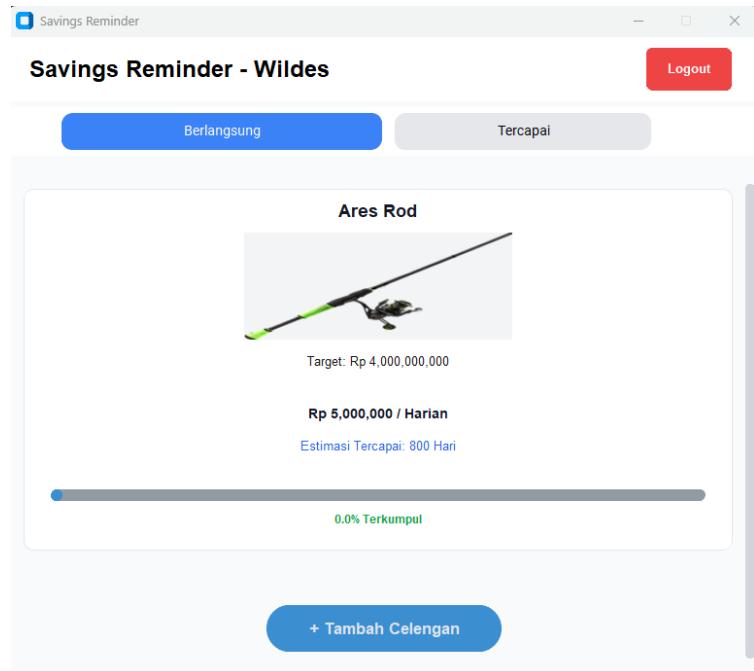
3.3.1. Halaman Utama (Main Page)



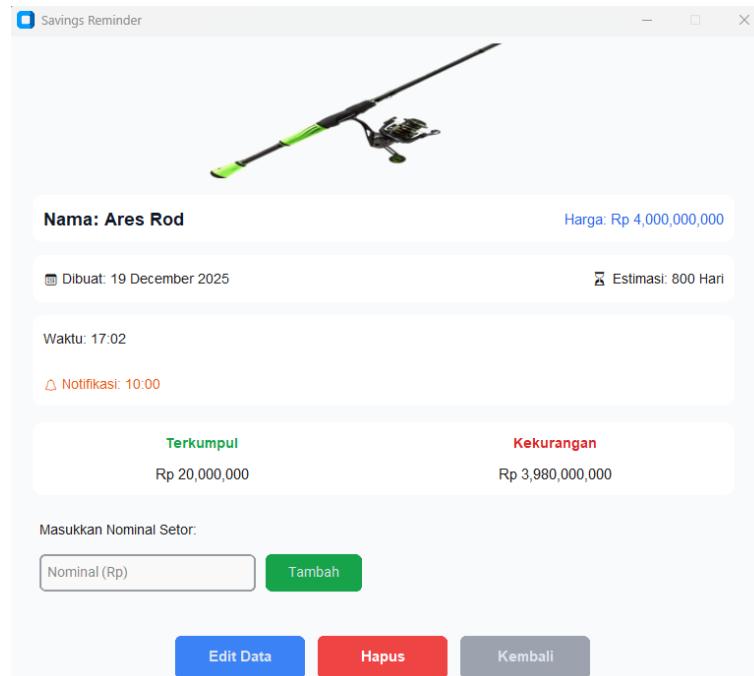
3.3.1. Halaman Input Celengan



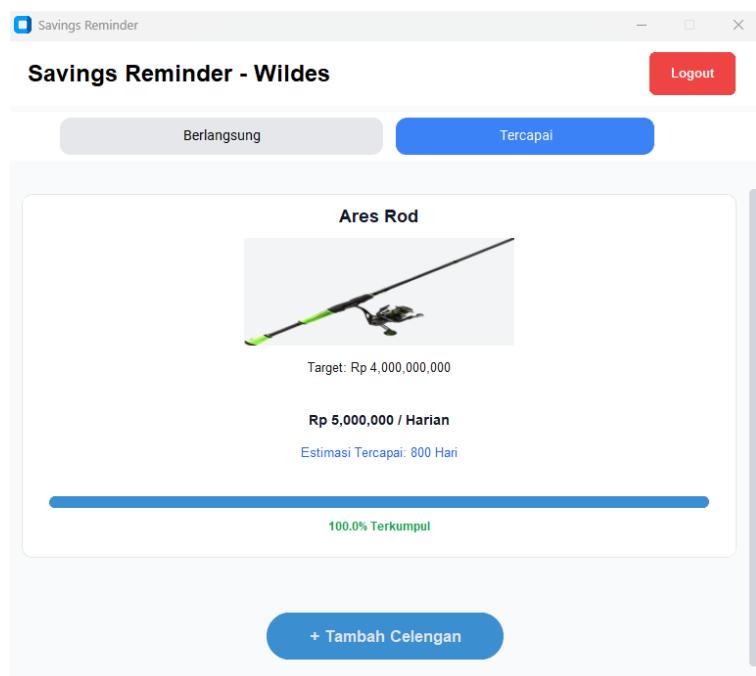
3.3.1. Halaman Utama dengan Data Tabungan Berlangsung



3.3.1. Halaman Detail Tabungan



3.3.1. Halaman Utama dengan Data Tabungan Tercapai



BAB 4

LAMPIRAN

4.1. LAMPIRAN FILE

Dibawah ini adalah folder lengkap kode lengkap Savings Reminder
[Link Drive Savings Reminder](#)

DAFTAR PUSTAKA

- Lusardi, A., & Mitchell, O. S. (2014). The economic importance of financial literacy: Theory and evidence. *Journal of Economic Literature*, 52(1), 5–44. <https://doi.org/10.1257/jel.52.1.5>
- Otoritas Jasa Keuangan. (2023). *Survei nasional literasi dan inklusi keuangan (SNLIK) 2022*. Otoritas Jasa Keuangan.
- Pratama, A., et al. (2022). Pengembangan aplikasi pengelola keuangan pribadi berbasis Android dengan metode Waterfall. *Jurnal Teknik Informatika*, 15(2).
- Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill Education.
- Ramsey, D. (2013). *The total money makeover: A proven plan for financial fitness*. Thomas Nelson.
- Celenganku. (2025). *Celenganku – Pencatat tabungan* [Aplikasi mobile]. Google Play Store. <https://play.google.com/store/apps/details?id=id.celenganku.app>