

LAPORAN AKHIR PROYEK

Aplikasi Latihan Soal UTBK



Nama Mahasiswa: Daaniys Balqis

NIM: 25031554060

Kelas: 2025 G

Mata Kuliah: Pemograman Dasar

Dosen Pengampu:

1. Hasanuddin Al-Habib, S.Si., M.Si.
2. Dr. Heri Purnawan, S.Si., M.Si.

PROGRAM STUDI S1 SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2025

DAFTAR ISI

DAFTAR ISI.....	ii
BAB I PENDAHULUAN	1
A. Latar Belakang	1
B. Rumusan Masalah	1
C. Tujuan	1
BAB II ANALISIS DAN PERANCANGAN	3
Analisis Kebutuhan Aplikasi.....	3
Diagram Alur	3
Sketsa Desain Antarmuka	4
BAB III IMPLEMENTASI.....	5
A. Penjelasan Kode Program	5
B. Manual Penggunaan Aplikasi.....	5
C. Screenshoot Aplikasi.....	6
BAB IV PENUTUP	8
A. Kesimpulan.....	7
B. Saran.....	8
LAMPIRAN.....	10
DAFTAR PUSTAKA	13

BAB I

PENDAHULUAN

A. Latar Belakang

Ujian Tulis Berbasis Komputer (UTBK) adalah salah satu tahapan penting bagi siswa yang ingin melanjutkan pendidikan ke perguruan tinggi negeri. Untuk menghadapi UTBK, siswa membutuhkan latihan soal yang cukup serta pembahasan yang jelas agar dapat memahami materi dengan baik.

Seiring dengan perkembangan teknologi, proses belajar tidak lagi terbatas pada buku cetak. Aplikasi digital dapat dijadikan solusi untuk membantu siswa belajar secara lebih fleksibel dan interaktif. Namun, masih banyak siswa yang kesulitan menemukan aplikasi latihan soal UTBK yang sederhana, mudah digunakan, dan memiliki pembahasan yang jelas.

Oleh karena itu, pada proyek ini dibuat sebuah aplikasi latihan soal UTBK berbasis Python dengan tampilan GUI (Graphical User Interface). Aplikasi ini diharapkan dapat membantu siswa dalam berlatih soal UTBK secara mandiri, efektif, dan mudah digunakan.

B. Rumusan Masalah

1. Bagaimana cara menyediakan soal-soal UTBK yang sesuai dengan kurikulum dan standar terbaru dalam bentuk aplikasi digital?
2. Bagaimana meningkatkan aksesibilitas dan efektivitas belajar siswa melalui aplikasi latihan soal UTBK?
3. Bagaimana menyajikan fitur pembahasan yang interaktif agar pengguna dapat memahami materi dengan lebih baik?
4. Apa saja tantangan teknis dan pedagogis dalam merancang aplikasi edukatif yang menarik dan mudah digunakan?

C. Tujuan

Sesuai dengan rumusan masalah yang telah dijabarkan di atas, maka dapat diketahui bahwa tujuan dari proyek ini antara lain:

1. Menyediakan platform digital yang berisi kumpulan soal UTBK dari berbagai mata pelajaran secara terstruktur dan terstandarisasi.
2. Mempermudah siswa dalam mengakses latihan soal UTBK kapan saja dan di mana saja melalui perangkat mobile atau komputer.
3. Menyediakan fitur pembahasan dan penilaian otomatis untuk membantu siswa memahami kesalahan dan meningkatkan kemampuan.
4. Mendorong semangat belajar mandiri dan meningkatkan kesiapan siswa menghadapi UTBK secara lebih efektif.

BAB II

ANALISIS DAN PERANCANGAN

Analisis Kebutuhan Aplikasi

Aplikasi latihan soal UTBK ini memiliki kebutuhan sebagai berikut:

1. Menampilkan menu utama (Start, Daily Challenge, History).
2. Menampilkan pilihan paket soal (PK, PM, PPU, PBM, dll).
3. Menampilkan soal, pilihan jawaban, dan pembahasan.
4. Menghitung skor secara otomatis.
5. Menyimpan riwayat pengerjaan soal (history).
6. Menyediakan fitur timer.

Diagram Alur



Sketsa Desain Antarmuka

Desain aplikasi dibuat sederhana agar mudah digunakan oleh pengguna.

Tampilan aplikasi terdiri dari:

- Menu utama
- Menu paket soal
- Halaman soal
- Halaman history

BAB III

IMPLEMENTASI

A. Penjelasan Kode Program

Aplikasi ini dibuat menggunakan bahasa pemrograman **Python** dengan library **Tkinter** untuk membangun antarmuka grafis.

1. Struktur Program

Folder/File	Fungsi
src/	Tempat kode utama program
main.py	File utama yang dijalankan
utils.py	Berisi fungsi bantu (baca file, simpan skor, dll)
data/	Menyimpan data soal & history
soal.json	Database soal UTBK
history.json	Riwayat skor pengguna

2. Contoh Kode Penting

Contoh penggunaan class pada program :

```
24
25  # ===== APP =====
26  class QuizApp(tk.Tk):
27      def __init__(self):
28          super().__init__()
29          self.title("Latihan Soal UTBK")
30          self.resizable(False, False)
```

Kode tersebut digunakan untuk membuat window utama aplikasi menggunakan konsep **Object Oriented Programming (OOP)**.

B. Manual Penggunaan Aplikasi

- Jalankan file main.py
- Pilih menu **Start**
- Pilih paket dan level soal
- Jawab soal yang ditampilkan
- Klik tombol **Next** untuk melanjutkan
- Lihat pembahasan
- Riwayat dapat dilihat pada menu **History**

C. Screenshoot Aplikasi

Gambar berikut merupakan tampilan **halaman utama (Home Screen)** dari aplikasi *Latihan Soal UTBK*.

Pada halaman ini, pengguna disediakan beberapa menu utama, yaitu:

- **Start** untuk memulai latihan soal UTBK berdasarkan paket dan tingkat kesulitan.
- **Daily Challenge** untuk mengerjakan tantangan harian dengan jumlah soal terbatas.
- **History** untuk melihat riwayat skor hasil pengerjaan sebelumnya.

Desain antarmuka aplikasi menggunakan warna pastel agar terlihat nyaman di mata dan mudah digunakan oleh pengguna.



BAB IV

PENUTUP

A. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian aplikasi latihan soal UTBK yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1. Aplikasi latihan soal UTBK dibuat menggunakan bahasa pemrograman Python dengan bantuan library Tkinter sebagai antarmuka pengguna.
2. Aplikasi ini mampu menyediakan kumpulan soal UTBK dari beberapa paket mata pelajaran yang disusun secara terstruktur dan mudah diakses oleh pengguna.
3. Fitur utama seperti pemilihan paket soal, level kesulitan, timer, penilaian otomatis, pembahasan soal, serta riwayat pengerjaan telah berjalan dengan baik sesuai dengan tujuan awal proyek.
4. Penggunaan format data JSON dalam penyimpanan soal dan history memudahkan proses pengelolaan data serta mendukung konsep file access.
5. Secara keseluruhan, aplikasi ini dapat membantu siswa dalam melakukan latihan soal UTBK secara mandiri dan meningkatkan kesiapan dalam menghadapi ujian.

B. Saran

Untuk pengembangan aplikasi di masa mendatang, terdapat beberapa saran yang dapat dipertimbangkan, antara lain:

1. Menambahkan lebih banyak variasi soal dan paket mata pelajaran agar latihan menjadi lebih lengkap.
2. Mengembangkan aplikasi ke platform mobile (Android atau iOS) sehingga dapat diakses melalui smartphone.
3. Menambahkan fitur grafik atau statistik perkembangan nilai pengguna agar pengguna dapat memantau kemajuan belajar.
4. Meningkatkan tampilan antarmuka agar lebih interaktif dan menarik.
5. Menambahkan sistem akun pengguna agar data hasil latihan dapat tersimpan secara personal.

LAMPIRAN

```

1 # -*- coding: utf-8 -*-
2
3 import tkinter as tk
4 from tkinter import ttk, messagebox
5 import random
6 import uuid
7 import datetime
8 import sys
9
10 from utils import (
11     ensure_dir, load_questions_for_package, normalize_correct_answer,
12     pick_questions_with_fresh_priority, pick_daily_challenge_by_level,
13     load_json, save_json, HISTORY_FILE, list_packages_from_json
14 )
15
16 # ===== CONFIG =====
17 ensure_dir()
18
19 QUESTIONS_PER_LEVEL = 8
20 SESSION_SECONDS = 75 * 60
21 DAILY_NUM = 5
22 DAILY_POINTS = {"easy": 1, "medium": 2, "hard": 3}
23
24 # ===== APP =====
25 class QuizApp(tk.Tk):
26     def __init__(self):
27         super().__init__()
28         self.title("Latihan Soal UTS")
29         try:
30             self.state("normal")
31         except Exception:
32             pass
33
34         self.style()

```

```

35
36     # ===== INIT =====
37     def _init__(self):
38         self.container = tk.Frame(self)
39         self.container.pack(fill="both", expand=True)
40
41         self.history = load_json(HISTORY_FILE, [])
42         self.packages = list_packages_from_json()
43
44         # state
45         self.questions = []
46         self.idx = 0
47         self.score = 0
48         self.daily_score = 0
49         self.daily_breakdown = {"easy": 0, "medium": 0, "hard": 0}
50         self.is_daily = False
51         self.showing_explanation = False
52
53         self.timer_level = None
54         self.remaining = SESSION_SECONDS
55         self.timer_job = None
56
57         self.current_package = None
58         self.current_level = None
59
60         self._home()
61
62     # ===== STYLE =====
63     def _style(self):
64         s = ttk.Style()
65         try:
66             s.theme_use("clam")
67         except Exception:
68             pass

```

```

69
70     # ===== WIDGET =====
71     def _widget(self):
72         bg = "#f0f0f0"
73         text = "#333333"
74         btn = "#ffffff"
75         hover = "#e0e0e0"
76
77         s.configure("Frame", background=bg)
78         s.configure("Label", background=bg, foreground=text)
79         s.configure("Title.Label", font=("Segoe UI", 16, "bold"))
80         s.configure("Subtitle.Label", font=("Segoe UI", 14))
81
82         s.configure(
83             "Button",
84             font=("Segoe UI", 16),
85             padding=11,
86             background=bg,
87             foreground=text,
88         )
89         s.map("Button", background=[("active", hover)])
90
91         s.configure(
92             "Choice.Radiobutton",
93             background=bg,
94             foreground=text,
95             font=("Segoe UI", 16),
96             padding=6
97         )
98         s.map("Choice.Radiobutton", background=[("active", hover)])
99
100     # ===== TILED =====
101     def _clear(self):
102         # Clear the self.container widget widget widget widget widget widget

```

```

103
104     # ===== TILED =====
105     def _start_timer(self):
106         if self.timer_job:
107             try:
108                 self.after_cancel(self.timer_job)
109             except Exception:
110                 pass
111             self.timer_job = None
112
113         self.remaining = SESSION_SECONDS
114         self.timer_job = self.after(1000, self._tick)
115
116     def _tick(self):
117         try:
118             s, r = divmod(self.remaining, 60)
119             if self.timer_label and getattr(self.timer_label, "info_exists", lambda: False)():
120                 try:
121                     self.timer_label.config(text="%02d:%02d" % (s, r))
122                 except tk.TclError:
123                     pass
124             except Exception:
125                 pass
126
127         if self.remaining <= 0:
128             if self.timer_job:
129                 try:
130                     self.after_cancel(self.timer_job)
131                 except Exception:
132                     pass

```

```

133
134     def _home(self):
135         # Messagebox.showinfo("Info", "Home screen")
136         except Exception:
137             pass
138         self._init()
139
140         self.remaining = s
141         try:
142             self.timer_job = self.after(1000, self._tick)
143         except Exception:
144             pass
145         self.timer_job = None
146
147     # ===== TILED =====
148     def _home(self):
149         if self.timer_job:
150             try:
151                 self.after_cancel(self.timer_job)
152             except Exception:
153                 pass
154             self.timer_job = None
155
156         self._clear()
157
158         bg = tk.Frame(self.container)
159         bg.grid(column=0, row=0, sticky="nsew")
160
161         title_label = tk.Label(bg, text="Title", style="Title.Label", justify="center")
162         title_label.grid(column=0, row=1, sticky="nsew")
163         subtitle_label = tk.Label(bg, text="Subtitle", style="Subtitle.Label", justify="center")
164         subtitle_label.grid(column=0, row=2, sticky="nsew")
165
166         btn_label = tk.Label(bg, text="Button", style="Button.Label", justify="center")
167         btn_label.grid(column=0, row=3, sticky="nsew")
168
169         choice_label = tk.Label(bg, text="Choice", style="Choice.Label", justify="center")
170         choice_label.grid(column=0, row=4, sticky="nsew")

```

```

171
172     # ===== TILED =====
173     def _package_menu(self):
174         self._clear()
175         bg = tk.Frame(self.container)
176         bg.grid(column=0, row=0, sticky="nsew")
177
178         title_label = tk.Label(bg, text="Title", style="Title.Label", justify="center")
179         title_label.grid(column=0, row=1, sticky="nsew")
180
181         subtitle_label = tk.Label(bg, text="Subtitle", style="Subtitle.Label", justify="center")
182         subtitle_label.grid(column=0, row=2, sticky="nsew")
183
184         btn_label = tk.Label(bg, text="Button", style="Button.Label", justify="center")
185         btn_label.grid(column=0, row=3, sticky="nsew")
186
187         choice_label = tk.Label(bg, text="Choice", style="Choice.Label", justify="center")
188         choice_label.grid(column=0, row=4, sticky="nsew")
189
190         # ===== TILED =====
191         self._start_timer(self, package, level)
192         self._package = package
193         self._level = level
194         self._start_timer(self, package, level)

```

[illegible][illegible][illegible][illegible][illegible][illegible]

```

def load_file(path, default=None):
    """Load file"""
    return default

def save_file(path, data):
    """Save file"""
    # Create a temporary file then replace the target file.
    # This is to avoid race conditions.
    tmp = path + ".tmp"
    try:
        with open(tmp, "w", encoding="utf-8") as f:
            f.write(data)
            f.flush()
            os.fsync(f.fileno())
        os.replace(tmp, path)
    except Exception:
        try:
            os.unlink(tmp)
        except:
            pass
    return data

def load_file(path):
    """Load file"""
    return load_file(path, default=None)

```

[illegible]

```

> # assign to guess_jan
def normalize_guess_answer(guess):
    """Returns True if guess is normalized.
    In a word, guess[0] is not '1'
    If a word is not a lowercase letter
    or 'correct_answer' is not a lowercase letter
    then guess is not normalized.
    If 'correct_answer' is not a lowercase letter
    then guess is not normalized.
    If 'correct_answer' is not a lowercase letter
    then guess is not normalized.
    """
    return guess

# ...

def load_data_for_package(history, package):
    """Returns a list of questions for that have been used for the given package in history.
    """
    target = _norm_pkg_name(package)
    used = set()
    for q in history[:10]:
        if _norm_pkg_name(q['package']) == target:
            for aid in q.get('aid', []):
                used.add(aid)
    return used

```

```

def pick_daily_challenge_by_level(lvl, package, level='all', count=5):
    """Deterministic daily selection based on data = package + level.
    Returns up to 'count' items from db matching level.
    """
    level = (level or 'all').strip().lower()
    pool = [q for q in db if (q['package'] == package or q['level'] == level) and q['norm_pkg_name'] != _norm_pkg_name(package)]
    if not pool:
        return []
    today = datetime.date.today().isoformat()
    seed_str = f'{today}_{package}_{level}'
    seed = sha256(seed_str).hexdigest()
    rng = Random.Random(seed)
    selected = rng.sample(pool, min(count, len(pool)))
    rng.shuffle(selected)
    return selected

```

```

def pick_daily_challenge_by_level(lvl, package, level='all', count=5):
    """Deterministic daily selection based on data = package + level.
    Returns up to 'count' items from db matching level.
    """
    level = (level or 'all').strip().lower()
    pool = [q for q in db if (q['package'] == package or q['level'] == level) and q['norm_pkg_name'] != _norm_pkg_name(package)]
    if not pool:
        return []
    today = datetime.date.today().isoformat()
    seed_str = f'{today}_{package}_{level}'
    seed = sha256(seed_str).hexdigest()
    rng = Random.Random(seed)
    selected = rng.sample(pool, min(count, len(pool)))
    rng.shuffle(selected)
    return selected

```

```

def pick_daily_challenge_by_level(lvl, package, level='all', count=5):
    """Deterministic daily selection based on data = package + level.
    Returns up to 'count' items from db matching level.
    """
    level = (level or 'all').strip().lower()
    pool = [q for q in db if (q['package'] == package or q['level'] == level) and q['norm_pkg_name'] != _norm_pkg_name(package)]
    if not pool:
        return []
    today = datetime.date.today().isoformat()
    seed_str = f'{today}_{package}_{level}'
    seed = sha256(seed_str).hexdigest()
    rng = Random.Random(seed)
    selected = rng.sample(pool, min(count, len(pool)))
    rng.shuffle(selected)
    return selected

```

```

def pick_daily_challenge_by_level(lvl, package, level='all', count=5):
    """Deterministic daily selection based on data = package + level.
    Returns up to 'count' items from db matching level.
    """
    level = (level or 'all').strip().lower()
    pool = [q for q in db if (q['package'] == package or q['level'] == level) and q['norm_pkg_name'] != _norm_pkg_name(package)]
    if not pool:
        return []
    today = datetime.date.today().isoformat()
    seed_str = f'{today}_{package}_{level}'
    seed = sha256(seed_str).hexdigest()
    rng = Random.Random(seed)
    selected = rng.sample(pool, min(count, len(pool)))
    rng.shuffle(selected)
    return selected

```

```

def pick_daily_challenge_by_level(lvl, package, level='all', count=5):
    """Deterministic daily selection based on data = package + level.
    Returns up to 'count' items from db matching level.
    """
    level = (level or 'all').strip().lower()
    pool = [q for q in db if (q['package'] == package or q['level'] == level) and q['norm_pkg_name'] != _norm_pkg_name(package)]
    if not pool:
        return []
    today = datetime.date.today().isoformat()
    seed_str = f'{today}_{package}_{level}'
    seed = sha256(seed_str).hexdigest()
    rng = Random.Random(seed)
    selected = rng.sample(pool, min(count, len(pool)))
    rng.shuffle(selected)
    return selected

```

```

def pick_daily_challenge_by_level(lvl, package, level='all', count=5):
    """Deterministic daily selection based on data = package + level.
    Returns up to 'count' items from db matching level.
    """
    level = (level or 'all').strip().lower()
    pool = [q for q in db if (q['package'] == package or q['level'] == level) and q['norm_pkg_name'] != _norm_pkg_name(package)]
    if not pool:
        return []
    today = datetime.date.today().isoformat()
    seed_str = f'{today}_{package}_{level}'
    seed = sha256(seed_str).hexdigest()
    rng = Random.Random(seed)
    selected = rng.sample(pool, min(count, len(pool)))
    rng.shuffle(selected)
    return selected

```

DAFTAR PUSTAKA

pkbm.id. (2024). *Syarat, Ketentuan & Informasi Umum UTBK SNBT 2024*.
<https://pkbm.id/blog/syarat,-ketentuan-&-informasi-umum-utbk-snbt-2024>

Python Software Foundation. (2024). *Python 3 documentation*.
<https://docs.python.org/3/>