

LAPORAN
IMPLEMENTASI APLIKASI TODO LIST DENGAN FLASK DAN MYSQL



Disusun oleh:

Achmad Yoga Alfandi (5230411291)

Alif Nanda Pangestu (5230411306)

Amarullah Arsadinata (5230411323)

Ibra Erlangga Putra (5230411321)

Yogi Pranoto (5230411284)

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA

2024

Pendahuluan

Kode di atas merupakan implementasi aplikasi Todo List menggunakan framework Flask untuk web development dan MySQL sebagai database backend. Aplikasi ini memungkinkan pengguna untuk menambahkan, melihat, dan menghapus tugas (tasks) dengan berbagai fitur tambahan seperti pengaturan prioritas, mata pelajaran, dan status tugas.

Fitur-Fitur Utama

Struktur Kode

1. Import Libraries:

```
1 from flask import Flask, render_template, request, redirect, url_for, flash
2 import mysql.connector
```

- Flask: Digunakan untuk membuat aplikasi web.
- mysql.connector: Digunakan untuk berinteraksi dengan database MySQL.

2. Inisialisasi Aplikasi:

```
1 app = Flask(__name__)
2 app.secret_key = 'your_secret_key' # Change to a random secret key
```

- Membuat instance dari aplikasi Flask dan menetapkan kunci rahasia untuk sesi.

Kelas Database

```
1 class Database:
2     def __init__(self):
3         self.connection = mysql.connector.connect(
4             host="localhost",
5             user="root", # Replace with your MySQL username
6             password="", # Replace with your MySQL password
7             database="todo_db"
8         )
```

- __init__: Menginisialisasi objek Database dan membuat koneksi ke basis data MySQL.
-

Metode

- `get_connection`: Mengembalikan koneksi database.
- `close`: Menutup koneksi database.

Kelas TaskManager

```
1 class TaskManager:
2     def __init__(self):
3         self.db = Database()
```

- Mengelola semua operasi terkait tugas, menggunakan kelas Database.

Metode

1. `fetch_tasks`:

```
1 def fetch_tasks(self):
2     db = self.db.get_connection()
3     cursor = db.cursor()
4     cursor.execute("""
5         SELECT t.id, t.task_name, t.deadline, p.priority_name, s.subject_name, st.status_name
6         FROM tasks t
7         JOIN priorities p ON t.priority_id = p.id
8         JOIN subjects s ON t.subject_id = s.id
9         JOIN statuses st ON t.status_id = st.id
10    """)
11    tasks = cursor.fetchall()
12    cursor.close()
13    return tasks
```

- Mengambil semua tugas dari database dengan melakukan join pada tabel terkait (priorities, subjects, statuses) dan mengembalikan daftar tugas.

2. `add_task`:

```
1 def add_task(self, task_name, task_deadline, task_priority, task_subject):
```

...

- Menambahkan tugas baru ke database. Metode ini juga menangani penambahan subjek baru jika belum ada di tabel subjects.

3. delete_tasks:

```
1 def delete_tasks(self, task_ids):
```

...

- Menghapus tugas berdasarkan ID yang dipilih.

4. delete_all_tasks:

```
1 def delete_all_tasks(self):
```

...

- Menghapus semua tugas dari tabel tasks.

Rute Aplikasi

1. @app.route('/'):

def index():

```
1 def index():
2     task_manager = TaskManager()
3     tasks = task_manager.fetch_tasks()
4     return render_template('index.html', tasks=tasks)
```

- Menampilkan halaman utama (index.html) dengan daftar tugas yang diambil dari database.

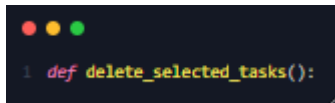
2. @app.route('/add', methods=['POST']):

```
1 def add_task():
```

...

- Menangani permintaan untuk menambahkan tugas baru. Jika semua field terisi, tugas akan ditambahkan; jika tidak, akan menampilkan pesan kesalahan.

3. @app.route('/delete-selected', methods=['POST']):

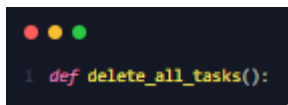


```
1 def delete_selected_tasks():
```

...

- Menghapus tugas yang dipilih oleh pengguna. Jika tidak ada tugas yang dipilih, akan menampilkan pesan kesalahan.

4. @app.route('/delete-all', methods=['POST']):



```
1 def delete_all_tasks():
```

...

- Menghapus semua tugas dari database dan memberi tahu pengguna tentang tindakan tersebut.

Menjalankan Aplikasi

- Aplikasi dapat dijalankan dengan menggunakan perintah python app.py di terminal.
- Akses aplikasi melalui browser di <http://127.0.0.1:5000/>.

Kesimpulan

Aplikasi ini merupakan contoh dasar manajemen tugas menggunakan Flask dan MySQL. Kode ini terstruktur dengan baik, memisahkan logika database dan manajemen tugas ke dalam kelas yang berbeda, sehingga memudahkan pemeliharaan dan pengembangan lebih lanjut. Implementasi metode CRUD (Create, Read, Update, Delete) untuk tugas memberikan fungsionalitas yang diperlukan untuk aplikasi manajemen tugas yang sederhana.

Penjelasan Kode Program index.html

1. Judul Program: To-Do List

Deskripsi Umum:

Program ini adalah aplikasi web sederhana untuk mengelola daftar tugas harian. Pengguna dapat menambahkan tugas, menghapus tugas yang dipilih, atau menghapus semua tugas.

2. Struktur Dasar HTML

2.1 Struktur Dasar Dokumen HTML

Kode diawali dengan deklarasi dokumen `<!DOCTYPE html>` dan elemen dasar seperti:

- `<html>`: Menentukan dokumen sebagai HTML.
- `<head>`: Berisi metadata seperti karakter encoding, viewport, judul, dan tautan ke pustaka CSS eksternal.
- `<body>`: Berisi konten utama, termasuk formulir dan tabel.

2.2 Meta Tag dan Eksternal CSS

- `meta charset="UTF-8"`: Mengatur karakter encoding ke UTF-8.
- `meta name="viewport"`: Memastikan tampilan responsif di perangkat mobile.
- `link rel="stylesheet"`: Mengimpor ikon dari Font Awesome untuk penggunaan simbol dan ikon tambahan.

3. CSS Internal Styling

3.1 Gaya Umum Halaman

- `Body`: Latar belakang berwarna biru muda, teks tengah, dan tata letak fleksibel.
- `h1`: Judul halaman dengan warna biru tua.
- `Formulir`: Diberi latar belakang putih, bayangan, padding, dan border radius.

3.2 Input dan Tombol

- `Input dan tombol` memiliki gaya konsisten (padding, border-radius, warna).
- `Tombol` memiliki efek hover untuk meningkatkan interaktivitas.

3.3 Tabel

- `Tabel` digunakan untuk menampilkan daftar tugas.
- `Header tabel (<th>)` memiliki warna latar biru dan teks putih.
- `Baris tabel` bergantian warna dan memiliki efek hover.

4. Formulir Pertama: Menambahkan Tugas

```
1 <form method="POST" action="/add">
2   <input type="text" name="task_name" placeholder="Nama Tugas" required>
3   <input type="text" name="task_deadline" placeholder="Tenggat Waktu (dd-mm-yyyy)" required>
4   <select name="task_priority" required>
5     <option value="" disabled selected>Prioritas</option>
6     <option value="Tinggi">Tinggi</option>
7     <option value="Sedang">Sedang</option>
8     <option value="Rendah">Rendah</option>
9   </select>
10  <input type="text" name="task_subject" placeholder="Mata Kuliah" required>
11  <button type="submit">Tambah Tugas</button>
12 </form>
```

Penjelasan:

- Formulir ini memungkinkan pengguna untuk menambahkan tugas baru.
- Input termasuk: Nama Tugas, Tenggat Waktu, Prioritas, dan Mata Kuliah.
- Tombol "Tambah Tugas" mengirim data ke server melalui metode POST dengan rute /add.

5. Formulir Kedua: Menghapus Tugas Terpilih

```
1 <form method="POST" action="/delete-selected">
2     <table>
3         <thead>
4             <tr>
5                 <th>Pilih</th>
6                 <th>Nama Tugas</th>
7                 <th>Tanggal Waktu</th>
8                 <th>Prioritas</th>
9                 <th>Mata Kuliah</th>
10                <th>Status</th>
11            </tr>
12        </thead>
13        <tbody>
14            {% for task in tasks %}
15            <tr>
16                <td><input type="checkbox" name="task_ids" value="{{ task.id }}"></td>
17                <td>{{ task[1] }}</td>
18                <td>{{ task[2] }}</td>
19                <td>{{ task[3] }}</td>
20                <td>{{ task[4] }}</td>
21                <td>{{ task[5] }}</td>
22            </tr>
23            {% endfor %}
24        </tbody>
25    </table>
26    <div>
27        <button type="submit">Hapus Tugas Terpilih</button>
28    </div>
29 </form>
```

Penjelasan:

- Tabel menampilkan daftar tugas menggunakan Jinja2 template engine.
- Setiap tugas memiliki checkbox untuk memilih tugas yang akan dihapus.
- Tombol "Hapus Tugas Terpilih" mengirimkan data tugas yang dipilih ke server dengan rute /delete-selected.

6. Formulir Ketiga: Menghapus Semua Tugas

```
1 <form method="POST" action="/delete-all">
2     <button type="submit">Hapus Semua Tugas</button>
3 </form>
```

Penjelasan:

- Formulir ini memungkinkan pengguna untuk menghapus semua tugas yang ada dalam daftar.
- Data dikirim menggunakan metode POST dengan rute /delete-all.

7. Jinja2 Template Engine

Kode menggunakan blok `{% for task in tasks %}` untuk merender daftar tugas yang diterima dari backend.

- `task[0]`: ID Tugas
- `task[1]`: Nama Tugas
- `task[2]`: Tenggat Waktu
- `task[3]`: Prioritas
- `task[4]`: Mata Kuliah
- `task[5]`: Status

8. Alur Kerja Aplikasi

1. Menambahkan Tugas:

- Pengguna mengisi formulir pertama dan menekan tombol "Tambah Tugas".
- Data dikirim ke server (`/add`) untuk disimpan dalam database.

2. Menampilkan Tugas:

- Server mengirimkan daftar tugas ke halaman dan merendernya menggunakan tabel.

3. Menghapus Tugas Terpilih:

- Pengguna mencentang checkbox dan menekan tombol "Hapus Tugas Terpilih".
- Data ID tugas yang dipilih dikirim ke server (`/delete-selected`) untuk dihapus.

4. Menghapus Semua Tugas:

- Pengguna menekan tombol "Hapus Semua Tugas".
- Server menghapus semua tugas dari database.

Kesimpulan

Kode ini membangun aplikasi To-Do List sederhana dengan fungsi berikut:

- Menambahkan tugas.
- Menampilkan tugas dalam tabel.
- Menghapus tugas berdasarkan pilihan pengguna.
- Menghapus semua tugas sekaligus.

Bagian Database:

1. Tabel Subject

					id	subject_name
<input type="checkbox"/>		Edit		Copy		Delete
					1	pbo
<input type="checkbox"/>		Edit		Copy		Delete
					2	PSD

□ Kolom Header:

- **id:** Merujuk pada kolom pertama yang berisi nilai unik (primary key) untuk setiap entri. Nilai ini biasanya digunakan untuk mengidentifikasi data secara unik di dalam tabel.
- **subject_name:** Kolom kedua yang berisi nama subjek atau kategori tertentu.

□ Baris Data:

- **Baris 1:**
 - **id** = 1
 - **subject_name** = "pbo" (kemungkinan singkatan dari "Pemrograman Berorientasi Objek").
- **Baris 2:**
 - **id** = 2
 - **subject_name** = "PSD" (kemungkinan singkatan dari sesuatu seperti "Pengantar Sistem Digital").

2. Tabel Task

← T →						id	task_name	deadline	priority_id	subject_id	status_id	
<input type="checkbox"/>		Edit		Copy		Delete	1	pbo tugas besar	01-12-2024	NULL	NULL	NULL
<input type="checkbox"/>		Edit		Copy		Delete	3	Pembuatan Tubes	19-12-2024	1	2	1
<input type="checkbox"/>		Edit		Copy		Delete	4	Pembuatan Tubes	19-12-2024	1	2	1

1. Kolom Header:

- **id:** Kolom ini adalah identifier unik (primary key) untuk setiap baris dalam tabel.
- **task_name:** Berisi nama tugas atau deskripsi pekerjaan.
- **deadline:** Menunjukkan batas waktu penyelesaian tugas (format tanggal: DD-MM-YYYY).
- **priority_id:** Berisi tingkat prioritas tugas, yang kemungkinan dihubungkan dengan tabel lain yang menjelaskan nilai prioritas (contoh: 1 = prioritas tinggi).
- **subject_id:** Menunjukkan identitas subjek terkait tugas, yang merujuk ke tabel "subject" (seperti pada gambar sebelumnya).
- **status_id:** Menandakan status tugas, yang mungkin terhubung dengan tabel lain yang mendefinisikan status (contoh: 1 = selesai, 2 = belum selesai).

2. Data Tabel:

- **Baris 1:**
 - **id:** 1
 - **task_name:** "pbo tugas besar"
 - **deadline:** 01-12-2024
 - **priority_id:** NULL (tidak ada prioritas yang ditentukan)
 - **subject_id:** NULL (tidak terhubung ke subjek tertentu)
 - **status_id:** NULL (status belum didefinisikan)
- **Baris 2:**
 - **id:** 3
 - **task_name:** "Pembuatan Tubes"
 - **deadline:** 19-12-2024
 - **priority_id:** 1 (kemungkinan prioritas tinggi)
 - **subject_id:** 2 (mengacu pada subjek dengan id 2, yaitu "PSD" dari tabel sebelumnya)
 - **status_id:** 1 (kemungkinan status selesai)
- **Baris 3:**
 - **id:** 4
 - **task_name:** "Pembuatan Tubes"
 - **deadline:** 19-12-2024
 - **priority_id:** 1 (prioritas tinggi)
 - **subject_id:** 2 (mengacu pada subjek "PSD")
 - **status_id:** 2 (kemungkinan status belum selesai)

3. Observasi Tambahan:

- Tugas "Pembuatan Tubes" muncul dua kali (baris 2 dan 3) dengan detail yang sama, kecuali pada kolom **status_id**.
- Kolom **priority_id**, **subject_id**, dan **status_id** tampaknya berfungsi sebagai foreign key yang terhubung dengan tabel lain untuk memberikan informasi lebih spesifik.

3. Tabel Priorities



	id	priority_name
<input type="checkbox"/>	1	Tinggi
<input type="checkbox"/> Edit Copy Delete	2	Sedang
<input type="checkbox"/> Edit Copy Delete	3	Rendah

1. Kolom Header:

- **id:** Kolom ini berfungsi sebagai identifier unik (primary key) untuk setiap baris dalam tabel.
- **priority_name:** Berisi nama atau deskripsi tingkat prioritas.

2. Data Tabel:

- **Baris 1:**
 - **id:** 1
 - **priority_name:** "Tinggi" (prioritas tinggi).
- **Baris 2:**
 - **id:** 2
 - **priority_name:** "Sedang" (prioritas sedang).
- **Baris 3:**
 - **id:** 3
 - **priority_name:** "Rendah" (prioritas rendah).

3. Observasi Tambahan:

- Kolom ini mungkin digunakan sebagai referensi (foreign key) dalam tabel lain, seperti pada kolom **priority_id** dalam tabel sebelumnya, yang merujuk pada nilai di tabel ini untuk menentukan tingkat prioritas suatu tugas.
- Hierarki prioritas dalam tabel ini adalah:
 - **1:** Tinggi
 - **2:** Sedang
 - **3:** Rendah

4. Tabel Statuses



	id	status_name
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	Belum Selesai
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	Selesai

1. Kolom id

- **id** adalah kolom yang biasanya digunakan sebagai primary key. Primary key berfungsi sebagai pengidentifikasi unik untuk setiap baris data dalam tabel.
- Dalam gambar ini, terdapat dua nilai dalam kolom **id**:
 - Baris pertama.
 - Baris kedua.

2. Kolom status_name

- **status_name** adalah kolom yang berisi deskripsi atau nama status.
- Terdapat dua entri dalam kolom **status_name**:
 - **Belum Selesai:** Ini menunjukkan status yang belum selesai.
 - **Selesai:** Ini menunjukkan status yang sudah selesai.

Pada setiap baris data, terdapat beberapa tombol aksi untuk mengelola data:

- **Edit:** Untuk mengedit data pada baris tertentu.
- **Copy:** Untuk menduplikasi data pada baris tersebut.
- **Delete:** Untuk menghapus data dari baris yang bersangkutan.

3. Konteks Penggunaan

- Tabel ini mungkin digunakan untuk mencatat status suatu tugas, aktivitas, atau proses. Sebagai contoh:
 - Status **Belum Selesai** untuk tugas yang sedang dikerjakan.
 - Status **Selesai** untuk tugas yang telah diselesaikan.

5. DataBase `todo_db`

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> <code>priorities</code>		3	InnoDB	utf8mb4_0900_ai_ci	32.0 KiB	-
<input type="checkbox"/> <code>statuses</code>		2	InnoDB	utf8mb4_0900_ai_ci	32.0 KiB	-
<input type="checkbox"/> <code>subjects</code>		2	InnoDB	utf8mb4_0900_ai_ci	32.0 KiB	-
<input type="checkbox"/> <code>tasks</code>		3	InnoDB	utf8mb4_0900_ai_ci	64.0 KiB	-
4 tables	Sum	10	InnoDB	utf8mb4_0900_ai_ci	160.0 KiB	0 B

1. Kolom Tabel (Table)

- Gambar ini menunjukkan bahwa terdapat empat tabel dalam database, yaitu:
 - **priorities**: Berisi informasi prioritas.
 - **statuses**: Berisi informasi status.
 - **subjects**: Berisi informasi subjek atau topik.
 - **tasks**: Berisi informasi tugas atau pekerjaan.

2. Kolom Aksi (Action)

- Kolom ini menyediakan opsi untuk mengelola setiap tabel, seperti:
 - **Browse**: Melihat data dalam tabel.
 - **Structure**: Melihat atau mengedit struktur tabel (kolom, tipe data, dll.).
 - **Search**: Melakukan pencarian data dalam tabel.
 - **Insert**: Menambahkan data baru ke dalam tabel.
 - **Empty**: Menghapus semua data dalam tabel tanpa menghapus tabelnya.
 - **Drop**: Menghapus tabel secara permanen dari database.

3. Kolom Baris (Rows)

- Menunjukkan jumlah baris (records) dalam setiap tabel:
 - **priorities** memiliki 3 baris.
 - **statuses** memiliki 2 baris.
 - **subjects** memiliki 2 baris.
 - **tasks** memiliki 3 baris.

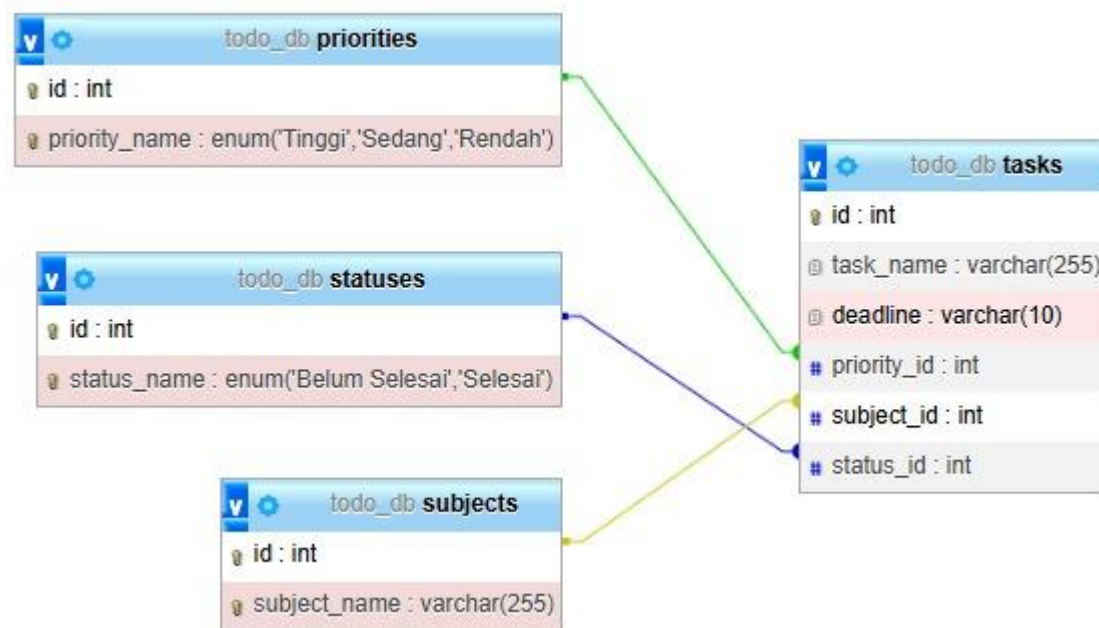
4. Kolom Tipe (Type)

- Semua tabel menggunakan tipe penyimpanan **InnoDB**, yang mendukung fitur seperti transaksi dan kunci asing.

Kesimpulan

Database ini dirancang untuk mengelola tugas (tasks) yang dapat memiliki atribut seperti prioritas (priorities), status (statuses), dan subjek (subjects). Struktur ini cukup umum dalam aplikasi manajemen proyek atau sistem pelacakan tugas.

6.Designer/UML



1. Tabel Prioritas

- **Kolom:**
 - **id**: Primary key, bertipe **integer**.
 - **priority_name**: Bertipe **ENUM** dengan nilai:
 - **Tinggi**
 - **Sedang**
 - **Rendah**
- **Fungsi:**
 - Menyimpan informasi tentang prioritas tugas (high, medium, low).
- **Relasi:**
 - Terhubung ke tabel tasks melalui kolom **priority_id**.

2. Tabel Statues

- **Kolom:**
 - **id**: Primary key, bertipe **integer**.
 - **status_name**: Bertipe **ENUM** dengan nilai:
 - **Belum Selesai**
 - **Selesai**
- **Fungsi:**
 - Menyimpan informasi status tugas (completed or not completed).
- **Relasi:**
 - Terhubung ke tabel tasks melalui kolom **status_id**.

3. Tabel subjects

- **Kolom:**
 - **id: Primary key**, bertipe **integer**.
 - **subject_name**: Bertipe **VARCHAR(255)** (panjang maksimum 255 karakter).
- **Fungsi:**
 - Menyimpan informasi subjek atau kategori tugas (contoh: matematika, pekerjaan, dll.).
- **Relasi:**
 - Terhubung ke tabel tasks melalui kolom **subject_id**.

4. Tabel tasks

- **Kolom:**
 - **id: Primary key**, bertipe **integer**.
 - **task_name**: Nama tugas, bertipe **VARCHAR(255)**.
 - **deadline**: Tanggal batas waktu tugas, bertipe **VARCHAR(10)** (biasanya format tanggal seperti **YYYY-MM-DD**).
 - **priority_id**: Foreign key yang terhubung ke tabel **priorities**.
 - **subject_id**: Foreign key yang terhubung ke tabel **subjects**.
 - **status_id**: Foreign key yang terhubung ke tabel **statuses**.
- **Fungsi:**
 - Menyimpan detail tugas, termasuk nama, batas waktu, prioritas, subjek, dan status.

Relasi Antar Tabel

1. **Relasi Prioritas ke Tugas (priorities → tasks):**
 - Setiap tugas memiliki satu prioritas.
 - Relasi: Satu ke banyak (One-to-Many).
2. **Relasi Status ke Tugas (statuses → tasks):**
 - Setiap tugas memiliki satu status.
 - Relasi: Satu ke banyak (One-to-Many).
3. **Relasi Subjek ke Tugas (subjects → tasks):**
 - Setiap tugas memiliki satu subjek.
 - Relasi: Satu ke banyak (One-to-Many).

Kesimpulan

Struktur ini adalah implementasi database yang dirancang untuk aplikasi manajemen tugas. Tabel tasks berfungsi sebagai tabel utama, yang bergantung pada tabel referensi lainnya (priorities, statuses, dan subjects) untuk menentukan detail terkait tugas.