BAB IV

IMPLEMENTASI DAN HASIL

1.1 Implementasi Sistem

Tahap implementasi adalah proses mengubah rancangan sistem menjadi aplikasi nyata. Implementasi dilakukan pada komputer/laptop dengan spesifikasi:

Tabel 4.1 Spesifikasi

No	Komponen	Spesifikasi
1	Sistem operasi	Windows 10 64-bit
2	Prosesor	Intel Core i3 atau setara
3	RAM	Minimal 4 GB
4	Penyimpanan	Minimal 500 MB
5	Bahasa pemrograman	Python 3.x
6	Framework Backend	Flask
7	Model Deteksi	YOLOv8 (Ultralytics)
8	Basis Data	SQLite 3
9	Library Pendukung	OpenCV,Torch,Ultralytics, OS, Datetime, SQLite

1.1.1 Struktur Folder Aplikasi

Struktur direktori aplikasi meliputi:

- /backend/ → folder utama backend
- app.py → script server Flask
- detect.py → modul deteksi YOLOv8
- model.pt → file model YOLOv8 terlatih
- templates/ → HTML (index, dashboard, data)
- static/ → CSS, gambar, hasil deteksi

Dengan struktur ini, pengembangan lebih terorganisir.

1.1.2 Implementasi Backend

Framework Flask dipilih karena ringan dan mudah digunakan. app.py berfungsi untuk:

1. Routing URL
2. Proses upload gambar
3. Pemanggilan fungsi deteksi
4. Simpan hasil ke database

Contoh cuplikan kode:

```python
@app.route('/deteksi', methods=['POST'])
def deteksi():
    file = request.files['file']
    # Proses deteksi menggunakan YOLO
    result = deteksi_kerusakan(file)
    return render_template('hasil.html', result=result)
```

1.1.3 Implementasi Deteksi YOLOv8

Model YOLOv8 dilatih secara terpisah menggunakan Google Colab. Hasil training disimpan dalam file best.pt sebagai arsip model terlatih. Namun, pada prototipe ini, fungsi detect.py belum memuat model YOLOv8 secara live, melainkan masih menggunakan simulasi bounding box dan label penyebab kerusakan untuk menggambarkan alur kerja deteksi

Contoh cuplikan:

```python
model = YOLO('model.pt')
results = model.predict(source=image_path, save=True)
```

1.1.4 Implementasi Basis Data SQLite

Aplikasi menggunakan SQLite sebagai database ringan untuk menyimpan histori pendeteksian:

- Nama file gambar
- Tanggal deteksi

- Jenis kerusakan

Contoh tabel:

Tabel 4.2 Riwayat Deteksi

Id	Nama file	Tanggal	Hasil deteksi
1	kardus1.jpg	07/07/2025	Kerusakan akibat gesekan

1.2 Tampilan Antarmuka

Antarmuka web dibangun menggunakan HTML, CSS, dan Bootstrap agar responsif.

Berikut beberapa tampilan:

1.2.1 Halaman Beranda / Landing Page

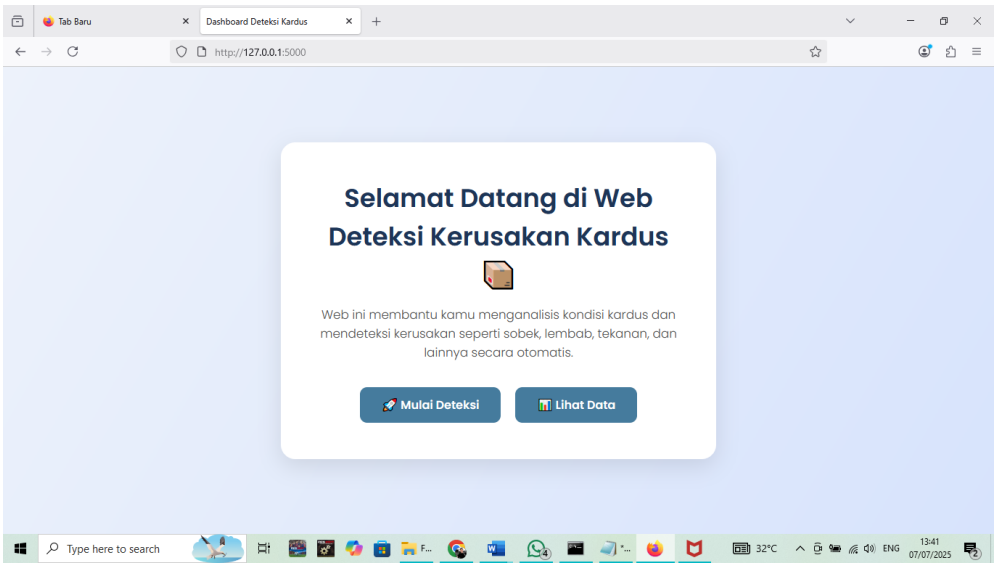
Halaman beranda merupakan tampilan pertama yang diakses pengguna saat membuka aplikasi deteksi kerusakan kardus. Pada halaman ini, ditampilkan sambutan berupa judul *"*Selamat Datang di Web Deteksi Kerusakan Kardus*"* yang menjelaskan tujuan aplikasi secara singkat.

Teks penjelasan di bawah judul menjelaskan manfaat aplikasi yaitu membantu pengguna menganalisis kondisi kardus dan mendeteksi kerusakan seperti sobek, lembab, tekanan, dan kerusakan fisik lainnya secara otomatis. Halaman ini juga menampilkan ikon kardus yang mendukung kesan visual lebih ramah pengguna.

Di bagian bawah terdapat dua tombol navigasi yaitu:

1. **Tombol "Mulai Deteksi"** untuk membuka halaman upload gambar kardus.
2. **Tombol "Lihat Data"** untuk mengakses dashboard riwayat pendeteksian.

Dengan desain antarmuka beranda yang sederhana dan informatif, diharapkan pengguna dapat memahami fungsi utama aplikasi secara cepat tanpa kebingungan. Halaman ini juga mendukung UX (*User Experience*) yang lebih baik karena mempermudah navigasi ke fitur utama.



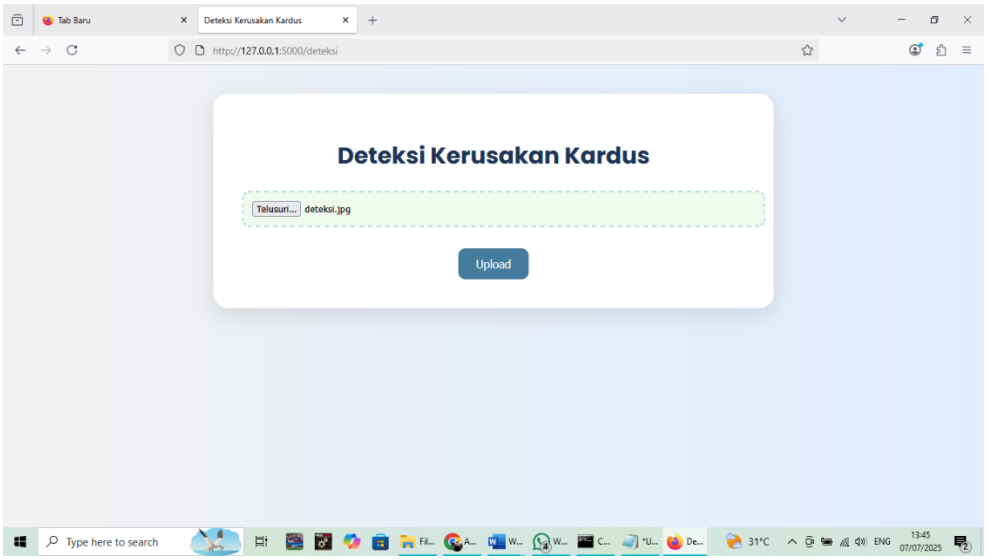
Gambar 4.1 Halaman Beranda

1.2.2 Halaman Upload Gambar

Halaman upload gambar merupakan halaman pertama yang akan diakses oleh pengguna. Pada halaman ini, pengguna dapat memilih file gambar kardus yang ingin dideteksi kerusakannya. Komponen utama yang ditampilkan yaitu **input file**, tombol *Upload*, dan informasi singkat mengenai format file yang diperbolehkan.

Sistem akan memvalidasi jenis file yang diunggah agar hanya mendukung format gambar seperti `.jpg`, `.jpeg`, atau `.png`. Hal ini bertujuan untuk mencegah error pada saat proses pendeteksian. Setelah file dipilih, pengguna cukup menekan tombol *Upload* atau *Deteksi* maka sistem secara otomatis akan memproses gambar menggunakan model *YOLOv8* yang sudah dilatih sebelumnya.

Dengan desain yang sederhana dan user-friendly, halaman upload gambar diharapkan mempermudah pengguna dari berbagai latar belakang untuk menggunakan aplikasi ini tanpa harus memiliki pengetahuan teknis yang mendalam. Fitur validasi file juga membantu menjaga keamanan dan stabilitas aplikasi.



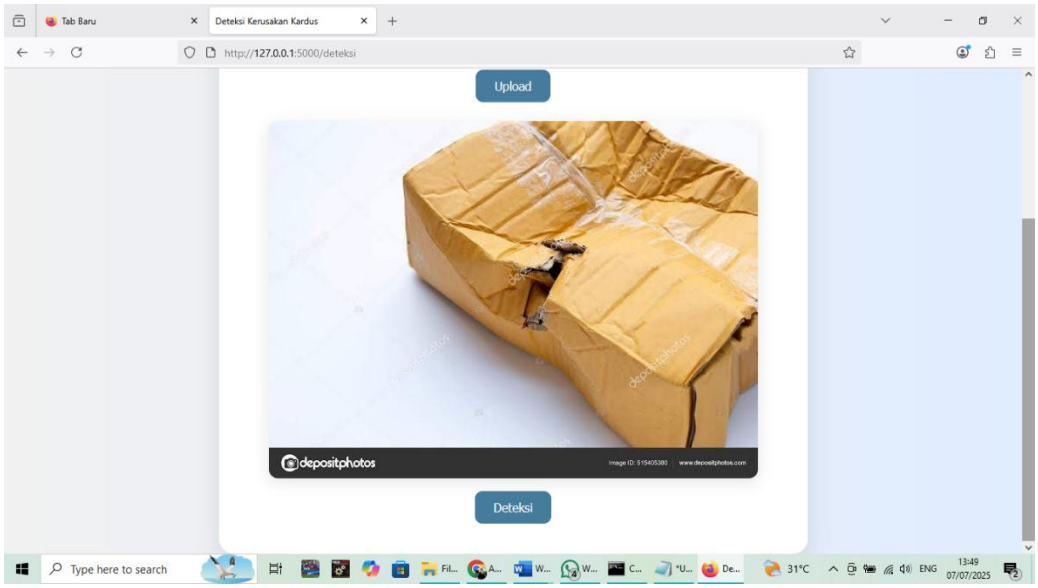
Gambar 4.2 Halaman Upload Gambar

1.2.3 Halaman Proses Deteksi

Setelah pengguna menekan tombol *Upload*, sistem akan berpindah ke halaman proses deteksi. Pada tahap ini, gambar yang diunggah ditampilkan ulang kepada pengguna sebagai konfirmasi bahwa file telah diterima oleh server.

Proses deteksi dilakukan di server Flask dengan memanggil model *YOLOv8*. Proses ini berjalan secara real-time, sehingga pengguna tidak perlu menunggu lama untuk melihat hasilnya. Selama proses berlangsung, sistem dapat menampilkan notifikasi status seperti *"Gambar sedang diproses"* untuk memberi umpan balik kepada pengguna.

Halaman ini juga berfungsi sebagai penghubung antara input pengguna dengan hasil deteksi, sehingga pengguna dapat melihat urutan proses deteksi dengan jelas.



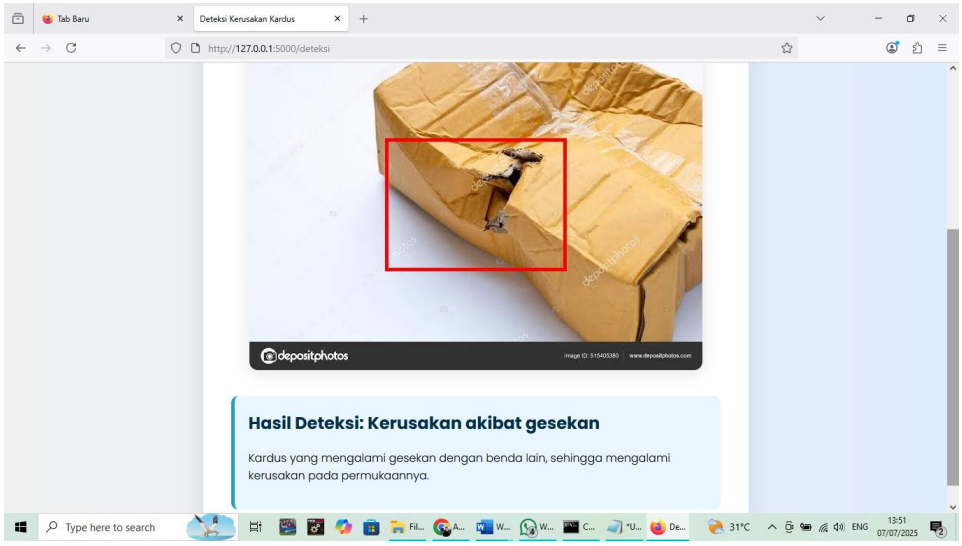
Gambar 4.3 Halaman Proses Deteksi

1.2.4 Hasil Deteksi Kerusakan Kardus

Hasil deteksi ditampilkan di halaman berikutnya setelah proses analisis gambar selesai dilakukan oleh model *YOLOv8*. Pada tampilan ini, gambar kardus akan diperlihatkan kembali dengan ***bounding box*** yang menandai area kerusakan.

Sistem menampilkan hasil simulasi berupa area kerusakan (*bounding box*) dan label penyebab kerusakan. Nilai *confidence score* hanya dihasilkan pada tahap pengujian model di Google Colab, namun tidak ditampilkan di antarmuka prototipe.

Tampilan hasil deteksi ini diharapkan mampu meminimalisir kesalahan manusia dalam mendeteksi kerusakan kardus secara manual, sehingga proses pengecekan kardus menjadi lebih cepat dan akurat.



Gambar 4.4 Halaman Hasil Deteksi

1.2.5 Dashboard Riwayat Deteksi

Selain menampilkan hasil deteksi secara real-time, sistem juga menyediakan halaman **Dashboard Riwayat Deteksi**. Pada halaman ini, seluruh hasil deteksi yang telah dilakukan sebelumnya akan disimpan ke basis data SQLite dan ditampilkan dalam bentuk daftar.

Pengguna dapat melihat detail setiap deteksi, seperti nama file gambar, tanggal deteksi, jenis kerusakan, dan tautan untuk melihat hasil gambar deteksi. Dashboard ini sangat bermanfaat untuk memantau histori pengecekan kardus,

sehingga pengguna memiliki data pendukung untuk pengambilan keputusan lebih lanjut, seperti menolak kardus yang rusak parah atau memisahkan kardus yang masih layak pakai.

Dashboard dirancang dengan tampilan tabel yang mudah dibaca, serta tombol navigasi jika riwayat deteksi sudah banyak. Dengan demikian, pengguna dapat mencari hasil deteksi tertentu dengan cepat.

Gambar 4.5 Halaman Riwayat Deteksi

Berdasarkan hasil implementasi yang telah dilakukan, seluruh komponen utama sistem mulai dari proses unggah gambar, simulasi deteksi, hingga penyajian hasil dan riwayat pendeteksian telah berhasil dijalankan sesuai dengan rancangan. Meskipun proses deteksi masih bersifat simulasi karena keterbatasan sumber daya lokal, prototipe aplikasi telah merepresentasikan alur kerja yang lengkap dan fungsional. Dengan demikian, sistem yang dibangun telah memenuhi tujuan awal penelitian dan dapat dijadikan dasar untuk pengembangan lebih lanjut di masa mendatang