**LAPORAN PROJECT**

**PEMROGRAMAN VISUAL**

**SADAR KULIT**

****

**Disusun Oleh :**

**MUHAMMAD RIZKI ASSAMSULI**

**F1D022146**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS MATARAM**

**2025**

1. Deskripsi Aplikasi

SadarKulit aplikasi desktop yang dirancang untuk meningkatkan kesadaran masyarakat terhadap kesehatan kulit. Aplikasi ini memungkinkan pengguna untuk melakukan deteksi dini terhadap beberapa jenis penyakit kulit hanya dengan mengunggah foto. Tujuannya adalah untuk memberikan informasi awal dan edukasi, bukan untuk menggantikan diagnosis medis profesional.

Proyek ini dilatarbelakangi oleh minimnya pengetahuan masyarakat dalam mengenali gejala awal penyakit kulit, yang seringkali menghambat penanganan yang tepat. Dengan memanfaatkan teknologi Machine Learning, SadarKulit menyediakan alat bantu yang mudah diakses untuk identifikasi awal, lengkap dengan informasi mengenai penyebab dan saran penanganan dasar.

1. Stack List

Front-End:

* Python (PyQt5)

Back-End:

* Node.js
* Express.js

Database:

* MongoDB

1. Flow Aplikasi

|  |
| --- |
|  |

1. Code

dashboard\_widget.py

|  |
| --- |
| # Load History      def load\_history(self):          try:              response = requests.get(f"{API\_URL}/history", headers=self.api\_headers)              response.raise\_for\_status()              self.history\_data = response.json()              print("=== DATA YANG DITERIMA DARI BACKEND ===")              for item in self.history\_data:                  print(item)              print("========================================")              self.populate\_table()          except requests.exceptions.RequestException as e:              QMessageBox.critical(self, 'Error', f"Gagal memuat riwayat:\n{e}")  . . . . . |

|  |
| --- |
| def perform\_prediction(self, file\_path):          loading\_msg = QMessageBox()          loading\_msg.setWindowTitle("Info")          loading\_msg.setText("Sedang mengirim gambar dan melakukan prediksi...")          loading\_msg.setStandardButtons(QMessageBox.NoButton)          loading\_msg.show()          QApplication.processEvents()          try:              mime\_type, \_ = mimetypes.guess\_type(file\_path)              if not mime\_type:                  mime\_type = 'application/octet-stream'              with open(file\_path, 'rb') as image\_file:                  files = {                      'image': (os.path.basename(file\_path), image\_file, mime\_type)                  }                  response = requests.post(                      f"{API\_URL}/predict",                      headers=self.api\_headers,                      files=files                  )                  response.raise\_for\_status()                  result = response.json()                  loading\_msg.close()                  disease\_name = (                      result.get('detectedDisease') or                      result.get('predicted\_disease') or                      'Tidak diketahui'                  )                  disease\_name = bersihkan\_nama\_penyakit(disease\_name)                  confidence = result.get('confidence', 'N/A')                  QMessageBox.information(                      self, "Hasil Prediksi",                      f"Penyakit yang terdeteksi:\n\n{disease\_name}\n\nTingkat Keyakinan: {confidence}"                  )                  self.load\_history()  . . . . . |

Login.py

|  |
| --- |
| class LoginWidget(QWidget):      login\_successful = pyqtSignal(str, dict)      def \_\_init\_\_(self, parent=None):          super().\_\_init\_\_(parent)          loadUi("ui/login\_widget.ui", self)  # Pastikan path benar           # Tampilkan logo dari file          pixmap = QPixmap("assets/images/logo.png")          self.logoLabel.setPixmap(pixmap.scaled(200, 200, aspectRatioMode=1))  # 100x100 dan preserve rasio          self.logoLabel.setStyleSheet("margin-bottom: 10px;")          # Event binding          self.loginButton.clicked.connect(self.attempt\_login)          self.passwordInput.returnPressed.connect(self.attempt\_login)      def attempt\_login(self):          email = self.emailInput.text().strip()          password = self.passwordInput.text().strip()          if not email or not password:              QMessageBox.warning(self, 'Input Kosong', 'Email dan password tidak boleh kosong.')              return          try:              response = requests.post(                  f"{API\_URL}/auth/login",                  json={'email': email, 'password': password}              )              response.raise\_for\_status()              data = response.json()              token = data.get('token')              user\_data = data.get('user')              if token and user\_data:                  self.login\_successful.emit(token, user\_data)              else:                  QMessageBox.warning(self, 'Login Gagal', 'Respons dari server tidak valid.')          except requests.exceptions.RequestException as e:              error\_message = "Email atau password salah."              if e.response:                  try:                      backend\_error = e.response.json().get('error')                      if backend\_error:                          error\_message = backend\_error                  except Exception:                      pass              else:                  error\_message = f"Gagal terhubung ke server.\n\n{e}"              QMessageBox.critical(self, 'Login Error', error\_message) |

1. Hasil Aplikasi

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |