

**LAPORAN TUGAS AHIR**  
**Mata Kuliah Pemrograman Berorientasi Objek**

**Judul Projek**

**KARTU TANDA MAHASISWA**



**Disusun Oleh:**

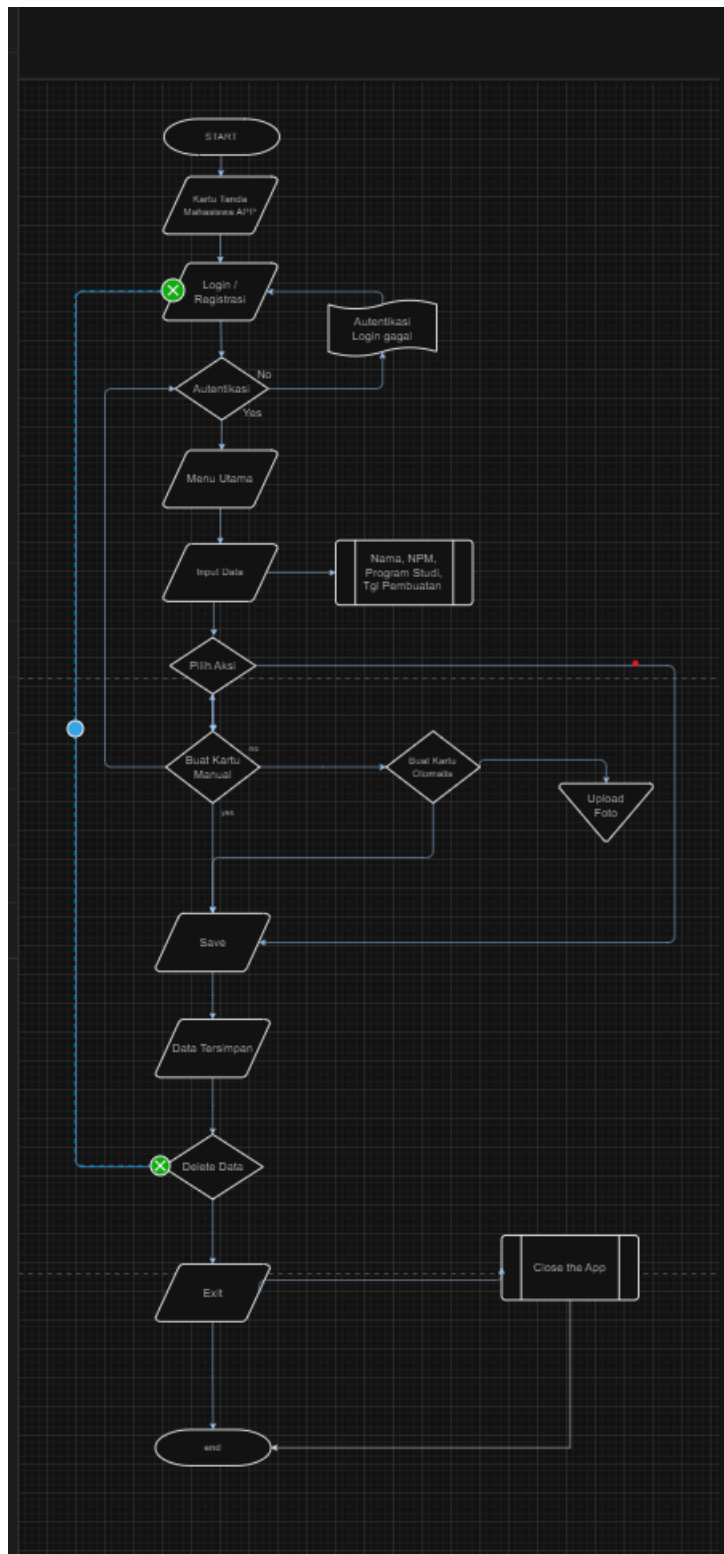
- 1. Elisabeth Dahu Seran (2213020021)**
- 2. Nadika Dimas (2213020167)**
- 3. Ilma Ma'rifatul Qur'ana (2213020195)**

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**FAKULTAS TEKNIK DAN ILMU KOMPUTER**  
**UNIVERSITAS NUSANTARA PGRI KEDIRI**  
**TAHUN 2023**

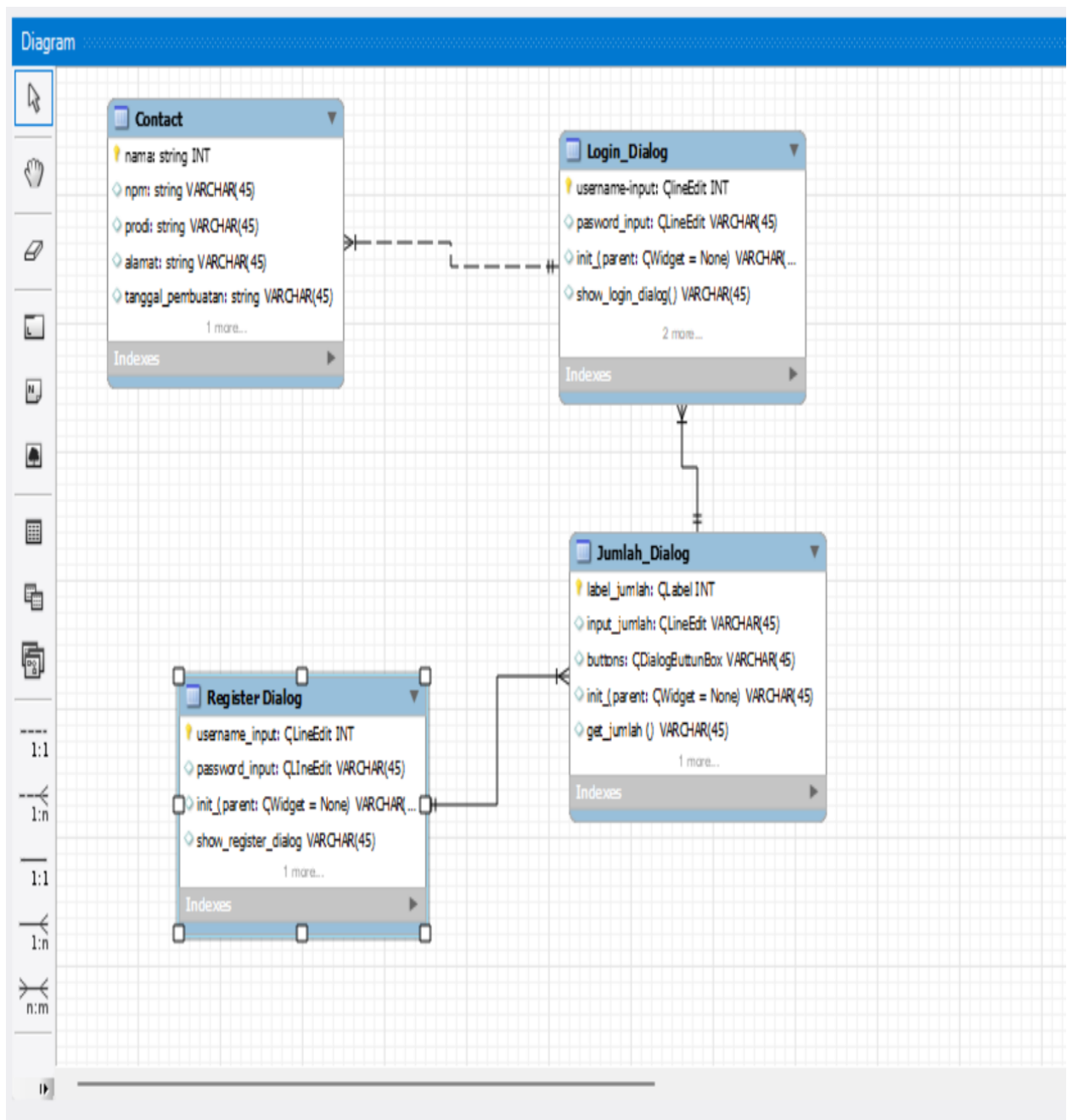
## DAFTAR ISI

<b>Sampul .....</b>	<b>i</b>
<b>Daftar Isi.....</b>	<b>ii</b>
<b>Flowchart Sistem.....</b>	<b>3</b>
<b>Class Diagram.....</b>	<b>4</b>
<b>Hasil Program Dan Penjelasan.....</b>	<b>5</b>
<b>Daftar Pustaka.....</b>	<b>15</b>

## FLOWCHART SISTEM

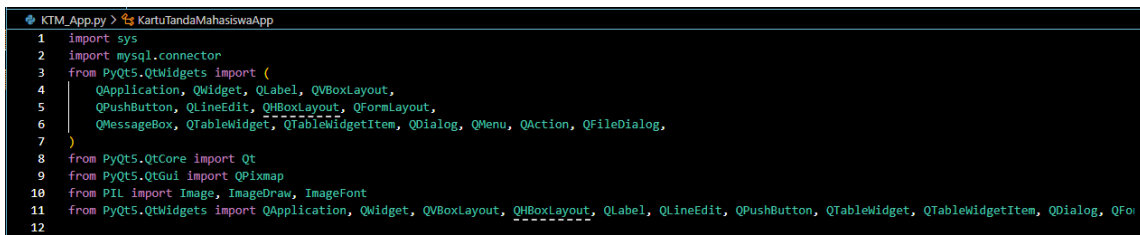


## CLASS DIAGRAM



## Hasil Program Dan Penjelasan

## 1. Import



```
1 import sys
2 import mysql.connector
3 from PyQt5.QtWidgets import (
4     QApplication, QWidget, QLabel, QVBoxLayout,
5     QPushButton, QLineEdit, QHBoxLayout, QFormLayout,
6     QMessageBox, QTableWidgetItem, QDialog, QMenu, QAction, QFileDialog,
7 )
8 from PyQt5.QtCore import Qt
9 from PyQt5.QtGui import QPixmap
10 from PIL import Image, ImageDraw, ImageFont
11 from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QTableWidgetItem, QDialog, QFo
12
```

Kode yang disediakan adalah skrip Python yang menggunakan perpustakaan PyQt5 untuk membuat antarmuka pengguna grafis (GUI) untuk berinteraksi dengan database MySQL. Berikut adalah rincian komponen utama dan tujuannya:

### 1. Pernyataan Import:

- **import sys:** Mengimpor modul sys, yang menyediakan akses ke beberapa variabel yang digunakan atau dikelola oleh interpreter Python dan ke fungsi yang berinteraksi kuat dengan interpreter.
- **import mysql.connector:** Mengimpor modul Konektor MySQL, yang digunakan untuk menyambung ke database MySQL dan menjalankan kueri SQL.
- **from PyQt5.QtWidgets import ...:** Mengimpor berbagai kelas dan fungsi dari perpustakaan PyQt5, yang merupakan kumpulan pengikatan Python untuk Qt, perangkat GUI yang populer.
- **from PyQt5.QtCore import Qt:** Mengimpor modul Qt dari PyQt5, yang digunakan untuk menangani fungsionalitas inti.
- **from PyQt5.QtGui import QPixmap:** Mengimpor kelas QPixmap dari modul QtGui PyQt5, yang digunakan untuk bekerja dengan gambar.

### 2. Widget PyQt5:

- Script mendefinisikan beberapa widget PyQt5 untuk membangun GUI, seperti **QApplication, QWidget, QLabel, QVBoxLayout, QPushButton, QLineEdit, QHBoxLayout, QFormLayout, QMessageBox, QTableWidgetItem, QDialog, QMenu, QAction, QFileDialog**, dan lain-lain.

### 3. Koneksi Basis Data:

- Skrip membuat koneksi ke database MySQL menggunakan **mysql.connector** modul. Namun, detail koneksi database spesifik tidak disediakan dalam cuplikan kode.

### 4. PIL (Perpustakaan Pencitraan Python):

- Skrip mengimpor kelas **Image, ImageDraw, dan ImageFont** dari Python Imaging Library (PIL). Kelas-kelas ini digunakan untuk bekerja dengan gambar. Hal ini menunjukkan bahwa skrip mungkin melibatkan pemrosesan atau manipulasi gambar.

### 5. Tata Letak dan Komponen GUI:

- Skrip menyiapkan antarmuka pengguna grafis dengan berbagai komponen seperti label, tombol, tabel, dll. Tata letak ditentukan menggunakan **QVBoxLayout, QHBoxLayout, dan QFormLayout**.

## 6. Kegunaan:

- Tanpa kode lengkap, sulit untuk menentukan fungsi sebenarnya. Namun, berdasarkan modul dan widget yang diimpor, sepertinya ini adalah skrip untuk membuat aplikasi GUI yang berinteraksi dengan database MySQL, yang kemungkinan melibatkan pemrosesan gambar menggunakan PIL.

## 7. Dialog dan Kotak Pesan:

- Skrip ini digunakan **QMessageBox** untuk menampilkan kotak pesan dan berpotensi mendapatkan masukan pengguna.
- **QFileDialog** mungkin digunakan untuk membuka dialog file untuk memilih file

## 8. Kode Tidak Lengkap:

- Tampaknya cuplikan kode yang diberikan tidak lengkap. Mungkin ada bagian kode yang hilang yang menentukan fungsi sebenarnya dari aplikasi.

Untuk memahami sepenuhnya tujuan dan fungsi skrip, Anda perlu meninjau keseluruhan kode, termasuk bagian yang hilang

## 2. Class Contact

```
class Contact:
    def __init__(self, nama, npm, prodi, alamat, tanggal):
        self.nama = nama
        self.npm = npm
        self.prodi = prodi
        self.alamat = alamat
        self.tanggal = tanggal
```

Class Contact pada gambar kode di atas bertujuan untuk merepresentasikan sebuah entitas kontak, dengan atribut-atribut seperti nama, npm (Nomor Pokok Mahasiswa), program studi (prodi), alamat, dan tanggal. Selain atribut tersebut, class ini memiliki metode `create_table_if_not_exists` yang bertujuan untuk membuat tabel di database jika tabel tersebut belum ada.

## 3. Class KartuTandaMahasiswaApp(QWidget)

```
15 class KartuTandaMahasiswaApp(QWidget):
```

Kode yang Anda sertakan menunjukkan definisi kelas **KartuTandaMahasiswaApp**, yang merupakan subkelas dari **QWidget**. Ini mungkin adalah bagian awal dari program atau modul yang bertanggung jawab untuk membuat antarmuka pengguna aplikasi Kartu Tanda Mahasiswa (KTM) menggunakan framework PyQt

#### 4. Inisialisasi Objek dan Koneksi ke Database

```
16 def __init__(self):
17     super().__init__()
18     self.init_ui()
19
20
21     self.db_connection = mysql.connector.connect(
22         host='localhost',
23         user='root',
24         password='',
25         database='Ktm',
26     )
27
28     self.cursor = self.db_connection.cursor()
29
30     self.create_table_if_not_exists()
31     self.is_authenticated = False
32     self.login_dialog = LoginDialog(self)
33     self.register_dialog = RegisterDialog(self)
34
35     # Tambahkan widget untuk menampilkan data
36     self.data_table = QTableWidgetItem(self)
37     self.data_table.setColumnCount(6)
38     self.data_table.setHorizontalHeaderLabels(['ID', 'Nama', 'NPM', 'Prodi', 'Alamat', 'Tanggal'])
39     self.layout.addWidget(self.data_table)
40
41     # Tambahkan QLabel untuk menampilkan data yang disimpan
42     self.saved_data_label = QLabel('Data yang disimpan:', self)
43     self.layout.addWidget(self.saved_data_label)
44
45     # Tambahkan stylesheet untuk warna latar belakang dan elemen-elemen lainnya
46     self.setStyleSheet("""
47     QWidget {
48         background-color: lightgray;
49     }
50     """)
51
```

##### a) Inisialisasi Objek dan Koneksi ke Database:

- Membuat objek dari kelas **KartuTandaMahasiswaApp** yang merupakan subclass dari **QWidget**.
- Memanggil konstruktor kelas induk (**super().\_\_init\_\_()**) untuk inisialisasi objek.
- Memanggil metode **init\_ui()** untuk menyiapkan antarmuka pengguna.
- Membuat koneksi ke database MySQL menggunakan modul **mysql.connector**.

##### b) Inisialisasi Kursor dan Tabel Data:

- Membuat objek kursor (**self.cursor**) untuk melakukan operasi database.
- Memanggil metode **create\_table\_if\_not\_exists()** untuk membuat tabel di database jika belum ada.
- Membuat widget tabel (**self.data\_table**) untuk menampilkan data mahasiswa.
- Menetapkan jumlah kolom dan header kolom pada tabel.
- Menambahkan tabel ke dalam layout utama (**self.layout**).

##### c) Widget QLabel untuk Menampilkan Data yang Disimpan:

- Membuat QLabel (**self.saved\_data\_label**) yang akan digunakan untuk menampilkan data yang telah disimpan.
- Menambahkan QLabel ke dalam layout utama.

##### d) Penyusunan Layout dan Penyesuaian Tampilan:

- Mengatur stylesheet untuk memberikan warna latar belakang dan gaya lainnya pada widget utama (**QWidget**).
- Menambahkan elemen-elemen (tabel dan QLabel) ke dalam layout utama (**self.layout**).

##### e) Inisialisasi Dialog Login dan Register:

- Menginisialisasi variabel **self.is\_authenticated** sebagai penanda autentikasi pengguna (belum dijelaskan dalam potongan kode).
- Membuat objek dari kelas **LoginDialog** dan **RegisterDialog** yang akan digunakan untuk proses login dan registrasi.

Seluruh alur ini membentuk dasar aplikasi Kartu Tanda Mahasiswa (KTM) dengan antarmuka pengguna yang menggunakan Qt dan dapat terhubung ke database MySQL. Beberapa fitur utama seperti tabel data, QLabel untuk menampilkan data, dan inisialisasi dialog login dan registrasi sudah disiapkan. Selanjutnya, pengembang dapat menambahkan logika tambahan dan fungsionalitas sesuai dengan kebutuhan aplikasi.

5. Menambahkan stylesheet

```
# Tambahkan stylesheet untuk warna latar belakang dan elemen-elemen lainnya
self.setStyleSheet("""
    QWidget {
        background-color: lightgray;
    }
    """)
```

untuk warna latar belakang dan elemen-elemen lainnya.

6. Menambahkan fungsi show\_data

```
def show_data(self):
    self.data_table.setRowCount(0) # Menghapus data yang sudah ditampilkan sebelumnya
    try:
        query = 'SELECT * FROM attendance'
        self.cursor.execute(query)
        data = self.cursor.fetchall()

        for row_number, row_data in enumerate(data):
            self.data_table.insertRow(row_number)
            for column_number, column_data in enumerate(row_data):
                item = QTableWidgetItem(str(column_data))
                self.data_table.setItem(row_number, column_number, item)

    except Exception as e:
        print(f"Failed to fetch data: {str(e)}")
```

Fungsi show\_data bertanggung jawab untuk menampilkan data dari database ke dalam tabel (QTableWidget). Mengatur jumlah baris tabel (self.data\_table) menjadi 0 untuk menghapus data yang sudah ditampilkan sebelumnya

7. Menambahkan create table

```
107
108 def create_table_if_not_exists(self):
109     try:
110         query = '''
111         CREATE TABLE IF NOT EXISTS attendance (
112             id INT AUTO_INCREMENT PRIMARY KEY,
113             nama VARCHAR(255),
114             npm VARCHAR(255),
115             prodi VARCHAR(255),
116             alamat VARCHAR(255),
117             tanggal VARCHAR(255)
118         )
119         '''
120         self.cursor.execute(query)
121         self.db_connection.commit()
122     except Exception as e:
123         print(f"Error creating table: {str(e)}")
124
```

bagian dari suatu kelas atau modul yang berfungsi untuk membuat tabel dalam sebuah database jika tabel tersebut belum ada. dan untuk Membuat query SQL untuk membuat table, Mengeksekusi query tersebut pada database, Menangani kesalahan jika ada

8. Menambahkan delete contact



```

124
125
126     def delete_contact_from_database(self, npm):
127         try:
128             query = 'DELETE FROM attendance WHERE npm = ?'
129             self.cursor.execute(query, (npm,))
130             self.db_connection.commit()
131             QMessageBox.information(self, 'Information', 'Data telah dihapus.')
132         except Exception as e:
133             QMessageBox.critical(self, 'Error', f"Failed to delete data: {str(e)}")
134             print(f"Failed to delete data: {str(e)}")
135

```

Pada bagian dari pemrograman ini untuk mengelola data mahasiswa, dan Membuat query SQLDELETE untuk menghapus data dari tabel 'attendance' berdasarkan NPM, Mengeksekusi query pada database, Menangani kesalahan dan memberikan informasi atau pesan kesalahan sesuai hasil eksekusi.

## 9. Menambahkan Init\_ui

```

135
136     def init_ui(self):
137         self.setGeometry(100, 100, 100, 200)
138         self.setWindowTitle('Aplikasi Kartu Tanda Mahasiswa')
139
140         self.layout = QVBoxLayout()
141
142         self.setup_widgets()

```

Pada bagian kode pemrograman ini adalah untuk menginisialisasi anatarmuka pengguna (UI) dalam suatu aplikasi. Dan untuk mengatur ukuran dan judul jendela aplikasi, membuat objek layout vertical sebagai layout utama.

## 10. Menambahkan setup\_widgets

```

143
144     def setup_widgets(self):
145         self.form_layout = QFormLayout()
146         self.name_input = QLineEdit()
147         self.npm_input = QLineEdit()
148         self.program_input = QLineEdit()
149         self.alamat_input = QLineEdit()
150         self.tanggal_input = QLineEdit()
151
152         self.title_label = QLabel('Kartu Tanda Mahasiswa', self)
153         self.layout.addWidget(self.title_label, alignment=Qt.AlignmentFlag.AlignTop | Qt.AlignmentFlag.AlignHCenter)
154         self.form_layout.addRow('Nama:', self.name_input)
155         self.form_layout.addRow('NPM:', self.npm_input)
156         self.form_layout.addRow('Program Studi:', self.program_input)
157         self.form_layout.addRow('Alamat:', self.alamat_input)
158         self.form_layout.addRow('Tanggal Pembuatan:', self.tanggal_input)
159
160         self.layout.addLayout(self.form_layout)
161
162         self.photo_label = QLabel('Foto Mahasiswa')
163         self.layout.addWidget(self.photo_label)
164
165         # Tambahkan menu dropdown saat mengklik tombol "Buat Kartu"
166         self.menu_button = QPushButton('Buat Kartu', self)
167         self.menu_button.clicked.connect(self.show_card_menu)
168         self.layout.addWidget(self.menu_button)
169
170         self.save_button = QPushButton('Save', self)
171         self.save_button.clicked.connect(self.save_data)
172         self.layout.addWidget(self.save_button)
173
174         # Tambahkan QPushButton untuk upload foto
175         self.upload_button = QPushButton('Upload Foto', self)
176         self.upload_button.clicked.connect(self.upload_photo)
177         self.layout.addWidget(self.upload_button)
178
179         self.delete_button = QPushButton('Hapus', self)
180         self.delete_button.clicked.connect(self.delete_data)
181
182         self.layout.addWidget(self.delete_button)
183
184         self.exit_button = QPushButton('Exit', self)
185         self.exit_button.clicked.connect(self.close)
186         self.layout.addWidget(self.exit_button)
187
188         self.informasi_label = QLabel('Klik Buat Kartu jika data yang dimasukkan sudah benar.', self)
189         self.layout.addWidget(self.informasi_label)
190
191         self.setLayout(self.layout)

```

Kode ini untuk mengatur dan menambahkan elemen-elemen antarmuka pengguna (UI) ke dalam layout. Membuat widget-widget untuk elemen-elemen antarmuka pengguna. Menambahkan widget-widget tersebut ke dalam layout utama. Menghubungkan tombol-tombol dengan fungsi-fungsi yang akan dijalankan saat tombol ditekan. Mengatur layout utama untuk kelas atau modul tersebut.

## 11. Menambahkan fungsi save\_data

```

192 def save_data(self):
193     e = None # Initialize the variable before the try block
194     try:
195         # Dapatkan data dari input pengguna
196         nama = self.name_input.text()
197         npm = self.npm_input.text()
198         program = self.program_input.text()
199         alamat = self.alamat_input.text()
200         tanggal = self.tanggal_input.text()
201
202         # Menampilkan data yang disimpan di QLabel
203         saved_data_text = f"Nama: {nama}, NPM: {npm}, Prodi: {program}, Alamat: {alamat}, Tanggal: {tanggal}"
204         self.saved_data_label.setText(saved_data_text)
205
206         # Save data to the database
207         contact = Contact(nama, npm, program, alamat, tanggal)
208         self.save_contact_to_database(contact)
209
210         QMessageBox.information(self, 'Information', 'Data sudah Disimpan!')
211
212         # Menampilkan data ke dalam tabel setelah data disimpan
213         self.show_data()
214     except Exception as e:
215         e = str(e) # Update the variable if an exception occurs
216         QMessageBox.critical(self, 'Error', f"An error occurred: {str(e)}")
217         print(f"An error occurred: {str(e)}")
218     print(e) # This will print the exception message or None if no exception occurred
219

```

program di atas melakukan beberapa tindakan untuk menyimpan data dari pengguna dan memasukkan ke dalam table setelah disimpan.

## 12. Menambahkan fungsi save\_contact\_to\_database

```

221 def save_contact_to_database(self, contact):
222     try:
223         query = '''
224             INSERT INTO attendance (nama, npm, prodi, alamat, tanggal)
225             VALUES (%s, %s, %s, %s, %s)
226         '''
227         self.cursor.execute(query, (contact.nama, contact.npm, contact.prodi, contact.alamat, contact.tanggal))
228         self.db_connection.commit()
229
230         self.show_data()
231
232     except Exception as e:
233         QMessageBox.critical(self, 'Error', f"Failed to save contact to database: {str(e)}")
234         print(f"Failed to save contact to database: {str(e)}")
235

```

Fungsi save\_contact\_to\_database bertanggung jawab untuk menyimpan data kontak ke dalam database. menggunakan CSS yang diterapkan pada elemen QWidget (yang mencakup semua elemen di dalam aplikasi).

## 13. Menambahkan fungsi show\_card\_menu

```

235 def show_card_menu(self):
236     menu = QMenu(self)
237
238     # Tambahkan opsi menu
239     manual_action = QAction('Buat Kartu Manual', self)
240     manual_action.triggered.connect(self.generate_card_manual)
241     menu.addAction(manual_action)
242
243     automatic_action = QAction('Buat Kartu Otomatis', self)
244     automatic_action.triggered.connect(self.generate_card_automatic)
245     menu.addAction(automatic_action)
246
247     # Tampilkan menu di posisi tombol
248     menu.exec_(self.menu_button.mapToGlobal(self.menu_button.pos()))
249
250

```

Fungsi `show_card_menu` bertanggung jawab untuk menampilkan menu konteks (menu kartu) saat tombol "Buat Kartu" diklik. Dan memilih untuk menambahkan fitur Buat Kartu Manual & Buat Kartu Otomatis.

#### 14. Menambahkan `generate_crate_manual`

```
251 def generate_card_manual(self):
252     # Dapatkan data dari input pengguna atau dari data yang sudah ada
253     name = self.name_input.text()
254     npm = self.npm_input.text()
255     program = self.program_input.text()
256     alamat = self.alamat_input.text()
257     tanggal = self.tanggal_input.text()
258
259     # Implementasikan logika pembuatan kartu manual di sini
260
261     # Implementasi logika pemilihan file gambar
262     file_dialog = QFileDialog()
263     file_dialog.setNameFilter("Image files (*.png *.jpg *.bmp)")
264     file_dialog.setWindowTitle("Pilih File Gambar")
265     file_dialog.setFileMode(QFileDialog.ExistingFile)
266
267     if file_dialog.exec_():
268         # Dapatkan path file yang dipilih
269         selected_file = file_dialog.selectedFiles()[0]
270
271         # Set teks pada label foto
272         self.photo_label.setText(f>Nama: {name}\nNPM: {npm}\nProgram Studi: {program}\nAlamat: {alamat}\nTanggal Pembuatan: {tanggal}")
273
274         # Misalnya, menggunakan QLabel untuk menampilkan informasi kartu
275         card_text = f>Nama: {name}\nNPM: {npm}\nProgram Studi: {program}\nAlamat: {alamat}\nTanggal Pembuatan: {tanggal}"
276
277         # Set teks pada label foto
278         self.photo_label.setText(card_text)
279
280         # Jika Anda ingin menyimpan gambar kartu, implementasikan logika penyimpanan di sini
281         # Misalnya, menggunakan library seperti Pillow untuk membuat gambar dan menyimpannya
282         # Simpan kartu sebagai gambar
283         self.save_card_to_image(selected_file, card_text)
```

Kode pemrograman diatas untuk pembuatan kartu manual. mengambil data dari input pengguna, memberikan tampilan informasi kartu pada label, memungkinkan pengguna memilih file gambar, dan memberikan opsi untuk menyimpan kartu sebagai gambar. Namun, logika sebenarnya untuk pembuatan kartu dan penyimpanan gambar perlu diimplementasikan di dalam bagian yang sesuai

#### 15. Menambahkan `save_card_to_image`

```
284
285 def save_card_to_image(self, file_path, card_text):
286     try:
287         # Contoh penyimpanan gambar menggunakan Pillow
288         image = Image.new('RGB', (50, 50), color='white')
289         draw = ImageDraw.Draw(image)
290         font = ImageFont.load_default()
291
292         # Atur teks pada gambar
293         draw.text((10, 10), card_text, fill='black', font=font)
294
295         # Simpan gambar sebagai file
296         image.save(file_path)
297         QMessageBox.information(self, 'Information', f"Kartu disimpan sebagai {file_path}")
298     except Exception as e:
299         QMessageBox.critical(self, 'Error', f"Failed to save card to image: {str(e)}")
300
```

untuk menyimpan informasi kartu dalam bentuk teks ke dalam sebuah gambar, menggunakan library Pillow (PIL). Dapatkan data dari input pengguna. Tampilkan dialog pemilihan file gambar dan dapatkan path file yang dipilih. Tetapkan teks pada label foto dan buat teks kartu. Simpan kartu sebagai gambar menggunakan library Pillow. Tampilkan pesan informasi jika penyimpanan berhasil atau pesan kesalahan jika terjadi kesalahan.

## 16. Metode generate\_card\_automatic

```
301 def generate_card_automatic(self):
302     # Dapatkan data dari database atau sumber data lainnya
303     # Implementasikan logika pembuatan kartu otomatis di sini
304     # Misalnya, menggunakan QLabel untuk menampilkan informasi kartu
305     card_text = "Informasi Kartu Otomatis"
306
307     # Set teks pada label foto
308     self.photo_label.setText(card_text)
309
310     # Jika Anda ingin menyimpan gambar kartu, implementasikan logika penyimpanan di sini
311     # Misalnya, menggunakan library seperti Pillow untuk membuat gambar dan menyimpannya
312
313     # Contoh penyimpanan gambar menggunakan Pillow
314     # from PIL import Image, ImageDraw, ImageFont
315     # image = Image.new('RGB', (width, height), color='white')
316     # draw = ImageDraw.Draw(image)
317     # font = ImageFont.load_default()
318     # draw.text((x, y), card_text, fill='black', font=font)
319     # image.save('kartu_mahasiswa.png')
320
```

Mengambil data dari database atau sumber data lainnya. Implementasi logika pembuatan kartu otomatis di sini (belum diimplementasikan). Misalnya, menggunakan QLabel untuk menampilkan informasi kartu. Jika Anda ingin menyimpan gambar kartu, implementasikan logika penyimpanan di sini (belum diimplementasikan).

## 17. Metode delete\_data:

```
341 def delete_data(self):
342     selected_rows = self.data_table.selectionModel().selectedRows()
343
344     # Periksa apakah ada baris yang dipilih
345     if not selected_rows:
346         QMessageBox.warning(self, 'Warning', 'Pilih baris yang akan dihapus.')
347         return
348
349     # Loop melalui baris yang dipilih
350     for selected_row in selected_rows:
351         row_index = selected_row.row()
352
353         # Dapatkan nilai NPM dari kolom ke-2 (indeks 1) dalam baris yang dipilih
354         npm = self.data_table.item(row_index, 2).text()
355
356         # Tampilkan dialog konfirmasi
357         confirmation = QMessageBox.question(self, 'Konfirmasi Hapus', 'Anda yakin ingin menghapus data terpilih?',
358                                           QMessageBox.Yes | QMessageBox.No)
359
360         if confirmation == QMessageBox.Yes:
361             # Panggil fungsi untuk menghapus data berdasarkan NPM
362             self.delete_contact_from_database(npm)
```

Mengambil baris yang dipilih dari tabel. Memeriksa apakah ada baris yang dipilih, menampilkan pesan peringatan jika tidak ada. Loop melalui baris yang dipilih dan menghapus data berdasarkan NPM dari setiap baris.

## 18. Metode delete\_contact

```
363 def delete_contact_from_database(self, npm):
364     error_message = "" # Initialize the variable before the try block
365     try:
366         query = 'DELETE FROM attendance WHERE npm = %s'
367         self.cursor.execute(query, (npm,))
368         self.db_connection.commit()
369         QMessageBox.information(self, 'Information', 'Data has been deleted.')
370     except Exception as e:
371         error_message = f"Failed to delete data: {str(e)}"
372         QMessageBox.critical(self, 'Error', error_message)
373     print(error_message)
```

Metode ini bertujuan untuk menghapus data dari database berdasarkan NPM yang diberikan. Jika operasi berhasil, sebuah pesan informasi ditampilkan; jika gagal, pesan kesalahan ditampilkan dalam bentuk dialog kritis, dan pesan tersebut juga dicetak ke konsol.

## 19. Metode closeEvent:

```
375
376 def closeEvent(self, event):
377     # ... (previous code)
378     # Panggil fungsi untuk membuat kartu di sini
379     self.create_mahasiswa_card(self.name_input.text(), self.npm_input.text(), self.program_input.text(), self.alamat_input.text(),
380                               self.tanggal_input.text())
381
```

Menggantikan metode closeEvent untuk menambahkan logika khusus saat aplikasi ditutup. Memanggil metode create\_mahasiswa\_card untuk membuat kartu mahasiswa berdasarkan data input.

## 20. Metode create\_mahasiswa\_card

```
382
383 def create_mahasiswa_card(self, name, npm, program, alamat, tanggal):
384     # Implementasikan logika pembuatan kartu di sini
385     # Misalnya, Anda dapat menggunakan library seperti Pillow untuk manipulasi gambar
386     # dan menyimpannya sebagai file gambar kartu
387
388     # Contoh sederhana
389     card_text = f>Nama: {name}\nNPM: {npm}\nProgram Studi: {program}\nAlamat: {alamat}\nTanggal Pembuatan: {tanggal}"
390     self.photo_label.setText(card_text)
391
```

Implementasi logika pembuatan kartu mahasiswa di sini (belum diimplementasikan). Misalnya, menggunakan QLabel untuk menampilkan informasi kartu.

## 21. Menjalankan Aplikasi:

```
391
392 if __name__ == '__main__':
393     app = QApplication(sys.argv)
394     main_app = KartuTandaMahasiswaApp()
395     main_app.show()
396     sys.exit(app.exec_())
397
```

Membuat objek QApplication.

Membuat objek KartuTandaMahasiswaApp.

Menampilkan aplikasi.

Memulai eksekusi aplikasi dengan app.exec\_().

Aplikasi Kartu Tanda Mahasiswa

Kartu Tanda Mahasiswa

Nama:

NPM:

Program Studi:

Alamat:

Tanggal Pembuatan:

Foto Mahasiswa •

Klik Buat Kartu jika data yang dimasukkan sudah benar.

ID	Nama	NPM	Prodi	Alamat	Tanggal
----	------	-----	-------	--------	---------

Data yang disimpan:

## DAFTAR PUSTAKA

- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.
- Sommerville, I. (2011). *Software Engineering* (9th ed.). Addison-Wesley.
- Keamanan Aplikasi:
- Bishop, M. (2003). *Computer Security: Art and Science*. Addison-Wesley.
- Desain Antarmuka Pengguna (UI/UX):
- Cooper, A., Reimann, R., & Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design*. Wiley.
- Pengembangan Web:
- W3Schools Online Tutorials. (<https://www.w3schools.com/>)
- McFarland, D. (2018). *JavaScript & jQuery: The Missing Manual*. O'Reilly Media.