BUKU LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

Nama Praktikan:	Raynaldi Paniroean Panjaitan
	Tazkya Mutia Ramadhan
	Zahara
NIM:	607012400099
	607012400128
	607012400136
Kelas:	D3SI- 48-05
Dosen Pengampu:	Hanang Priambodo, S.T., M.Kom.
Tanggal Dokumen:	2 Juni, 2025

Daftar Isi

DAFTAR KODE SUMBER

Daftar Gambar

Daftar Tabel

1 Tugas Besar dan Assessment Tiga

Aplikasi Pengelolaan Tugas Kuliah dibuat untuk mendukung mahasiswa dalam mengatur beragam tugas akademis dengan lebih efisien. Aplikasi ini menawarkan kemampuan untuk menambahkan, mengubah, dan menghapus tugas beserta informasi seperti tenggat waktu, prioritas, dan mata kuliah yang relevan. Ada dua kategori tugas yang diatur dalam sistem, yakni TugasIndividu dan TugasKelompok, yang keduanya merupakan subkelas dari kelas abstrak Tugas. Mahasiswa dapat memiliki berbagai tugas, membentuk hubungan one-to-many antara entitas Mahasiswa dan Tugas. Di samping itu, sistem menawarkan fungsi kolaborasi untuk pekerjaan kelompok, memungkinkan sejumlah mahasiswa berinteraksi dalam satu tugas. Aplikasi juga menggunakan antarmuka Notifikasi untuk mengirimkan pengingat otomatis agar pengguna dapat menghindari keterlambatan saat tenggat waktu tugas semakin dekat

1.1 Tujuan Assessment

Tujuan dari pengembangan Aplikasi Manajemen Tugas Kuliah ini adalah untuk menyediakan solusi digital yang memudahkan mahasiswa dalam mengelola berbagai tugas akademik secara efektif dan efisien.

1. Membantu mahasiswa dalam mengelola dan mengorganisasi tugas-tugas kuliah secara sistematis melalui fitur tambah, edit, dan hapus tugas dengan informasi penting seperti deadline, prioritas, dan mata kuliah.
2. Meminimalkan risiko keterlambatan pengumpulan tugas dengan menyediakan sistem notifikasi otomatis yang mengingatkan peng-

guna saat tenggat waktu sudah dekat.

3. Mendukung kerja sama tim dalam tugas kelompok, dengan menyediakan fitur kolaborasi yang memungkinkan beberapa mahasiswa berkontribusi dalam satu tugas bersama.
4. Mempermudah dosen atau koordinator akademik dalam memantau tugas mahasiswa, melalui struktur data yang rapi dan relasi jelas antara mahasiswa dan tugas-tugas mereka.
5. Mengimplementasikan konsep pemrograman berorientasi objek seperti inheritance dan interface dalam struktur aplikasi, untuk menciptakan sistem yang modular, efisien, dan mudah dikembangkan di masa depan.

1.2 Alat dan Bahan

Alat dan bahan yang digunakan dalam praktikum ini terdiri dari dua kategori utama, yaitu perangkat keras dan perangkat lunak. Perangkat keras mencakup semua komponen fisik yang diperlukan untuk menjalankan program, sedangkan perangkat lunak mencakup aplikasi dan sistem yang digunakan untuk menulis, menguji, dan menjalankan kode program.

1.2.1 Perangkat Lunak

- Integrated Development Environment (IDE): Visual Studio Code (Yang telah diinstal dan dikonfigurasi pada praktikum minggu pertama sebagai editor Java

1.2.2 Perangkat Keras

- Prosesor: intel core i5
- RAM: 16
- Ruang Penyimpanan: 512 GB

1.3 Langkah Kerja

Pelaksanaan tugas besar/assessment 3 ini terdiri dari dua tahapan utama, yaitu persiapan dan implementasi program Java, yang dirancang secara sistematis untuk memastikan mahasiswa dapat mengikuti setiap langkah dengan baik serta mencapai tujuan tugas secara optimal. Penjelasan lebih rinci mengenai kedua tahapan tersebut diuraikan pada subsection 1.4.1 Persiapan dan 1.4.2 Implementasi Program.

1.3.1 Persiapan

Sebelum memulai praktikum, terdapat beberapa langkah persiapan yang perlu dilakukan guna memastikan kelancaran pelaksanaan kegiatan. Persiapan ini bertujuan untuk menyiapkan lingkungan pengembangan yang sesuai serta memastikan bahwa semua komponen yang diperlukan telah tersedia.

Persiapan 1: Siapkan Visual Studio Code untuk JavaFX Adapun langkah-langkah persiapan visual studio code yang harus dilakukan adalah sebagai berikut:

1. Buka jendela baru dengan memilih opsi file->New Window.

2. Kemudian pada search bar yang berada dibagian tengah atas, ketik >create dan Pilih "Java: Create Java Project".

3. Kemudian pilih opsi "Java Fx".

4. Masukan Group Id pada proyek yaitu: com.cms-project, kemudian tekan enter.

5. Selanjutnya masih pada search bar, masukan nama proyek, yaitu: "cms-project", kemudian tekan enter.

6. Akan tampil halaman lokasi folder. Pilih folder yang akan dijadikan tempat penyimpanan source code. Setelah folder dipilih, klik "Select the project location".

7. Pada bagian terminal ditengah bawah, akan muncul kalimat Define value for property 'version' 1.0-SNAPSHOT: :, tekan tombol enter pada keyboard. 8. Pada bagian terminal ditengah bawah, akan muncul kalimat Y: :, tekan tombol enter pada keyboard.

9. Kemudian ketik tombol apapun pada keyboard ketika terdapat

kalimat Terminal will be reused by tasks, press any key to close it. pada terminal bagian tengah bawah.

10. Selanjutnya pilih menufile→OpenFolder. Kemudianpilihfolder yang telah dibuat (cms-project).

11. Pada halaman VSC bagian kiri atas akan muncul hierarki direktori dan berkas proyek yang terdiri dari:

> src

> target

pom.xml

12. PadabagiankiribawahterdapatmenubernamaJAVAPROJECTS. Klik menu JAVA PROJECTS → Referenced Libraries. Kemudian klik simbol +, masukan pustaka dengan berkas berekstensi .jar yang berada pada folder javafx-sdk-X.Y.Z. Adapun berkasnya adalah:

- javafx.web.jar
- javafx.swing.jar
- javafx.media.jar
- javafx.grapichs.jar
- javafx.fxml.jar
- javafx.controls.jar
- javafx.base.jar
- javafx-swt.jar

Catatan: jika Referenced Libraries tidak ditemukan, klik tanda · · · padaJAVAPROJECTS,kemudianpilih"ConfigureClassPath". Tam bahkan semua file .jar pada menu Libraries . Kemudia pada menu JDK Runtime , pilih runtime JavaSE-21 atau JavaSE-23

13. Selanjutnya buka berkas pom.xml, ubah versi javafx sesuai dengan versi javafx yang digunakan. Pengubahan dilakukan pada bagian version, seperti kode sumber dibawah ini:

```
1     <dependencies>
2    <dependency>
3        <groupId>org.openjfx</groupId>
4        <artifactId>javafx-controls</artifactId>
5        <version>21</version>
6    </dependency>
7    <dependency>
```

```
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-fxml</artifactId>
        <version>21</version>
    </dependency>
</dependencies>


```

14. Pada direktori src terdapat hierarki, seperti:

> java

> com

> cms-project

J App.java

J PrimaryController.java

J SecondaryController.java

J module-info.java

> resource

> com

> cms-project Û primary.fxml Û secondary.fxml 15. Selanjutnya hapus semua berkas yang ada di dalam direktori /java/com/cms-project , yaitu: App.java , PrimaryController.java , dan SecondaryController.java 16. Kemudian hapus semua berkas fxml yang ada di dalam di rektori /java/resource/com/cms-project , yaitu: primary.fxml , dan secondary.fxml .

1.3.2 Implementasi Tugas Besar dan Assessment Tiga

Masukan Sumber Kode hasil dan penjelasan dari implementasi selama praktikum.

Konfigurasi Proyek

File `pom.xml` File `pom.xml` ini mendefinisikan proyek sebagai aplikasi Maven. Ini mencakup groupId `com.coursemanagementsystem`, artifactId `course-management-system`, dan version `1.0-SNAPSHOT`. Properti proyek mengatur encoding ke UTF-8 dan versi Java compiler ke 23. Dependensi utama adalah JavaFX controls dan FXML dengan versi $javafx.version(23.0.2).RepositoriOpenJFXjugadisertakanuntukmengambilartefakJavaF$

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
    apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.coursemanagementsystem</groupId>
    <artifactId>course-management-system</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>23</maven.compiler.source>
        <maven.compiler.target>23</maven.compiler.target>
        <javafx.version>23.0.2</javafx.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-controls</artifactId>
            <version>${javafx.version}</version>
        </dependency>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-fxml</artifactId>
            <version>${javafx.version}</version>
        </dependency>
        </dependencies>

    <repositories>
        <repository>
            <id>openjfx</id>
            <url>https://repo.maven.apache.org/maven2</url>
        </repository>
    </repositories>

    <build>
        <plugins>
            <plugin>
                <groupId>kr.motd.maven</groupId>
                <artifactId>os-maven-plugin</artifactId>
                <version>1.7.0</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>detect</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.11.0</version>
                <configuration>
```

```
                    <source>${maven.compiler.source}</source>
                    <target>${maven.compiler.target}</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.openjfx</groupId>
                <artifactId>javafx-maven-plugin</artifactId>
                <version>0.0.8</version>
                <configuration>
                    <mainClass>com.coursemanagementsystem.MainApp</mainClass>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>

```

Kode Sumber 1.1: pom.xml - Konfigurasi Proyek Maven

File `.vscode/settings.json` File ini berisi pengaturan spesifik untuk editor Visual Studio Code. Pengaturan `"java.configuration.updateBuildConfigurati` `"interactive"` menunjukkan bahwa VSCode akan berinteraksi dengan pengguna saat konfigurasi build Java perlu diperbarui.

```
{
    "java.configuration.updateBuildConfiguration": "interactive"
}

```

Kode Sumber 1.2: .vscode/settings.json - Pengaturan VSCode

Struktur Aplikasi JavaFX

File `MainApp.java` Kelas `MainApp` adalah titik masuk utama untuk aplikasi JavaFX, yang meng-extend `javafx.application.Application`. Metode `start(Stage primaryStage)` bertanggung jawab untuk memuat antarmuka pengguna dari file FXML (`/com/coursemanagementsystem/dashboard.fxml`) dan menerapkan stylesheet CSS (`/com/coursemanagementsystem/style.css`). Stage utama diatur dengan judul "Sistem Manajemen Tugas Mahasiswa", ukuran minimum, dan ditampilkan di tengah layar. Metode `main` meluncurkan aplikasi JavaFX.

```
package com.coursemanagementsystem;

import java.net.URL;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
```

```
8  import javafx.scene.Scene;
9  import javafx.stage.Stage;
10
11 public class MainApp extends Application {
12
13     @Override
14     public void start(Stage primaryStage) {
15         try {
16             // Load FXML
17             FXMLLoader loader = new FXMLLoader(getClass().getResource("/com/
       coursemanagementsystem/dashboard.fxml"));
18             Parent root = loader.load();
19             Scene scene = new Scene(root);
20
21             // Load final CSS
22             URL cssResource = getClass().getResource("/com/
       coursemanagementsystem/style.css");
23             if (cssResource != null) {
24                 scene.getStylesheets().add(cssResource.toExternalForm());
25                 System.out.println("    CSS berhasil dimuat: " + cssResource.
       toExternalForm());
26             } else {
27                 System.out.println("    CSS tidak ditemukan di path: /com/
       coursemanagementsystem/style.css");
28             }
29
30             // Setup stage
31             primaryStage.setTitle("Sistem Manajemen Tugas Mahasiswa");
32             primaryStage.setScene(scene);
33             primaryStage.setMinWidth(960);
34             primaryStage.setMinHeight(600);
35             primaryStage.centerOnScreen();
36             primaryStage.show();
37
38         } catch (Exception e) {
39             System.out.println("    Error saat memuat aplikasi:");
40             e.printStackTrace();
41         }
42     }
43
44     public static void main(String[] args) {
45         launch(args);
46     }
47 }
48
```

Kode Sumber 1.3: MainApp.java - Kelas Utama Aplikasi

File `module-info.java` File ini mendefinisikan modul Java bernama `com.coursemanagementsystem`. Modul ini memerlukan beberapa modul JavaFX seperti `javafx.controls`, `javafx.fxml`, dan `javafx.graphics` (secara transitif). Selain itu, modul ini juga memerlukan `java.sql` (secara transitif) untuk koneksi database dan `java.prefs`. Beberapa

paket seperti `com.coursemanagementsystem`, `com.coursemanagementsystem.controller`, `com.coursemanagementsystem.model`, dan `com.coursemanagementsystem.database` dibuka (opens) dan diekspor (exports) agar dapat diakses oleh JavaFX dan modul lain jika diperlukan.

```
module com.coursemanagementsystem {
    requires javafx.controls;
    requires javafx.fxml;
    requires transitive java.sql;
    requires transitive javafx.graphics;
    requires java.prefs;

    opens com.coursemanagementsystem to javafx.fxml;
    exports com.coursemanagementsystem;
    exports com.coursemanagementsystem.controller;
    opens com.coursemanagementsystem.controller to javafx.fxml;
    exports com.coursemanagementsystem.model;
    opens com.coursemanagementsystem.model to javafx.fxml;
    opens com.coursemanagementsystem.database to javafx.fxml;
    exports com.coursemanagementsystem.database;
}

```

Kode Sumber 1.4: module-info.java - Definisi Modul

Model Data

File `Tugas.java` Kelas `Tugas` adalah model dasar untuk representasi sebuah tugas dalam sistem. Atribut-atributnya seperti `id`, `judul`, `deskripsi`, `deadline`, `prioritas`, `mataKuliah`, `tipe`, dan `status` didefinisikan menggunakan JavaFX Properties (`IntegerProperty` dan `StringProperty`). Penggunaan Properties ini memfasilitasi data binding dengan komponen UI JavaFX, seperti `TableView`. Kelas ini memiliki beberapa konstruktor untuk berbagai skenario inisialisasi objek tugas, serta menyediakan metode getter, setter, dan property accessor untuk setiap atribut. Terdapat juga atribut `nama` yang tampaknya digunakan dalam konstruktor tertentu.

```
package com.coursemanagementsystem.model;

import javafx.beans.property.*;

public class Tugas {
    private final IntegerProperty id;
    private final StringProperty judul;
    private final StringProperty deskripsi;
    private final StringProperty deadline;
    private final StringProperty prioritas;
    private final StringProperty mataKuliah;
```

```
12      private final StringProperty tipe;
13      private final StringProperty status;
14
15      private String nama;
16
17      public Tugas(int id, String judul, String deskripsi, String deadline,
        String prioritas, String mataKuliah, String tipe, String status) {
18          this.id = new SimpleIntegerProperty(id);
19          this.judul = new SimpleStringProperty(judul);
20          this.deskripsi = new SimpleStringProperty(deskripsi);
21          this.deadline = new SimpleStringProperty(deadline);
22          this.prioritas = new SimpleStringProperty(prioritas);
23          this.mataKuliah = new SimpleStringProperty(mataKuliah);
24          this.tipe = new SimpleStringProperty(tipe);
25          this.status = new SimpleStringProperty(status);
26      }
27
28      public Tugas(String nama, String deadline, String prioritas, String
        mataKuliah) {
29          this.nama = nama;
30          this.deadline = new SimpleStringProperty(deadline);
31          this.prioritas = new SimpleStringProperty(prioritas);
32          this.mataKuliah = new SimpleStringProperty(mataKuliah);
33          // If other StringProperty fields are required, initialize them as
        needed
34          this.id = new SimpleIntegerProperty(0); // Default id
35          this.judul = new SimpleStringProperty(nama); // Menggunakan nama
        sebagai judul jika relevan
36          this.deskripsi = new SimpleStringProperty("");
37          this.tipe = new SimpleStringProperty("");
38          this.status = new SimpleStringProperty("");
39      }
40
41      public Tugas(String nama, String deadline, String prioritas) {
42          this.nama = nama;
43          this.deadline = new SimpleStringProperty(deadline);
44          this.prioritas = new SimpleStringProperty(prioritas);
45          this.id = new SimpleIntegerProperty(0); // Default id
46          this.judul = new SimpleStringProperty(nama); // Menggunakan nama
        sebagai judul jika relevan
47          this.deskripsi = new SimpleStringProperty("");
48          this.mataKuliah = new SimpleStringProperty(""); // Inisialisasi
        default
49          this.tipe = new SimpleStringProperty("");
50          this.status = new SimpleStringProperty("");
51      }
52
53      // Getter
54      public int getId() { return id.get(); }
55      public String getJudul() { return judul.get(); }
56      public String getDeskripsi() { return deskripsi.get(); }
57      public String getDeadline() { return deadline.get(); }
58      public String getPrioritas() { return prioritas.get(); }
59      public String getMataKuliah() { return mataKuliah.get(); }
60      public String getTipe() { return tipe.get(); }
61      public String getStatus() { return status.get(); }
62      public String getNama() { return nama; }
```

```
63
64      // Property
65      public IntegerProperty idProperty() { return id; }
66      public StringProperty judulProperty() { return judul; }
67      public StringProperty deskripsiProperty() { return deskripsi; }
68      public StringProperty deadlineProperty() { return deadline; }
69      public StringProperty prioritasProperty() { return prioritas; }
70      public StringProperty mataKuliahProperty() { return mataKuliah; }
71      public StringProperty tipeProperty() { return tipe; }
72      public StringProperty statusProperty() { return status; }
73
74      // Setter
75      public void setId(int id) { this.id.set(id); }
76      public void setJudul(String judul) { this.judul.set(judul); }
77      public void setDeskripsi(String deskripsi) { this.deskripsi.set(deskripsi)
        ; }
78      public void setDeadline(String deadline) { this.deadline.set(deadline); }
79      public void setPrioritas(String prioritas) { this.prioritas.set(prioritas)
        ; }
80      public void setMataKuliah(String mataKuliah) { this.mataKuliah.set(
        mataKuliah); }
81      public void setTipe(String tipe) { this.tipe.set(tipe); }
82      public void setStatus(String status) { this.status.set(status); }
83      public void setNama(String nama) { this.nama = nama; }
84  }
85
```

Kode Sumber 1.5: Tugas.java - Model Tugas Dasar

File `TugasIndividu.java` Kelas `TugasIndividu` merupakan turunan dari kelas `Tugas` dan mengimplementasikan interface `Notifikasi`. Konstruktornya memanggil konstruktor superclass `Tugas` dan secara spesifik mengatur mata kuliah serta tipe tugas menjadi "Individu". Metode `kirimPengingat` diimplementasikan untuk mencetak pesan pengingat spesifik untuk tugas individu ke konsol.

```
1  package com.coursemanagementsystem.model;
2
3  public class TugasIndividu extends Tugas implements Notifikasi {
4      public TugasIndividu(String nama, String deadline, String prioritas,
       String mataKuliah) {
5          super(nama, deadline, prioritas); // Memanggil konstruktor Tugas yang
       sesuai
6          setMataKuliah(mataKuliah); // Mengatur mata kuliah
7          setTipe("Individu"); // Mengatur tipe tugas
8      }
9
10     @Override
11     public void kirimPengingat(Tugas tugas) {
12         // Implementasi pengingat spesifik untuk tugas individu
13         // Menggunakan getNama() dari superclass Tugas (yang di-pass sebagai
       parameter)
```

```
14          System.out.println("Pengingat: Tugas Individu " + tugas.getNama() + "
    deadline: " + tugas.getDeadline());
15      }
16  }
17
```

Kode Sumber 1.6: TugasIndividu.java - Model Tugas Individu

File `TugasKelompok.java` Kelas `TugasKelompok` adalah subkelas dari `Tugas` yang juga mengimplementasikan interface `Notifikasi`, dirancang untuk merepresentasikan tugas kelompok. Selain atribut yang diwarisi dari `Tugas`, kelas ini memiliki atribut tambahan berupa `List<Mahasiswa>` untuk menyimpan daftar anggota kelompok. Terdapat dua konstruktor: satu untuk inisialisasi dasar dan satu lagi yang menerima daftar anggota. Metode untuk menambah anggota (`tambahAnggota`) dan mendapatkan/mengatur daftar anggota juga disediakan. Metode `kirimPengingat` diimplementasikan untuk mengirim notifikasi ke semua anggota kelompok.

```
1  package com.coursemanagementsystem.model;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  public class TugasKelompok extends Tugas implements Notifikasi {
7      private List<Mahasiswa> anggota;
8
9      public TugasKelompok(String nama, String deadline, String prioritas,
       String mataKuliah) {
10         super(nama, deadline, prioritas, mataKuliah);
11         setTipe("Kelompok");
12         this.anggota = new ArrayList<>();
13     }
14
15     public TugasKelompok(String nama, String deadline, String prioritas,
       String mataKuliah, List<Mahasiswa> anggota) {
16         super(nama, deadline, prioritas, mataKuliah);
17         setTipe("Kelompok");
18         this.anggota = anggota;
19     }
20
21     public List<Mahasiswa> getAnggota() {
22         return anggota;
23     }
24
25     public void setAnggota(List<Mahasiswa> anggota) {
26         this.anggota = anggota;
27     }
28
29     public void tambahAnggota(Mahasiswa mahasiswa) {
30         this.anggota.add(mahasiswa);
```

```
31      }
32
33      @Override
34      public void kirimPengingat(Tugas tugas) {
35          // Menggunakan getNama() dari objek Tugas yang di-pass
36          System.out.println("Pengingat: Tugas Kelompok " + tugas.getNama() + "
        deadline: " + tugas.getDeadline());
37          if (this.anggota != null) { // Menggunakan anggota dari instance
        TugasKelompok ini
38              for (Mahasiswa m : this.anggota) {
39                  System.out.println("Notifikasi ke: " + m.getNama());
40              }
41          }
42      }
43  }
44  // No changes needed here; fix required in Tugas.java (add constructor and
        getters).
45
```

Kode Sumber 1.7: TugasKelompok.java - Model Tugas Kelompok

File `Mahasiswa.java` Kelas `Mahasiswa` adalah model data sederhana yang digunakan untuk merepresentasikan seorang mahasiswa. Kelas ini memiliki atribut-atribut dasar seperti `id` (integer), `nama` (String), `nim` (String), `email` (String), dan `password` (String). Semua atribut diinisialisasi melalui konstruktor. Metode getter disediakan untuk mengakses nilai dari setiap atribut.

```
1   package com.coursemanagementsystem.model;
2
3   public class Mahasiswa {
4       private int id;
5       private String nama;
6       private String nim;
7       private String email;
8       private String password; // Pertimbangkan implikasi keamanan jika ini
        adalah plain text password
9
10      public Mahasiswa(int id, String nama, String nim, String email, String
        password) {
11          this.id = id;
12          this.nama = nama;
13          this.nim = nim;
14          this.email = email;
15          this.password = password;
16      }
17
18      public int getId() { return id; }
19      public String getNama() { return nama; }
20      public String getNim() { return nim; }
21      public String getEmail() { return email; }
22      public String getPassword() { return password; } // Pertimbangkan untuk
        hashing jika ini password login
```

```
}

```

Kode Sumber 1.8: Mahasiswa.java - Model Mahasiswa

File `Notifikasi.java` `Notifikasi.java` adalah sebuah interface fungsional. Interface ini mendefinisikan satu metode abstrak, yaitu `kirimPengingat(Tugas tugas)`. Kelas-kelas yang mengimplementasikan interface ini (seperti `TugasIndividu` dan `TugasKelompok`) diharapkan menyediakan implementasi spesifik untuk metode ini, yang bertujuan untuk mengirimkan pengingat terkait objek `Tugas` yang diberikan sebagai argumen.

```
package com.coursemanagementsystem.model;

public interface Notifikasi {
    void kirimPengingat(Tugas tugas);
}

```

Kode Sumber 1.9: Notifikasi.java - Interface Notifikasi

Koneksi Database

File `DatabaseConnection.java` Kelas `DatabaseConnection` berfungsi sebagai utilitas untuk mengelola koneksi ke database MySQL. Kelas ini mendefinisikan konstanta privat untuk URL database (`jdbc:mysql://localhost:3306`, nama pengguna (`root`), dan password (dibiarkan kosong untuk diisi pengguna). Metode publik statis `getConnection()` menggunakan `DriverManager.getConnection()` untuk membuat dan mengembalikan objek `Connection` ke database, yang dapat melempar `SQLException` jika koneksi gagal.

```
package com.coursemanagementsystem.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/
    coursemanagementdb";
    private static final String USER = "root";
    private static final String PASS = ""; // ganti dengan password MySQL-mu

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASS);
    }
```

```
}

```

Kode Sumber 1.10: DatabaseConnection.java - Utilitas Koneksi Database

File `TestKoneksi.java` Kelas `TestKoneksi` digunakan untuk melakukan pengujian sederhana terhadap koneksi database yang disediakan oleh kelas `DatabaseConnection`. Metode `main` di kelas ini mencoba untuk mendapatkan koneksi menggunakan `DatabaseConnection.getConnection()` dalam blok try-with-resources. Jika koneksi berhasil diperoleh (objek Connection tidak null), pesan sukses akan dicetak ke konsol. Jika terjadi `SQLException` selama proses koneksi, atau jika objek Connection null, pesan kesalahan akan ditampilkan.

```
package com.coursemanagementsystem.database;

import java.sql.Connection;
import java.sql.SQLException;

public class TestKoneksi {
    public static void main(String[] args) {
        try (Connection conn = DatabaseConnection.getConnection()) {
            if (conn != null) {
                System.out.println("    Koneksi ke database berhasil!");
            } else {
                System.out.println("    Gagal: objek Connection null.");
            }
        } catch (SQLException e) {
            System.out.println("    Terjadi kesalahan koneksi:");
            e.printStackTrace();
        }
    }
}

```

Kode Sumber 1.11: TestKoneksi.java - Tes Koneksi Database

Controller Aplikasi

File `DashboardController.java` Controller `DashboardController` mengatur logika untuk tampilan utama aplikasi (dashboard). Ini mengelola `TableView` untuk menampilkan daftar tugas (`Tugas`). Fitur-fitur utamanya termasuk: memuat data tugas dari database MySQL, menginisialisasi kolom-kolom tabel, dan menyediakan fungsionalitas pencarian melalui `TextField searchField` serta filter berdasarkan

status tugas menggunakan `ComboBox filterStatus`. Controller ini juga menangani aksi pengguna seperti menambah tugas baru (membuka form `tambah_tugas.fxml`), mengedit, dan menghapus tugas. Status tugas dapat diubah langsung dari tabel melalui ComboBox di dalam sel status. Selain itu, controller ini memperbarui label ringkasan untuk jumlah tugas yang mendesak (`urgentCount`), sedang dikerjakan (`inProgressCount`), dan selesai (`completedCount`), serta memberikan pengingat untuk tugas yang mendekati deadline.

```
1  package com.coursemanagementsystem.controller;
2
3  import java.sql.Connection;
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.time.LocalDate;
8  import java.time.format.DateTimeFormatter;
9  import java.time.temporal.ChronoUnit;
10 import java.util.prefs.Preferences;
11
12 import com.coursemanagementsystem.database.DatabaseConnection;
13 import com.coursemanagementsystem.model.Tugas;
14
15 import javafx.collections.FXCollections;
16 import javafx.collections.ObservableList;
17 import javafx.collections.transformation.FilteredList;
18 import javafx.fxml.FXML;
19 import javafx.fxml.FXMLLoader;
20 import javafx.scene.Scene;
21 import javafx.scene.control.Alert;
22 import javafx.scene.control.Button;
23 import javafx.scene.control.ButtonType;
24 import javafx.scene.control.ComboBox;
25 import javafx.scene.control.Label;
26 import javafx.scene.control.TableCell;
27 import javafx.scene.control.TableColumn;
28 import javafx.scene.control.TableView;
29 import javafx.scene.control.TextField;
30 import javafx.scene.control.ToggleButton;
31 import javafx.scene.layout.HBox;
32 import javafx.stage.Stage;
33
34 public class DashboardController {
35
36     @FXML private TextField searchField;
37     @FXML private ComboBox<String> filterStatus;
38     @FXML private TableView<Tugas> taskTable;
39     @FXML private TableColumn<Tugas, String> titleColumn;
40     @FXML private TableColumn<Tugas, String> deadlineColumn;
41     @FXML private TableColumn<Tugas, String> priorityColumn;
42     @FXML private TableColumn<Tugas, String> subjectColumn;
43     @FXML private TableColumn<Tugas, String> typeColumn;
44     @FXML private TableColumn<Tugas, String> statusColumn;
45     @FXML private TableColumn<Tugas, Void> actionColumn;
```

```
46
47      @FXML private Label urgentCount;
48      @FXML private Label inProgressCount;
49      @FXML private Label completedCount;
50      @FXML private ToggleButton toggleThemeBtn; // Fungsionalitas belum
        diimplementasikan
51
52      private ObservableList<Tugas> tugasList = FXCollections.
        observableArrayList();
53      private FilteredList<Tugas> filteredTugas;
54      private Preferences prefs = Preferences.userNodeForPackage(
        DashboardController.class); // Preferences belum banyak digunakan
55
56      @FXML
57      public void initialize() {
58          filteredTugas = new FilteredList<>(tugasList, p -> true);
59          taskTable.setItems(filteredTugas);
60
61          filterStatus.getItems().addAll("Semua", "Belum Dikerjakan", "Sedang
        Dikerjakan", "Selesai");
62          filterStatus.setValue("Semua");
63
64          searchField.textProperty().addListener((obs, oldVal, newVal) ->
        filterTugas());
65          filterStatus.valueProperty().addListener((obs, oldVal, newVal) ->
        filterTugas());
66
67          titleColumn.setCellValueFactory(data -> data.getValue().judulProperty
        ());
68          deadlineColumn.setCellValueFactory(data -> data.getValue().
        deadlineProperty());
69          priorityColumn.setCellValueFactory(data -> data.getValue().
        prioritasProperty());
70          subjectColumn.setCellValueFactory(data -> data.getValue().
        mataKuliahProperty());
71          typeColumn.setCellValueFactory(data -> data.getValue().tipeProperty())
        ;
72          statusColumn.setCellValueFactory(data -> data.getValue().
        statusProperty());
73
74          setupStatusColumn();
75          setupActionColumn();
76          loadTugasFromDatabase();
77      }
78
79      @FXML
80      private void handleAddNewTask() {
81          openFormTugas(null);
82      }
83
84      private void openFormTugas(Tugas tugasToEdit) {
85          try {
86              FXMLLoader loader = new FXMLLoader(getClass().getResource("/com/
        coursemanagementsystem/tambah_tugas.fxml"));
87              Stage stage = new Stage();
88              stage.setTitle(tugasToEdit == null ? "Tambah Tugas" : "Edit Tugas"
        );
```

```
            stage.setScene(new Scene(loader.load()));
            stage.initModality(javafx.stage.Modality.APPLICATION_MODAL);

            TambahTugasController controller = loader.getController();
            if (tugasToEdit != null) {
                controller.setTugasToEdit(tugasToEdit);
            }
            controller.setOnTugasAdded(() -> {
                loadTugasFromDatabase();
            });

            stage.showAndWait();
        } catch (Exception e) {
            showAlert("Error", "Gagal membuka form tugas: " + e.getMessage(),
Alert.AlertType.ERROR);
        }
    }

    private void loadTugasFromDatabase() {
        tugasList.clear();
        try (Connection conn = DatabaseConnection.getConnection()) {
            String sql = "SELECT id, judul, deskripsi, deadline, prioritas,
mata_kuliah, tipe, status FROM tugas";
            PreparedStatement st = conn.prepareStatement(sql);
            ResultSet rs = st.executeQuery();
            while (rs.next()) {
                Tugas tugas = new Tugas(
                        rs.getInt("id"),
                        rs.getString("judul"),
                        rs.getString("deskripsi"),
                        rs.getString("deadline"),
                        rs.getString("prioritas"),
                        rs.getString("mata_kuliah"),
                        rs.getString("tipe"),
                        rs.getString("status") == null ? "Belum Dikerjakan" :
rs.getString("status")
                );
                tugasList.add(tugas);
            }
        } catch (SQLException e) {
            showAlert("Database Error", "Gagal mengambil data tugas: " + e.
getMessage(), Alert.AlertType.ERROR);
        }

        filterTugas();
        updateSummaryCards();
    }

    private void filterTugas() {
        String search = searchField.getText() == null ? "" : searchField.
getText().toLowerCase();
        String selectedStatus = filterStatus.getValue();
        filteredTugas.setPredicate(tugas -> {
            boolean matchesStatus = selectedStatus == null || selectedStatus.
equals("Semua") || tugas.getStatus().equals(selectedStatus);
            boolean matchesSearch = search.isEmpty()
```

```
                    || (tugas.getJudul() != null && tugas.getJudul().
toLowerCase().contains(search))
                    || (tugas.getMataKuliah() != null && tugas.getMataKuliah()
.toLowerCase().contains(search))
                    || (tugas.getPrioritas() != null && tugas.getPrioritas().
toLowerCase().contains(search));
            return matchesStatus && matchesSearch;
        });
    }

    private void setupStatusColumn() {
        statusColumn.setCellFactory(param -> new TableCell<Tugas, String>() {
            private final ComboBox<String> statusComboBox = new ComboBox<>();

            {
                statusComboBox.getItems().addAll("Belum Dikerjakan", "Sedang
Dikerjakan", "Selesai");
                statusComboBox.setOnAction(e -> {
                    Tugas tugas = getTableView().getItems().get(getIndex());
                    String newStatus = statusComboBox.getValue();
                    if (newStatus != null && !newStatus.equals(tugas.getStatus
())) {
                        updateStatusInDatabase(tugas, newStatus);
                    }
                });
            }

            @Override
            protected void updateItem(String status, boolean empty) {
                super.updateItem(status, empty);
                if (empty || status == null) {
                    setGraphic(null);
                } else {
                    statusComboBox.setValue(status);
                    setGraphic(statusComboBox);
                }
            }
        });
    }

    private void updateStatusInDatabase(Tugas tugas, String newStatus) {
        try (Connection conn = DatabaseConnection.getConnection()) {
            String sql = "UPDATE tugas SET status = ? WHERE id = ?";
            PreparedStatement st = conn.prepareStatement(sql);
            st.setString(1, newStatus);
            st.setInt(2, tugas.getId());
            int rowsAffected = st.executeUpdate();

            if (rowsAffected > 0) {
                tugas.setStatus(newStatus);
                updateSummaryCards();
                filterTugas(); // Memastikan tabel terfilter dengan benar
setelah update
                showAlert("Sukses", "Status tugas berhasil diperbarui!", Alert
.AlertType.INFORMATION);
            }
        } catch (SQLException e) {
```

```
189             showAlert("Database Error", "Gagal mengupdate status: " + e.
    getMessage(), Alert.AlertType.ERROR);
190         }
191     }
192 
193     private void setupActionColumn() {
194         actionColumn.setCellFactory(param -> new TableCell<>() {
195             private final Button btnEdit = new Button("Edit");
196             private final Button btnHapus = new Button("Hapus");
197             private final HBox hbox = new HBox(8, btnEdit, btnHapus);
198 
199             {
200                 btnEdit.getStyleClass().addAll("action-button", "edit-button")
    ;
201                 btnHapus.getStyleClass().addAll("action-button", "delete-
    button");
202 
203                 btnEdit.setOnAction(e -> {
204                     Tugas tugas = getTableView().getItems().get(getIndex());
205                     openFormTugas(tugas);
206                 });
207 
208                 btnHapus.setOnAction(e -> {
209                     Tugas tugas = getTableView().getItems().get(getIndex());
210                     Alert confirmDialog = new Alert(Alert.AlertType.
    CONFIRMATION);
211                     confirmDialog.setTitle("Konfirmasi Hapus");
212                     confirmDialog.setHeaderText("Hapus Tugas");
213                     confirmDialog.setContentText("Apakah Anda yakin ingin
    menghapus tugas \"" + tugas.getJudul() + "\"?");
214                     confirmDialog.showAndWait().ifPresent(result -> {
215                         if (result == ButtonType.OK) {
216                             hapusTugas(tugas);
217                         }
218                     });
219                 });
220             }
221 
222             @Override
223             protected void updateItem(Void item, boolean empty) {
224                 super.updateItem(item, empty);
225                 setGraphic(empty ? null : hbox);
226             }
227         });
228     }
229 
230     private void hapusTugas(Tugas tugas) {
231         try (Connection conn = DatabaseConnection.getConnection()) {
232             String sql = "DELETE FROM tugas WHERE id = ?";
233             PreparedStatement st = conn.prepareStatement(sql);
234             st.setInt(1, tugas.getId());
235             st.executeUpdate();
236             tugasList.remove(tugas);
237             updateSummaryCards();
238             showAlert("Sukses", "Tugas berhasil dihapus!", Alert.AlertType.
    INFORMATION);
239         } catch (SQLException e) {
```

```
240             showAlert("Database Error", "Gagal menghapus tugas: " + e.
    getMessage(), Alert.AlertType.ERROR);
241         }
242     }
243 
244     private void updateSummaryCards() {
245         int urgent = 0, inProgress = 0, completed = 0;
246         LocalDate today = LocalDate.now();
247         DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd"
    );
248         StringBuilder urgentTasks = new StringBuilder();
249 
250         for (Tugas tugas : tugasList) {
251             try {
252                 String deadlineStr = tugas.getDeadline();
253                 if (deadlineStr == null || deadlineStr.trim().isEmpty()) {
254                     continue;
255                 }
256                 LocalDate deadline = deadlineStr.contains(" ") ? LocalDate.
    parse(deadlineStr.split(" ")[0], formatter) : LocalDate.parse(deadlineStr,
     formatter);
257                 long daysDiff = ChronoUnit.DAYS.between(today, deadline);
258 
259                 if (daysDiff <= 3 && daysDiff >= 0 && !"Selesai".
    equalsIgnoreCase(tugas.getStatus())) {
260                     urgent++;
261                     String formattedDeadline = deadline.atStartOfDay().format(
    DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"));
262                      if (deadlineStr.contains(":")) {
263                          formattedDeadline = tugas.getDeadline();
264                     }
265                     urgentTasks.append("Tugas: \"").append(tugas.getJudul()).
    append("\" deadline: ").append(formattedDeadline).append(" (").append(
    daysDiff).append(" hari lagi)\n");
266                 }
267             } catch (Exception e) {
268                 System.err.println("Gagal parsing deadline: " + tugas.
    getDeadline() + " untuk tugas '" + (tugas.getJudul() != null ? tugas.
    getJudul() : "N/A") + "' - " + e.getMessage());
269             }
270 
271             if ("Sedang Dikerjakan".equalsIgnoreCase(tugas.getStatus())) {
272                 inProgress++;
273             } else if ("Selesai".equalsIgnoreCase(tugas.getStatus())) {
274                 completed++;
275             }
276         }
277 
278         if (urgent > 0 && urgentTasks.length() > 0) {
279             showAlert("Pengingat Tugas", "Beberapa tugas memiliki deadline
    mendekati:\n\n" + urgentTasks.toString(), Alert.AlertType.WARNING);
280         }
281 
282         urgentCount.setText(String.valueOf(urgent));
283         inProgressCount.setText(String.valueOf(inProgress));
284         completedCount.setText(String.valueOf(completed));
285     }
```

```
286
287     private void showAlert(String title, String msg, Alert.AlertType type) {
288         Alert alert = new Alert(type);
289         alert.setTitle(title);
290         alert.setHeaderText(null);
291         alert.setContentText(msg);
292         alert.showAndWait();
293     }
294 }
295
```

Kode Sumber 1.12: DashboardController.java - Controller Dashboard Utama

File `TambahTugasController.java` Controller `TambahTugasController` bertanggung jawab untuk menangani logika pada form tambah dan edit tugas (`tambah_tugas.fxml`). Ini mencakup inisialisasi komponen UI seperti `TextField` untuk judul dan mata kuliah, `TextArea` untuk deskripsi, `DatePicker` untuk tanggal deadline, `ComboBox` untuk jam, menit, prioritas, tipe, dan status tugas. Jika tipe tugas adalah "Kelompok", controller mengaktifkan bagian untuk menambah anggota kelompok (`VBox vboxAnggotaKelompok`, `TextField txtNamaAnggota`, `txtNimAnggota`, `Button btnTambahAnggota`, `ListView listViewAnggota`). Metode `simpanTugas` menangani validasi input dan interaksi dengan database (insert atau update) menggunakan `DatabaseConnection`. Controller juga dapat mengisi form jika dalam mode edit (`setTugasToEdit`) dan memanggil callback (`onTugasAdded`) setelah tugas berhasil disimpan.

```
1  package com.coursemanagementsystem.controller;
2
3  import java.sql.Connection;
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.sql.Statement;
8  import java.sql.Timestamp;
9  import java.time.LocalDate;
10 import java.time.format.DateTimeFormatter;
11 import java.util.Optional;
12
13 import com.coursemanagementsystem.database.DatabaseConnection;
14 import com.coursemanagementsystem.model.Mahasiswa;
15 import com.coursemanagementsystem.model.Tugas;
16
17 import javafx.application.Platform;
18 import javafx.collections.FXCollections;
19 import javafx.collections.ObservableList;
20 import javafx.fxml.FXML;
21 import javafx.scene.control.Alert;
```

```
22 import javafx.scene.control.Button;
23 import javafx.scene.control.ButtonType;
24 import javafx.scene.control.ComboBox;
25 import javafx.scene.control.ContextMenu;
26 import javafx.scene.control.DateCell;
27 import javafx.scene.control.DatePicker;
28 import javafx.scene.control.ListCell;
29 import javafx.scene.control.ListView;
30 import javafx.scene.control.MenuItem;
31 import javafx.scene.control.TextArea;
32 import javafx.scene.control.TextField;
33 import javafx.scene.layout.VBox;
34 import javafx.stage.Stage;
35
36 public class TambahTugasController {
37
38     @FXML private TextField txtJudul;
39     @FXML private TextField txtMataKuliah;
40     @FXML private TextArea txtDeskripsi;
41     @FXML private DatePicker datePicker;
42     @FXML private ComboBox<String> cbJam;
43     @FXML private ComboBox<String> cbPrioritas;
44     @FXML private ComboBox<String> cbTipe;
45     @FXML private ComboBox<String> cbStatus;
46     @FXML private VBox vboxAnggotaKelompok;
47     @FXML private TextField txtNamaAnggota;
48     @FXML private TextField txtNimAnggota;
49     @FXML private Button btnTambahAnggota;
50     @FXML private ListView<String> listViewAnggota;
51     @FXML private Button btnSimpan;
52     @FXML private Button btnBatal;
53     @FXML private ComboBox<String> cbMenit;
54
55     private ObservableList<Mahasiswa> anggotaKelompok = FXCollections.
       observableArrayList();
56     private ObservableList<String> anggotaDisplayList = FXCollections.
       observableArrayList();
57     private Tugas tugasToEdit = null;
58     private Runnable onTugasAdded;
59     private boolean isInitialized = false;
60
61     public TambahTugasController() {}
62
63     @FXML
64     public void initialize() {
65         Platform.runLater(() -> {
66             try {
67                 initializeComponents();
68                 isInitialized = true;
69             } catch (Exception e) {
70                 e.printStackTrace();
71                 showAlert("Error", "Gagal inisialisasi form: " + e.getMessage
       (), Alert.AlertType.ERROR);
72             }
73         });
74     }
75
```

```
76      private void initializeComponents() {
77          if (cbPrioritas != null) {
78              cbPrioritas.getItems().clear();
79              cbPrioritas.getItems().addAll("Rendah", "Menengah", "Tinggi");
80              cbPrioritas.setValue("Rendah");
81          }
82
83          if (cbTipe != null) {
84              cbTipe.getItems().clear();
85              cbTipe.getItems().addAll("Individu", "Kelompok");
86              cbTipe.setValue("Individu");
87          }
88
89          if (cbStatus != null) {
90              cbStatus.getItems().clear();
91              cbStatus.getItems().addAll("Belum Dikerjakan", "Sedang Dikerjakan"
    , "Selesai");
92              cbStatus.setValue("Belum Dikerjakan");
93          }
94
95          initializeTimeComboBoxes();
96          initializeDatePicker();
97
98          if (listViewAnggota != null) {
99              listViewAnggota.setItems(anggotaDisplayList);
100             setupListView();
101         }
102
103         setupButtonActions();
104         setupTypeChangeListener();
105         handleTipeChange("Individu"); // default
106     }
107
108     private void initializeTimeComboBoxes() {
109         if (cbJam != null) {
110             cbJam.getItems().clear();
111             for (int i = 0; i < 24; i++) {
112                 cbJam.getItems().add(String.format("%02d", i));
113             }
114             cbJam.setValue("00");
115         }
116
117         if (cbMenit != null) {
118             cbMenit.getItems().clear();
119             for (int i = 0; i < 60; i += 5) {
120                 cbMenit.getItems().add(String.format("%02d", i));
121             }
122             cbMenit.setValue("00");
123         }
124     }
125
126     private void initializeDatePicker() {
127         if (datePicker != null) {
128             datePicker.setConverter(new javafx.util.StringConverter<LocalDate
    >() {
129                 private DateTimeFormatter dateFormatter = DateTimeFormatter.
    ofPattern("dd/MM/yyyy");
```

```
130
131                 @Override
132                 public String toString(LocalDate date) {
133                     return (date != null) ? dateFormatter.format(date) : "";
134                 }
135
136                 @Override
137                 public LocalDate fromString(String string) {
138                     if (string != null && !string.isEmpty()) {
139                         try {
140                             return LocalDate.parse(string, dateFormatter);
141                         } catch (Exception e) { return null; }
142                     }
143                     return null;
144                 }
145             });
146
147             datePicker.setDayCellFactory(picker -> new DateCell() {
148                 @Override
149                 public void updateItem(LocalDate date, boolean empty) {
150                     super.updateItem(date, empty);
151                     setDisable(empty || date.isBefore(LocalDate.now()));
152                 }
153             });
154             datePicker.setValue(LocalDate.now());
155         }
156     }
157
158     private void setupButtonActions() {
159         if (btnTambahAnggota != null) btnTambahAnggota.setOnAction(e ->
    tambahAnggotaKelompok());
160         if (btnSimpan != null) btnSimpan.setOnAction(e -> simpanTugas());
161         if (btnBatal != null) btnBatal.setOnAction(e -> close());
162     }
163
164     private void setupTypeChangeListener() {
165         if (cbTipe != null) {
166             cbTipe.valueProperty().addListener((obs, oldVal, newVal) -> {
167                 if (newVal != null) handleTipeChange(newVal);
168             });
169         }
170     }
171
172     private void handleTipeChange(String newType) {
173         boolean isKelompok = "Kelompok".equals(newType);
174         if (vboxAnggotaKelompok != null) {
175             vboxAnggotaKelompok.setVisible(isKelompok);
176             vboxAnggotaKelompok.setManaged(isKelompok);
177         }
178         if (!isKelompok) {
179             anggotaKelompok.clear();
180             anggotaDisplayList.clear();
181         }
182     }
183
184     private void setupListView() {
185         if (listViewAnggota != null) {
```

```
            listViewAnggota.setCellFactory(lv -> {
                ListCell<String> cell = new ListCell<String>() {
                    @Override
                    protected void updateItem(String item, boolean empty) {
                        super.updateItem(item, empty);
                        setText(empty || item == null ? null : item);
                    }
                };

                ContextMenu contextMenu = new ContextMenu();
                MenuItem deleteItem = new MenuItem("       Hapus");
                deleteItem.setOnAction(e -> {
                    int index = cell.getIndex();
                    if (index >= 0 && index < anggotaKelompok.size()) {
                        String nama = anggotaKelompok.get(index).getNama();
                        Alert confirm = new Alert(Alert.AlertType.CONFIRMATION
, "Yakin hapus " + nama + "?", ButtonType.OK, ButtonType.CANCEL);
                        confirm.setTitle("Hapus Anggota");
                        confirm.setHeaderText(null);
                        confirm.showAndWait().ifPresent(response -> {
                            if (response == ButtonType.OK) {
                                anggotaKelompok.remove(index);
                                anggotaDisplayList.remove(index);
                            }
                        });
                    }
                });
                contextMenu.getItems().add(deleteItem);
                cell.emptyProperty().addListener((obs, wasEmpty, isNowEmpty)
-> {
                    cell.setContextMenu(isNowEmpty ? null : contextMenu);
                });
                return cell;
            });
        }
    }

    private void tambahAnggotaKelompok() {
        String nama = (txtNamaAnggota != null) ? txtNamaAnggota.getText().trim
() : "";
        String nim = (txtNimAnggota != null) ? txtNimAnggota.getText().trim()
: "";

        if (nama.isEmpty() || nim.isEmpty()) {
            showAlert("Validasi", "Nama dan NIM anggota harus diisi!", Alert.
AlertType.WARNING);
            return;
        }
        if (anggotaKelompok.stream().anyMatch(a -> a.getNim().equals(nim))) {
            showAlert("Validasi", "NIM sudah ada di daftar anggota!", Alert.
AlertType.WARNING);
            return;
        }
        anggotaKelompok.add(new Mahasiswa(0, nama, nim, "", "")); // ID, Email
, Password default
        anggotaDisplayList.add("       " + nama + " (" + nim + ")");
        if (txtNamaAnggota != null) txtNamaAnggota.clear();
```

```
236         if (txtNimAnggota != null) txtNimAnggota.clear();
237     }
238 
239     private void simpanTugas() {
240         String judul = (txtJudul != null) ? txtJudul.getText().trim() : "";
241         String deskripsi = (txtDeskripsi != null) ? txtDeskripsi.getText().
    trim() : "";
242         LocalDate tanggal = (datePicker != null) ? datePicker.getValue() :
    null;
243         String jam = (cbJam != null) ? cbJam.getValue() : null;
244         String menit = (cbMenit != null) ? cbMenit.getValue() : null;
245         String prioritas = (cbPrioritas != null) ? cbPrioritas.getValue() :
    null;
246         String matkul = (txtMataKuliah != null) ? txtMataKuliah.getText().trim
    () : "";
247         String tipe = (cbTipe != null) ? cbTipe.getValue() : null;
248         String status = (cbStatus != null) ? cbStatus.getValue() : null;
249 
250         if (judul.isEmpty() || tanggal == null || jam == null || menit == null
     || prioritas == null || matkul.isEmpty() || tipe == null || status ==
    null) {
251             showAlert("Validasi", "Semua field wajib diisi!", Alert.AlertType.
    WARNING);
252             return;
253         }
254         if ("Kelompok".equals(tipe) && anggotaKelompok.isEmpty()) {
255             showAlert("Validasi", "Tugas kelompok memerlukan minimal 1 anggota
    .", Alert.AlertType.WARNING);
256             return;
257         }
258 
259         try {
260             Timestamp deadline = Timestamp.valueOf(tanggal.toString() + " " +
    jam + ":" + menit + ":00");
261             try (Connection conn = DatabaseConnection.getConnection()) {
262                 conn.setAutoCommit(false);
263                 if (tugasToEdit == null) {
264                     insertNewTask(conn, judul, deskripsi, deadline, prioritas,
     matkul, tipe, status);
265                 } else {
266                     updateExistingTask(conn, judul, deskripsi, deadline,
    prioritas, matkul, tipe, status);
267                 }
268                 conn.commit();
269                 if (onTugasAdded != null) onTugasAdded.run();
270                 close();
271             } catch (SQLException e) {
272                 e.printStackTrace();
273                 showAlert("Error Database", "Gagal menyimpan tugas ke database
    : " + e.getMessage(), Alert.AlertType.ERROR);
274             }
275         } catch (Exception e) {
276             e.printStackTrace();
277             showAlert("Error Sistem", "Gagal memproses penyimpanan tugas: " +
    e.getMessage(), Alert.AlertType.ERROR);
278         }
279     }
```

```
280
281     private void insertNewTask(Connection conn, String j, String d, Timestamp
        dl, String p, String mk, String t, String s) throws SQLException {
282         String sql = "INSERT INTO tugas (judul, deskripsi, deadline, prioritas
        , mata_kuliah, tipe, status) VALUES (?, ?, ?, ?, ?, ?, ?)";
283         try (PreparedStatement st = conn.prepareStatement(sql, Statement.
        RETURN_GENERATED_KEYS)) {
284             st.setString(1, j); st.setString(2, d); st.setTimestamp(3, dl); st
        .setString(4, p);
285             st.setString(5, mk); st.setString(6, t); st.setString(7, s);
286             st.executeUpdate();
287             try (ResultSet rs = st.getGeneratedKeys()) {
288                 if (rs.next() && "Kelompok".equals(t) && !anggotaKelompok.
        isEmpty()) {
289                     insertGroupMembers(conn, rs.getInt(1));
290                 }
291             }
292         }
293         showAlert("Sukses", "Tugas berhasil ditambahkan!", Alert.AlertType.
        INFORMATION);
294     }
295
296     private void updateExistingTask(Connection conn, String j, String d,
        Timestamp dl, String p, String mk, String t, String s) throws SQLException
         {
297         String sql = "UPDATE tugas SET judul=?, deskripsi=?, deadline=?,
        prioritas=?, mata_kuliah=?, tipe=?, status=? WHERE id=?";
298         try (PreparedStatement st = conn.prepareStatement(sql)) {
299             st.setString(1, j); st.setString(2, d); st.setTimestamp(3, dl); st
        .setString(4, p);
300             st.setString(5, mk); st.setString(6, t); st.setString(7, s); st.
        setInt(8, tugasToEdit.getId());
301             st.executeUpdate();
302         }
303         // Hapus anggota lama dan masukkan yang baru (jika ada perubahan atau
        tugas kelompok)
304         try (PreparedStatement del = conn.prepareStatement("DELETE FROM
        anggota_kelompok WHERE tugas_id=?")) {
305             del.setInt(1, tugasToEdit.getId());
306             del.executeUpdate();
307         }
308         if ("Kelompok".equals(t) && !anggotaKelompok.isEmpty()) {
309             insertGroupMembers(conn, tugasToEdit.getId());
310         }
311         showAlert("Sukses", "Tugas berhasil diperbarui!", Alert.AlertType.
        INFORMATION);
312     }
313
314     private void insertGroupMembers(Connection conn, int tugasId) throws
        SQLException {
315         String sql = "INSERT INTO anggota_kelompok (tugas_id, nama, nim)
        VALUES (?, ?, ?)";
316         try (PreparedStatement st = conn.prepareStatement(sql)) {
317             for (Mahasiswa m : anggotaKelompok) {
318                 st.setInt(1, tugasId); st.setString(2, m.getNama()); st.
        setString(3, m.getNim());
319                 st.addBatch();
```

```
320             }
321             st.executeBatch();
322         }
323     }
324 
325     public void setTugasToEdit(Tugas tugas) {
326         this.tugasToEdit = tugas;
327         if (!isInitialized) {
328             Platform.runLater(() -> setTugasToEdit(tugas));
329             return;
330         }
331         if (tugas != null) {
332             if (btnSimpan != null) btnSimpan.setText("Update");
333             if (txtJudul != null) txtJudul.setText(tugas.getJudul());
334             if (txtDeskripsi != null) txtDeskripsi.setText(tugas.getDeskripsi
    ());
335             if (txtMataKuliah != null) txtMataKuliah.setText(tugas.
    getMataKuliah());
336             if (cbPrioritas != null) cbPrioritas.setValue(tugas.getPrioritas()
    );
337             if (cbTipe != null) cbTipe.setValue(tugas.getTipe());
338             if (cbStatus != null) cbStatus.setValue(tugas.getStatus());
339             parseAndSetDeadline(tugas.getDeadline());
340             if ("Kelompok".equals(tugas.getTipe())) {
341                 handleTipeChange("Kelompok"); // Pastikan UI untuk anggota
    terlihat
342                 loadAnggotaKelompok(tugas.getId());
343             } else {
344                 handleTipeChange("Individu"); // Sembunyikan UI anggota jika
    bukan kelompok
345             }
346         }
347     }
348 
349     private void parseAndSetDeadline(String deadlineStr) {
350         try {
351             if (deadlineStr != null && !deadlineStr.isEmpty()) {
352                 Timestamp deadline = Timestamp.valueOf(deadlineStr); // Format
     yyyy-mm-dd hh:mm:ss.fffffffff
353                 LocalDate tanggal = deadline.toLocalDateTime().toLocalDate();
354                 String jam = String.format("%02d", deadline.toLocalDateTime().
    getHour());
355                 String menit = String.format("%02d", deadline.toLocalDateTime
    ().getMinute());
356                 if (datePicker != null) datePicker.setValue(tanggal);
357                 if (cbJam != null) cbJam.setValue(jam);
358                 if (cbMenit != null) cbMenit.setValue(menit);
359             }
360         } catch (IllegalArgumentException e) {
361             System.err.println("Error parsing deadline string '" + deadlineStr
     + "': " + e.getMessage());
362             // Mungkin tampilkan alert ke pengguna jika format salah
363             showAlert("Error Format Deadline", "Format deadline tidak valid: "
     + deadlineStr + ". Harap gunakan format yyyy-MM-dd HH:mm:ss.", Alert.
    AlertType.ERROR);
364         }
365     }
```

```
366
367     private void loadAnggotaKelompok(int tugasId) {
368         anggotaKelompok.clear();
369         anggotaDisplayList.clear();
370         String sql = "SELECT nama, nim FROM anggota_kelompok WHERE tugas_id=?"
    ;
371         try (Connection conn = DatabaseConnection.getConnection();
372              PreparedStatement st = conn.prepareStatement(sql)) {
373             st.setInt(1, tugasId);
374             try (ResultSet rs = st.executeQuery()) {
375                 while (rs.next()) {
376                     Mahasiswa m = new Mahasiswa(0, rs.getString("nama"), rs.
    getString("nim"), "", "");
377                     anggotaKelompok.add(m);
378                     anggotaDisplayList.add("       " + m.getNama() + " (" + m.
    getNim() + ")");
379                 }
380             }
381         } catch (SQLException e) {
382             e.printStackTrace();
383             showAlert("Error Database", "Gagal memuat anggota kelompok: " + e.
    getMessage(), Alert.AlertType.ERROR);
384         }
385     }
386
387     public void setOnTugasAdded(Runnable callback) {
388         this.onTugasAdded = callback;
389     }
390
391     private void close() {
392         try {
393             if (btnBatal != null && btnBatal.getScene() != null && btnBatal.
    getScene().getWindow() != null) {
394                 ((Stage) btnBatal.getScene().getWindow()).close();
395             }
396         } catch (Exception e) {
397             e.printStackTrace(); // Log error jika gagal menutup stage
398         }
399     }
400
401     private void showAlert(String title, String msg, Alert.AlertType type) {
402         try {
403             Alert alert = new Alert(type);
404             alert.setTitle(title);
405             alert.setHeaderText(null);
406             alert.setContentText(msg);
407             alert.showAndWait();
408         } catch (Exception e) { // Menangkap Exception yang lebih umum
409             e.printStackTrace();
410             System.err.println("ALERT DISPLAY ERROR: Title=" + title + ", Msg=
    " + msg + ", Type=" + type + " --- " + e.getMessage());
411         }
412     }
413 }
```

```
414
```

Kode Sumber 1.13: TambahTugasController.java - Controller Form Tugas

Styling Aplikasi

File `style.css` File `style.css` ini bertanggung jawab untuk mendefinisikan tampilan visual dari berbagai elemen antarmuka pengguna dalam aplikasi JavaFX. Ini mencakup styling untuk elemen root seperti latar belakang dan font default, bagian header (judul utama dan subjudul), area pencarian (`.search-field`), tombol tambah (`.add-button`), kartu statistik (`.stat-card`, `.stat-number`, `.stat-label`), kontainer dan elemen tabel (`.table-view`, `.column-header`, `.table-cell`), ComboBox filter (`.filter-combo`), serta tombol aksi di dalam tabel (`.action-button`). Gaya spesifik juga diterapkan untuk status prioritas dan status pengerjaan tugas, serta scrollbar. File CSS ini dimuat oleh kelas `MainApp` untuk diterapkan pada scene utama aplikasi.

```
/* Root styling - background bersih */
.root {
    -fx-background-color: linear-gradient(to bottom, #f8fafc, #f1f5f9);
    -fx-font-family: "Segoe UI", "System", sans-serif;
    -fx-font-size: 14px;
}

/* Header section */
.header-section {
    -fx-spacing: 8;
    -fx-padding: 0 0 20 0;
    -fx-alignment: center;
}

.main-title {
    -fx-font-size: 32px;
    -fx-font-weight: bold;
    -fx-text-fill: #1a202c;
}

.subtitle {
    -fx-font-size: 16px;
    -fx-text-fill: #718096;
    -fx-font-weight: normal;
}

/* Search section */
.search-section {
    -fx-spacing: 15;
    -fx-alignment: center-left;
    -fx-padding: 0 0 20 0;
```

```
}

.search-field {
    -fx-background-color: #ffffff;
    -fx-border-color: #3182ce;
    -fx-border-width: 2;
    -fx-border-radius: 8;
    -fx-background-radius: 8;
    -fx-padding: 12 16;
    -fx-font-size: 15px;
    -fx-pref-width: 800;
    -fx-pref-height: 45;
}

.search-field:focused {
    -fx-border-color: #2c5282;
    -fx-effect: dropshadow(gaussian, rgba(49, 130, 206, 0.3), 5, 0, 0, 0);
}

/* Add button */
.add-button {
    -fx-background-color: #e2e8f0;
    -fx-text-fill: #4a5568;
    -fx-font-weight: normal;
    -fx-padding: 12 20;
    -fx-border-radius: 8;
    -fx-background-radius: 8;
    -fx-cursor: hand;
    -fx-font-size: 14px;
    -fx-pref-height: 45;
}

.add-button:hover {
    -fx-background-color: #cbd5e0;
}

/* Stat cards */
.stat-card {
    -fx-background-color: #ffffff;
    -fx-padding: 24;
    -fx-background-radius: 12;
    -fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.08), 8, 0, 0, 2);
    -fx-spacing: 8;
    -fx-pref-width: 280;
    -fx-pref-height: 120;
}

.stat-card:hover {
    -fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.12), 12, 0, 0, 4);
}

.stat-number {
    -fx-font-size: 40px;
    -fx-font-weight: bold;
}

.urgent-number {
```

```
    -fx-text-fill: #e53e3e;
}

.progress-number {
    -fx-text-fill: #3182ce;
}

.completed-number {
    -fx-text-fill: #38a169;
}

.stat-label {
    -fx-font-size: 16px;
    -fx-font-weight: bold;
    -fx-padding: 4 0 0 0;
}

.urgent-label {
    -fx-text-fill: #e53e3e;
}

.progress-label {
    -fx-text-fill: #3182ce;
}

.completed-label {
    -fx-text-fill: #38a169;
}

.stat-description {
    -fx-text-fill: #718096;
    -fx-font-size: 13px;
    -fx-padding: 2 0 0 0;
}

/* Table container */
.table-container {
    -fx-padding: 0;
    -fx-spacing: 0;
}

.title-filter-hbox {
    -fx-spacing: 20;
    -fx-alignment: center-left;
}

.section-title {
    -fx-font-size: 20px;
    -fx-font-weight: bold;
    -fx-text-fill: #2d3748;
}

.filter-combo {
    -fx-background-color: #ffffff;
    -fx-border-color: #cbd5e0;
    -fx-border-width: 1;
    -fx-border-radius: 6;
```

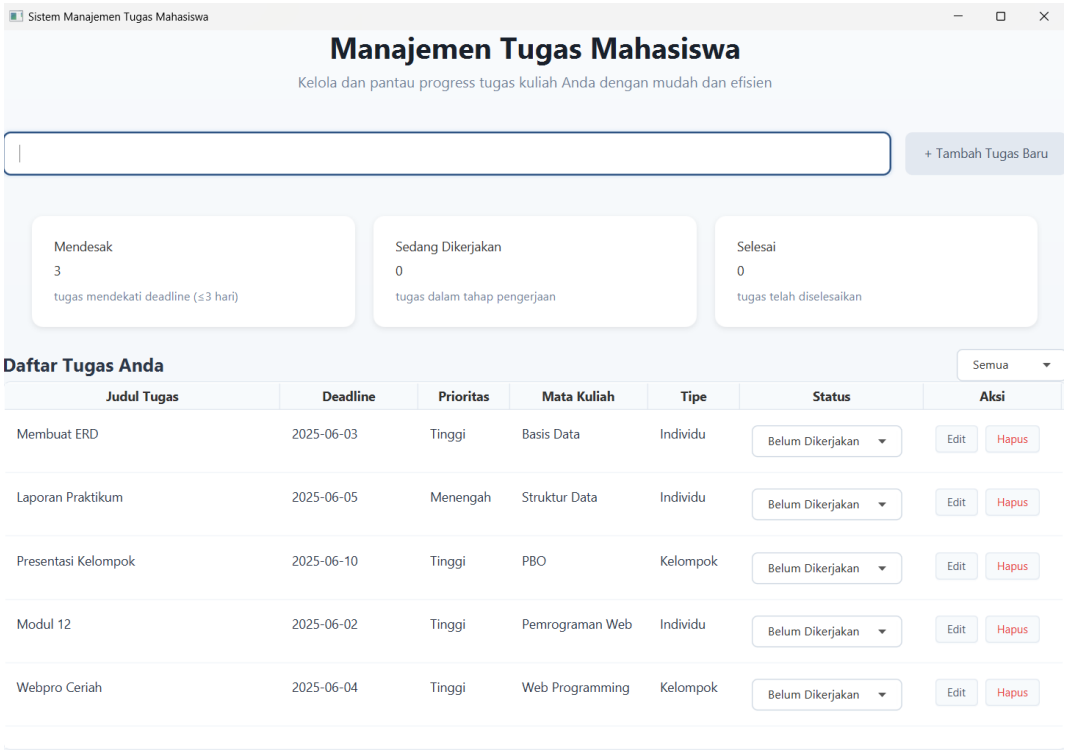
```
    -fx-background-radius: 6;
    -fx-padding: 8 12;
    -fx-font-size: 14px;
    -fx-pref-width: 120;
}

/* Table styling */
.table-view {
    -fx-background-color: #ffffff;
    -fx-border-color: #e2e8f0;
    -fx-border-width: 1;
    -fx-border-radius: 8;
    -fx-background-radius: 8;
}

.table-view .column-header-background {
    -fx-background-color: #f7fafc;
    -fx-border-color: #e2e8f0;
    -fx-border-width: 0 0 1 0;
}

.table-view .column-header {
    -fx-background-color: transparent;
    -fx-padding: 16 12;
    -fx-font-weight: bold;
    -fx-text-fill: #4a5568;
    -fx-font-size: 14px;
    -fx-border-color: #e2e8f0;
    -fx-border-width: 0 1 0 0;
}

.table-view .filler {
    -fx-background-color: #f7fafc;
}

.table-view .table-cell {
    -fx-padding: 16 12;
    -fx-text-fill: #2d3748;
    -fx-font-size: 14px;
    -fx-border-color: #f1f5f9;
    -fx-border-width: 0 0 1 0;
    -fx-background-color: #ffffff;
}

.table-row-cell {
    -fx-background-color: #ffffff;
    -fx-border-color: transparent;
}

.table-row-cell:hover {
    -fx-background-color: #f7fafc;
}

.table-row-cell:selected {
    -fx-background-color: #ebf8ff;
}

```

```
.table-row-cell:selected .table-cell {
    -fx-background-color: transparent;
}

/* Priority styling */
.priority-tinggi {
    -fx-text-fill: #e53e3e;
}

.priority-menengah {
    -fx-text-fill: #d69e2e;
}

.priority-rendah {
    -fx-text-fill: #38a169;
}

/* Status styling */
.status-belum {
    -fx-text-fill: #718096;
}

.status-progress {
    -fx-text-fill: #3182ce;
}

.status-selesai {
    -fx-text-fill: #38a169;
}

/* Action buttons */
.action-button {
    -fx-background-color: #f7fafc;
    -fx-text-fill: #4a5568;
    -fx-padding: 6 12;
    -fx-background-radius: 4;
    -fx-font-size: 12px;
    -fx-cursor: hand;
    -fx-border-color: #e2e8f0;
    -fx-border-width: 1;
    -fx-border-radius: 4;
}

.action-button:hover {
    -fx-background-color: #edf2f7;
}

.edit-button {
    -fx-text-fill: #4a5568;
}

.delete-button {
    -fx-text-fill: #e53e3e;
}

.delete-button:hover {
    -fx-background-color: #fed7d7;
```

```
}

/* ComboBox (umum) */
.combo-box {
    -fx-background-color: #ffffff;
    -fx-border-color: #cbd5e0;
    -fx-border-radius: 6;
    -fx-background-radius: 6;
    -fx-padding: 4 8;
    -fx-font-size: 13px;
}

/* Scrollbar */
.scroll-bar:vertical {
    -fx-background-color: transparent;
    -fx-pref-width: 10;
}

.scroll-bar:vertical .track {
    -fx-background-color: #f1f5f9;
    -fx-background-radius: 5;
}

.scroll-bar:vertical .thumb {
    -fx-background-color: #cbd5e1;
    -fx-background-radius: 5;
}

.scroll-bar:vertical .thumb:hover {
    -fx-background-color: #a0aec0;
}

/* Stats horizontal layout */
.stats-hbox {
    -fx-spacing: 20;
    -fx-alignment: center;
}
```

Kode Sumber 1.14: style.css - Styling Aplikasi

Screenshot Tampilan Dashboard Manajemen Tugas Mahasiswa



Gambar 1.1: Dashboard Manajemen Tugas Mahasiswa

1.4 Hasil dan Pembahasan

Bagian ini membahas hasil dari implementasi sistem manajemen tugas mahasiswa yang telah dikembangkan selama praktikum. Pembahasan mencakup verifikasi fungsionalitas utama, analisis struktur aplikasi, implementasi model data, fitur-fitur yang berhasil diimplementasikan pada dashboard, serta pengelolaan data tugas melalui form.

1.4.1 Verifikasi Fungsionalitas Inti

Implementasi sistem manajemen tugas mahasiswa telah berhasil mencapai fungsionalitas inti yang direncanakan. Operasi *Create, Read, Update, Delete* (CRUD) untuk data tugas dapat dijalankan melalui antarmuka pengguna grafis yang disediakan. Pengguna dapat menambahkan tugas baru, melihat daftar tugas yang ada, memperbarui detail tugas, serta menghapus tugas dari sistem. Keberhasilan fung-

sionalitas ini didukung oleh implementasi pada `DashboardController.java` yang menangani penampilan dan interaksi dasar dengan data tugas, serta `TambahTugasController.java` yang mengelola form untuk input dan modifikasi data.

Konektivitas dengan database MySQL juga telah berhasil diimplementasikan dan diuji. Kelas `DatabaseConnection.java` menyediakan mekanisme untuk terhubung ke database, dan pengujian koneksi melalui `TestKoneksi.java` menunjukkan bahwa koneksi dapat dibangun dengan sukses. Seluruh operasi CRUD yang melibatkan persistensi data ke database berjalan sesuai harapan.

1.4.2 Struktur dan Arsitektur Aplikasi

Aplikasi dikembangkan menggunakan JavaFX untuk antarmuka pengguna grafis, dengan mengikuti pola yang memisahkan antara tampilan (didefinisikan dalam file FXML), logika kontrol (Controller), dan model data. Titik masuk utama aplikasi adalah kelas `MainApp.java`, yang bertanggung jawab untuk memuat tampilan awal (`dashboard.fxml`) dan menerapkan styling global melalui file `style.css`.

Penggunaan modul Java, sebagaimana didefinisikan dalam `module-info.java`, memastikan bahwa dependensi terhadap pustaka JavaFX (seperti `javafx.controls`, `javafx.fxml`) dan pustaka lain (seperti `java.sql`) terdefinisi dengan baik. Pembukaan (opens) dan pengeksporan (exports) paket yang relevan memungkinkan kerangka kerja JavaFX untuk mengakses dan mengelola komponen aplikasi dengan benar.

1.4.3 Implementasi Model Data

Model data aplikasi dirancang untuk merepresentasikan entitas-entitas utama dalam sistem. Kelas `Tugas.java` berfungsi sebagai kelas dasar untuk semua jenis tugas, memanfaatkan JavaFX Properties untuk atribut-atributnya. Hal ini terbukti sangat berguna untuk integrasi dengan komponen UI seperti `TableView` di `DashboardController.java`, memungkinkan pembaruan UI secara otomatis ketika data di model berubah.

Spesialisasi tugas diimplementasikan melalui subkelas `TugasIndividu.java` dan `TugasKelompok.java`. `TugasKelompok.java` juga mengelola daftar

objek `Mahasiswa.java` sebagai anggota kelompok. Implementasi interface `Notifikasi.java` pada kedua subkelas tugas menunjukkan penerapan konsep polimorfisme, meskipun fungsionalitas pengingat yang diimplementasikan saat ini bersifat sederhana (output ke konsol).

1.4.4 Analisis Fitur Dashboard

Dashboard utama, yang dikontrol oleh `DashboardController.java`, berhasil menampilkan data tugas dalam `TableView` yang interaktif. Beberapa fitur kunci yang berhasil diimplementasikan dan diuji meliputi:

- **Visualisasi Data Tugas:** Daftar tugas ditampilkan dengan jelas beserta atribut-atribut penting seperti judul, deadline, prioritas, mata kuliah, tipe, dan status.
- **Pencarian dan Filter:** Fitur pencarian memungkinkan pengguna untuk dengan cepat menemukan tugas berdasarkan kata kunci pada judul, mata kuliah, atau prioritas. Filter berdasarkan status (Semua, Belum Dikerjakan, Sedang Dikerjakan, Selesai) juga berfungsi dengan baik untuk menyaring tampilan data.
- **Update Status Interaktif:** Pengguna dapat mengubah status tugas secara langsung dari `TableView` melalui `ComboBox` yang terintegrasi dalam sel status. Perubahan ini langsung tersimpan ke database dan memperbarui tampilan.
- **Manajemen Tugas (Edit dan Hapus):** Tombol aksi "Edit" dan "Hapus" pada setiap baris tugas memungkinkan pengguna untuk memodifikasi detail tugas atau menghapusnya dari sistem setelah konfirmasi.
- **Ringkasan Tugas dan Pengingat:** Kartu statistik pada dashboard memberikan ringkasan visual mengenai jumlah tugas yang mendesak (mendekati deadline), sedang dikerjakan, dan telah selesai. Implementasi pengingat untuk tugas yang mendekati deadline (dalam 3 hari) melalui `Alert` juga berhasil dilakukan, meningkatkan kesadaran pengguna terhadap tenggat waktu.

1.4.5 Pengelolaan Data melalui Form Tambah/Edit Tugas

Formulir untuk menambah dan mengedit tugas, yang dikelola oleh `TambahTugasController.java`, menyediakan antarmuka yang komprehensif untuk input data. Penggunaan komponen JavaFX seperti `DatePicker`, `ComboBox` untuk waktu, prioritas, tipe, dan status, serta `TextField` dan `TextArea` telah diimplementasikan dengan benar. Validasi input dasar untuk memastikan semua field wajib diisi telah diterapkan sebelum data disimpan ke database.

Untuk tugas bertipe "Kelompok", fungsionalitas untuk menambah dan menghapus anggota kelompok dari `ListView` juga berjalan sesuai rencana. Data anggota kelompok disimpan dan diambil dari tabel `anggota_kelompok` di database. Logika untuk menampilkan atau menyembunyikan bagian input anggota kelompok berdasarkan tipe tugas yang dipilih juga berhasil diimplementasikan.

1.4.6 Tantangan dan Potensi Pengembangan

Selama implementasi, beberapa tantangan minor mungkin muncul, seperti penanganan format tanggal dan waktu dari `DatePicker` dan `ComboBox` untuk disimpan sebagai `Timestamp` di database, atau memastikan sinkronisasi data yang konsisten antara `ObservableList` dan database setelah operasi CRUD. Namun, tantangan-tantangan ini berhasil diatasi.

Potensi pengembangan lebih lanjut untuk sistem ini meliputi:

- Implementasi sistem login pengguna dan manajemen hak akses.
- Fitur notifikasi yang lebih canggih (misalnya, email atau notifikasi desktop).
- Kemampuan untuk mengunggah file terkait tugas.
- Fitur kolaborasi yang lebih mendalam untuk tugas kelompok.
- Peningkatan validasi input dan penanganan error yang lebih robas.
- Implementasi fungsionalitas *dark mode* atau tema kustom (merujuk pada `toggleThemeBtn` yang ada di FXML namun belum diim-

plementasikan sepenuhnya di `DashboardController.java`).

Secara keseluruhan, hasil implementasi menunjukkan bahwa sistem manajemen tugas mahasiswa ini telah berhasil dikembangkan dengan fungsionalitas dasar yang solid dan antarmuka pengguna yang responsif.

1.5 Kesimpulan

Implementasi sistem manajemen tugas mahasiswa ini telah berhasil mencakup fungsionalitas inti seperti penambahan, pengeditan, penghapusan, dan penampilan daftar tugas. Koneksi ke database MySQL telah berhasil diimplementasikan dan diuji. Antarmuka pengguna (UI) dirancang menggunakan JavaFX dan FXML, dengan styling yang diterapkan melalui CSS untuk meningkatkan pengalaman pengguna.

Model data telah didefinisikan untuk merepresentasikan Tugas (dengan variasi Individu dan Kelompok) serta Mahasiswa, memanfaatkan JavaFX Properties untuk kemudahan binding data ke UI. Controller aplikasi mengelola logika bisnis, interaksi antara view dan model, serta operasi CRUD ke database.

Fitur-fitur utama yang diimplementasikan pada dashboard meliputi:

- Penampilan daftar tugas dalam bentuk tabel.
- Kemampuan untuk menambah, mengedit, dan menghapus tugas.
- Pembaruan status tugas secara langsung dari tabel.
- Filter tugas berdasarkan status.
- Pencarian tugas berdasarkan judul, mata kuliah, atau prioritas.
- Ringkasan jumlah tugas berdasarkan status (urgent/mendekati deadline, sedang dikerjakan, selesai).
- Pengingat visual untuk tugas-tugas yang mendekati deadline.

Form tambah/edit tugas juga mendukung input detail tugas, termasuk manajemen anggota untuk tugas kelompok. Validasi input dasar dan penanganan kesalahan juga telah dipertimbangkan dalam implementasi controller.

Secara keseluruhan, aplikasi ini menyediakan platform dasar yang fungsional untuk manajemen tugas mahasiswa.