PRAKTIKUM REKAYASA PERANGKAT LUNAK 2

MANUAL BOOK

***"Daily Journal"* Menggunakan Netbeans**

Nama Anggota :

1. **Kautsar Hasby Dastien Fredila (50422770)**

Kelas : 4IA20
Fakultas : Teknologi Industri
Jurusan : Teknik Informatika
PJ : *Rizky Putra Perdana*

Ditulis Guna Melengkapi Sebagian Syarat Praktikum Rekayasa Perangkat Lunak 2
Jenjang S1
Universitas Gunadarma
2025

BAB I
PENDAHULUA
N

1.1 Latar Belakang

Perkembangan Teknologi Informasi saat ini sangatlah cepat. Teknologi Informasi memegang peranan yang sangat penting dalam kehidupan manusia. Batas-batas yang muncul karena perbedaan ruang dan waktu kini dapat ditembus melalui kehadiran teknologi, khususnya internet. Berbagai aktivitas seperti belanja, ujian, membaca berita, hingga melihat katalog buku kini dapat dilakukan secara online. Namun, semakin kompleks suatu sistem teknologi, semakin rumit pula proses pengembangan serta pemeliharaannya. Oleh karena itu, diperlukan pemahaman yang baik mengenai perancangan sistem agar aplikasi yang dibuat mampu bekerja secara efektif dan efisien.

Dalam konteks pengembangan perangkat lunak, aplikasi desktop tetap memiliki peran penting meskipun tren aplikasi berbasis web dan mobile semakin meningkat. Aplikasi desktop menawarkan stabilitas, performa yang konsisten, serta kemudahan akses tanpa memerlukan koneksi internet. Berdasarkan kebutuhan tersebut, dibuatlah sebuah program desktop *Daily Journal* menggunakan NetBeans sebagai lingkungan pengembangan. Aplikasi ini dirancang untuk membantu pengguna dalam mencatat, mengelola, serta menyimpan data jurnal secara terstruktur dan aman. Dengan pendekatan berbasis Model-View-Controller (MVC), aplikasi ini tidak hanya mudah dikembangkan, tetapi juga mudah dipelihara dan dikembangkan lebih lanjut.

Aplikasi *Daily Journal* ini hadir sebagai solusi sederhana namun fungsional untuk kebutuhan pencatatan harian, dokumentasi kegiatan, maupun pengelolaan catatan akademik atau pekerjaan dengan dibuatkannya fitur emotion. Melalui fitur-fitur seperti penambahan, pengeditan, dan penghapusan entri, aplikasi ini memberikan pengalaman penggunaan yang intuitif. Implementasi aplikasi menggunakan Java dan NetBeans juga memungkinkan aplikasi berjalan lintas platform, sehingga dapat digunakan pada berbagai sistem operasi. Dengan adanya aplikasi ini, pengguna diharapkan dapat mengelola catatan jurnal secara lebih efektif dan terorganisir.

1.2 Tujuan

Tujuan dari pembuatan aplikasi Daily Journal ini adalah untuk menyediakan media pencatatan harian yang mudah digunakan, terstruktur, dan dapat membantu pengguna dalam mendokumentasikan aktivitas, perasaan, dan pengalaman sehari-hari. Aplikasi ini dirancang untuk mempermudah pengguna dalam menulis, menyimpan, mengedit, serta melihat kembali catatan mereka dengan tampilan yang sederhana namun informatif. Selain itu, aplikasi ini bertujuan untuk membantu pengguna mengenali pola emosi melalui fitur pemilihan mood, serta menyediakan sistem penyimpanan data yang aman dan terorganisir. Dengan adanya aplikasi ini, diharapkan proses pencatatan harian menjadi lebih efektif, teratur, dan dapat mendukung introspeksi diri pengguna dalam jangka panjang.

BAB II
PEMBAHASAN

2.1. NetBeans

NetBeans adalah sebuah Integrated Development Environment (IDE) yang berperan sebagai pusat komando bagi pengembang, menyediakan semua alat yang dibutuhkan mulai dari editor kode yang cerdas hingga debugger untuk menulis, mengompilasi, dan mengelola proyek pemrograman, terutama yang berbasis Java. IDE open source ini sangat membantu untuk meningkatkan produktivitas karena menyediakan antarmuka yang terintegrasi, memungkinkan programmer untuk fokus pada logika aplikasi tanpa harus beralih antara berbagai program yang berbeda.

2.2. Spring

Spring Framework merupakan solusi komprehensif untuk membangun aplikasi enterprise yang besar dan terstruktur dalam ekosistem Java. Tujuan utamanya adalah menyederhanakan pengembangan aplikasi yang kompleks melalui konsep inti seperti Inversion of Control (IoC) dan Dependency Injection (DI), di mana framework mengambil alih tanggung jawab pengelolaan objek dan dependensi, sehingga kode aplikasi menjadi lebih bersih, modular, dan mudah diuji

2.3. Hibernate

Hibernate dalam Sistem Operasi (OS) seperti Windows atau macOS merupakan sebuah fitur penting dalam manajemen daya yang memungkinkan komputer untuk dimatikan sepenuhnya tanpa kehilangan kemajuan atau sesi kerja yang sedang berjalan. Hibernate bekerja dengan cara mengambil snapshot lengkap dari semua yang ada di RAM termasuk semua program dan dokumen yang terbuka dan secara permanen menyimpan snapshot tersebut ke hard disk (atau SSD).

2.3.1. Fungsi Hibernate

Fungsi utama Hibernate adalah menyimpan seluruh sesi kerja yang sedang aktif ke dalam penyimpanan permanen sehingga pengguna dapat melanjutkan aktivitasnya persis dari kondisi terakhir. Seluruh aplikasi, dokumen, dan proses yang berjalan akan disimpan dalam bentuk snapshot, membuat proses pemulihan jauh lebih mudah dan konsisten. Selain itu, Hibernate juga berfungsi untuk menghemat konsumsi daya karena komputer dapat mati sepenuhnya tanpa tetap memberi daya ke RAM seperti pada mode Sleep.

Dengan begitu, fitur ini memberikan fleksibilitas bagi pengguna untuk berpindah aktivitas atau meninggalkan perangkat dalam waktu lama tanpa harus melakukan shutdown penuh. Hibernate membantu mempercepat proses melanjutkan pekerjaan dibandingkan melakukan booting ulang, sehingga menjadi fitur yang sangat berguna bagi pengguna yang membutuhkan efisiensi waktu dan kenyamanan.

2.3.2. Manfaat Hibernate

Manfaat paling signifikan dari Hibernate adalah efisiensi energi karena perangkat benar-benar tidak menggunakan daya saat mode ini aktif. Hal ini sangat membantu pengguna laptop yang sering kehabisan baterai namun ingin menyimpan pekerjaannya tanpa risiko kehilangan data. Selain itu, Hibernate sangat praktis bagi pengguna yang membuka banyak aplikasi sekaligus karena sesi kerja dapat dipulihkan dengan cepat tanpa perlu membuka ulang semuanya secara manual. Mode ini juga cocok digunakan untuk pekerjaan jangka panjang atau multitasking yang kompleks karena memungkinkan pengguna berpindah aktivitas tanpa kehilangan fokus atau progress. Tidak hanya itu, Hibernate juga mendukung mobilitas tinggi, sehingga pengguna tetap dapat membawa perangkatnya berpindah tempat tanpa melakukan proses shutdown dan restart yang memakan waktu.

2.3.3. Kerugian dan Masalah Hibernate

Salah satu kerugian Hibernate adalah proses masuk dan keluar dari mode ini cenderung lebih lama dibanding Sleep, terutama pada perangkat yang masih menggunakan HDD atau memiliki RAM besar. Selain itu, Hibernate membutuhkan ruang penyimpanan yang cukup besar untuk menampung file hibernasi (hiberfil.sys), yang ukurannya bisa mencapai sebagian besar kapasitas RAM. Dalam beberapa kasus, Hibernate dapat menimbulkan error atau crash karena tidak semua driver atau aplikasi mampu mempertahankan compatibility saat sistem dipulihkan dari snapshot. Potensi korupsi data juga bisa terjadi jika ada gangguan ketika snapshot sedang ditulis ke penyimpanan, sehingga dapat mengakibatkan kegagalan resume atau bahkan kerusakan file tertentu. Mode ini juga kurang optimal pada perangkat lama atau lambat karena proses penyimpanan data RAM ke disk sangat bergantung pada

kecepatan media penyimpanan.

2.4. ORM (Object Relational Mapping)

Object-Relational Mapping (ORM) adalah sebuah teknik dan alat penting dalam pengembangan perangkat lunak modern yang berfungsi sebagai penerjemah dan penghubung antara model pemrograman berorientasi objek (Object-Oriented Programming - OOP) yang digunakan oleh bahasa seperti Java, dengan model data relasional yang digunakan oleh database tradisional seperti MySQL atau PostgreSQL. ORM memungkinkan para pengembang untuk berinteraksi dengan data yang tersimpan dalam tabel database menggunakan objek dan metode kelas bahasa pemrograman yang mereka kenal, alih-alih harus menulis dan mengelola query SQL (Structured Query Language) secara manual. Misalnya, alih-alih menulis perintah INSERT INTO users..., developer cukup memanggil metode save() pada objek User.

BAB III
ANALISA DAN PERANCANGAN

Project pada praktikum Rekayasa Perangkat Lunak 2 ini bertema *Daily Journal*, yang bertujuan untuk membuatkan pengguna sebuah platform untuk mengetik jurnal harian, dengan fitur menunjukkan emosi setiap kali pengguna mengetik jurnal, sehingga pengguna dapat melihat keseharian jurnal sebelumnya seperti sedang sedih atau bahagia.

Project ini dibuat menggunakan Java with Maven sebagai platform utamanya dan Java Application atau Desktop Application yang akan digunakan, seperti gambar dibawah ini. Jika project sudah dipilih, langkah selanjutnya cukup mengklik tombol *next*.

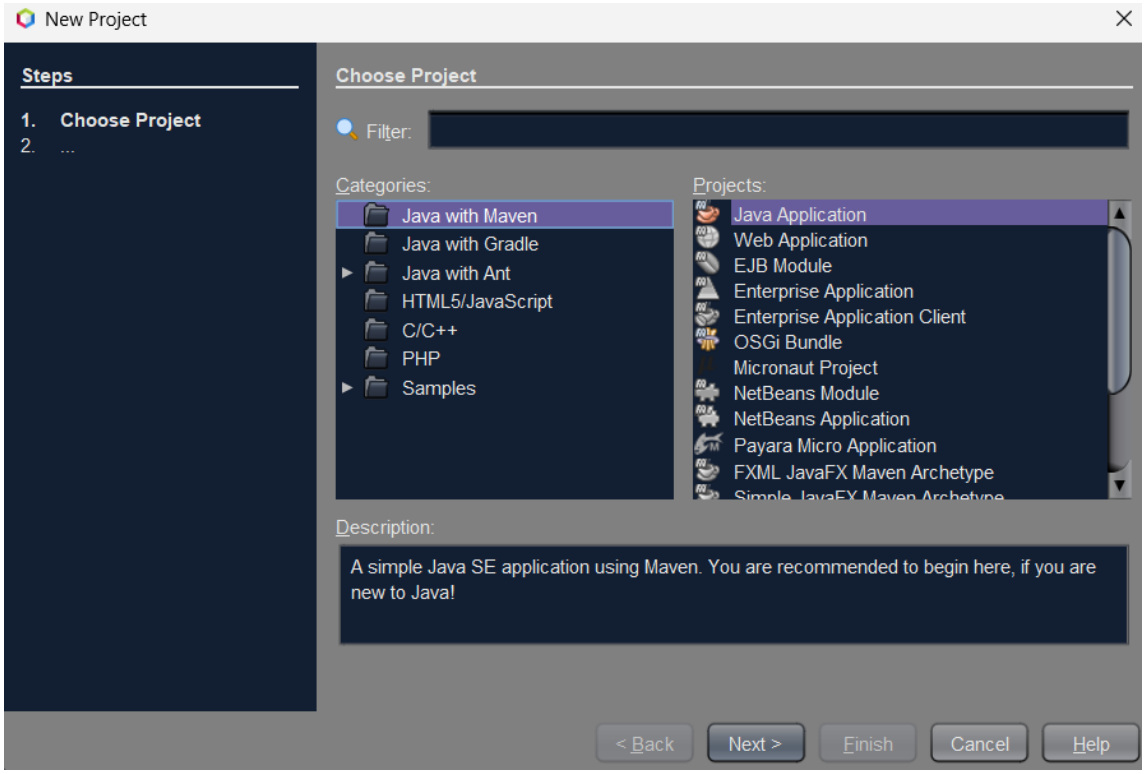


Figure 1 Pemilihan Konfigurasi Project

Setelah pemilihan categories dan projects selesai, maka akan ditampilkan menu untuk menamai project, lokasi project, group id, version dan package. Untuk group id, version dan package, Netbeans biasanya sudah memberikan konfigurasi secara default, sehingga pengembang hanya perlu konfigurasi nama dan lokasi dari project. Menu tersebut dalam dilihat pada gambar dibawah ini.

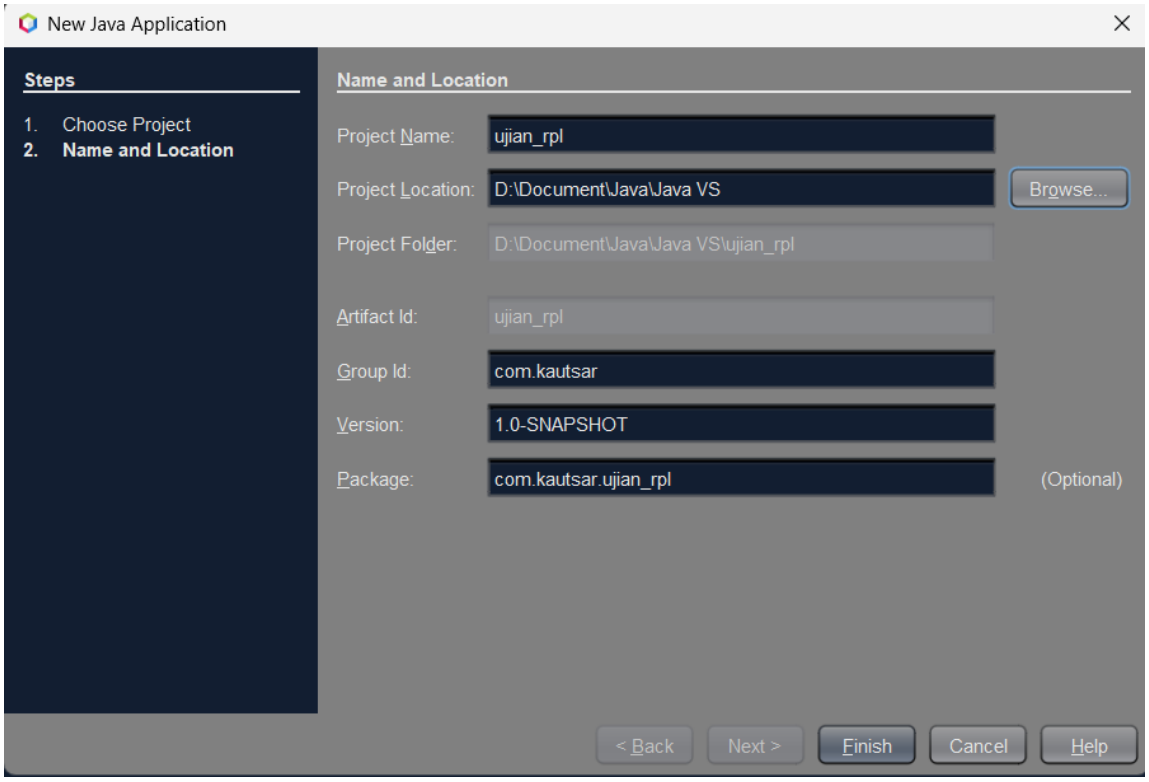


Figure 2 Konfigurasi Nama dan Lokasi Project

a. Model Journal

Dalam pengerjaan project, diperlukan model sebagai blueprint dari suatu data, berikut kode dari model yang dibuat.

```
package com.kautsar.ujian_rpl.model;
/**
 *
 * @author kauts
 */
import jakarta.persistence.*;
import java.time.LocalDateTime;

@Entity
@Table(name = "journal")
public class JournalModel {

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id")
        private int id;

        @Column(name = "judul")
        private String judul;

        @Column(name = "penulis")
        private String penulis;

        @Column(name = "emotion")
        private String emotion;

        @Column(name = "text")
```

```
        private String text;

        @Column(name = "created_at")
        private LocalDateTime createdAt;

        public JournalModel() {

        }

        public JournalModel(String judul, String penulis, String emotion,
           String text) {
            this.judul = judul;
            this.penulis = penulis;
            this.emotion = emotion;
            this.text = text;
            this.createdAt = LocalDateTime.now();

        }

        public int getId() {
            return id;
        }

        public String getJudul() {
            return judul;
        }

        public void setJudul(String judul) {
            this.judul = judul;
        }

        public String getPenulis() {
            return penulis;
        }

        public void setPenulis(String penulis) {
            this.penulis = penulis;
        }

        public String getEmotion() {
            return emotion;
        }

        public void setEmotion(String emotion) {
            this.emotion = emotion;
        }

        public String getText() {
            return text;
        }

        public void setText(String text) {
            this.text = text;
        }

         public LocalDateTime getCreatedAt() {
            return createdAt;
        }

        public void setCreatedAt(LocalDateTime createdAt) {
            this.createdAt = createdAt;
        }
}
```

Kode ini mendefinisikan class JournalModel yang berfungsi sebagai representasi dari

satu baris dalam tabel database bernama journal. Setiap properti (field) dalam class ini, seperti judul, penulis, emotion, text, dan createdAt, dipetakan (mapped) secara langsung ke kolom yang sesuai dalam tabel tersebut, yang dikendalikan oleh anotasi @Entity dan @Table. Secara ringkas, fungsi utama kode ini adalah bertindak sebagai Persistence Model atau POJO (Plain Old Java Object) yang digunakan oleh framework ORM untuk melakukan operasi data (seperti menyimpan, mengambil, atau memperbarui) ke dalam database, sehingga pengembang dapat bekerja dengan data sebagai objek Java alih-alih harus menulis kueri SQL secara manual.

b. Repository Journal

Berikut kode untuk membuat repository journal yang berfungsi untuk komunikasi data dasar antara aplikasi dan database.

```
package com.kautsar.ujian_rpl.repository;

/**
 *
 * @author kauts
*/
import com.kautsar.ujian_rpl.model.JournalModel;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface JournalRepository extends JpaRepository<JournalModel,
Integer>  {


}
```

Kode ini mendefinisikan interface JournalRepository yang berperan penting dalam aplikasi Spring Boot dan JPA. Dengan mengimplementasikan antarmuka JpaRepository<JournalModel, Integer>, kode ini secara otomatis menciptakan lapisan Repository yang menangani semua komunikasi data dasar (Create, Read, Update, Delete) antara logika aplikasi dan tabel journal di database. Fungsi utamanya adalah menyediakan serangkaian metode akses data yang sudah siap pakai tanpa perlu menulis kode implementasi SQL atau JDBC, seperti save(), findById(), dan findAll(). Dengan demikian, interface ini memisahkan logika bisnis (yang ada di lapisan Service) dari detail teknis persistensi data, menjadikannya pola kunci untuk pengembangan aplikasi enterprise yang bersih dan mudah dipelihara.

c. Service Journal

Service berguna untuk layanan atau fungsi dasar dalam CRUD yang dimana nanti akan digunakan oleh controller untuk terhubung ke view.

```
package com.kautsar.ujian_rpl.service;

import com.kautsar.ujian_rpl.model.JournalModel;
import com.kautsar.ujian_rpl.repository.JournalRepository;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
/**
*
* @author kauts
*/
@Service
public class JournalService {
            @Autowired
            private JournalRepository repository;

            public void addJournal(JournalModel journal) {
                repository.save(journal);
            }

            public JournalModel getJournal(int id) {
                return repository.findById(id).orElse(null);
            }

            public void updateJournal(JournalModel mhs) {
                repository.save(mhs);
            }

            public void deleteJournal(int id) {
                repository.deleteById(id);
            }

            public List<JournalModel> getAllJournal() {
                return repository.findAll();
            }

}
```

Kode diatas mendefinisikan class JournalService, yang merupakan komponen inti dalam arsitektur aplikasi Spring Boot dan berfungsi sebagai lapisan Service (bisnis logika). Ditandai dengan anotasi @Service, class ini mengintegrasikan (melalui @Autowired) JournalRepository yang sebelumnya Anda tunjukkan, sehingga Service dapat berinteraksi dengan database. Fungsi utama dari lapisan ini adalah menampung dan mengeksekusi aturan bisnis aplikasi. Dalam kode ini, Service menyediakan serangkaian metode operasi CRUD (Create, Read, Update, Delete) seperti addJournal(), getJournal(), updateJournal(), deleteJournal(), dan getAllJournal(). Metode-metode ini secara langsung memanggil metode persistence yang diwarisi dari JpaRepository (seperti save(), findById(), deleteById(), dan findAll()), bertindak sebagai perantara yang memastikan data dimanipulasi dengan benar sebelum atau sesudah berinteraksi dengan database, dan mempersiapkan data tersebut untuk dikonsumsi oleh lapisan Controller aplikasi.

d. Controller Journal

Controller berfungsi untuk mengontrol atau mengatur alur dari pembuatan jurnal, pembaruan,sampai dengan penghapusan jurnal.

```
package com.kautsar.ujian_rpl.controller;

import com.kautsar.ujian_rpl.model.JournalModel;
import com.kautsar.ujian_rpl.service.JournalService;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

/**
 *
 * @author kauts
 */
@Controller
public class JournalController {

        @Autowired
        private JournalService journalService;

            public void addJournal(JournalModel journal) {
                journalService.addJournal(journal);
            }

        public JournalModel getJournal(int id) {
            return journalService.getJournal(id);
        }

        public void updateJournal(JournalModel mhs) {
            journalService.updateJournal(mhs);
        }

        public void deleteJournal(int id) {
            journalService.deleteJournal(id);
        }

        public List<JournalModel> getAllJournal() {
            return journalService.getAllJournal();
        }
}
```

Fungsi utama dari kode JournalController adalah bertindak sebagai penerima dan pemroses permintaan (request) dari pengguna atau klien web. Ditandai dengan anotasi @Controller, class ini bertanggung jawab untuk menerima input (seperti data yang dikirim melalui formulir web atau API) dan mengelola output (data yang dikirim kembali ke pengguna). Controller ini mengintegrasikan (melalui @Autowired) JournalService yang sudah Anda buat sebelumnya. Setiap metode di dalam Controller seperti addJournal(), getJournal(), dan deleteJournal() berfungsi sebagai endpoint yang memetakan permintaan spesifik ke aksi tertentu, dan tugasnya hanyalah mendelegasikan pemrosesan logika bisnis yang sebenarnya ke lapisan Service. Dengan kata lain, Controller adalah gerbang depan aplikasi yang menangani interaksi I/O dan

meneruskan pekerjaan berat (heavy lifting) ke Service untuk memastikan pemisahan tanggung jawab yang jelas dalam aplikasi.

e. GUI / View Journal

1. GUI MainFrame

Mainframe digunakan untuk menampilkan halaman wrapper atau pembungkus agar halaman lain dapat berpindah pindah dengan menggunakan Jframe sebagai wadahnya.

```
package com.kautsar.ujian_rpl.view;
/**
*
* @author kauts
*/
import java.awt.CardLayout;
import javax.swing.*;
import org.springframework.stereotype.Component;

public class MainFrame extends JFrame {

        private CardLayout cardLayout;
        private JPanel mainPanel;
        private Runnable currentPageCallback;

        public MainFrame() {
            setTitle("Aplikasi Journal");
            setSize(800, 500);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            cardLayout = new CardLayout();
            mainPanel = new JPanel(cardLayout);
            add(mainPanel);
        }

        public void addPage(JPanel page, String name) {
            mainPanel.add(page, name);

        }

        public void showPage(String name) {
            cardLayout.show(mainPanel, name);

            if (currentPageCallback != null) {
                currentPageCallback.run();          }
        }
}
```

2. GUI Journal (Halaman utama untuk input jurnal)

JournalGUI digunakan untuk halaman utama pada aplikasi, berisi input untuk memasukkan journal dengan bermacam macam parameter, yaitu judul, nama penulis, emotion, dan isi dari journal.

```
package com.kautsar.ujian_rpl.view;
/**
 *
 * @author kauts
 */
import com.kautsar.ujian_rpl.controller.JournalController;
```

```
import com.kautsar.ujian_rpl.model.JournalModel;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.Image;
import javax.swing.*;

public class JournalGUI extends JPanel {

    private JTextField tfJudul;
    private JTextField tfPenulis;
    private JComboBox cbEmotion;
    private JTextArea taIsi;
    private JButton btnSave;
    private JButton btnLihatDaftar;
    private String s1[] = {"Happy", "Sad", "Confused", "Angry"};

    private String emotionSelected;
    private MainFrame mainFrame;

    public JournalGUI(JournalController controller, MainFrame mainFrame) {
        this.mainFrame = mainFrame;

        setLayout(new GridLayout(1, 2));

        // LEFT PANEL
        JPanel leftPanel = new JPanel();
        leftPanel.setLayout(null);
        add(leftPanel);

        JLabel lblJudul = new JLabel("Judul:");
        lblJudul.setBounds(20, 20, 100, 25);
        leftPanel.add(lblJudul);

        tfJudul = new JTextField();
        tfJudul.setBounds(120, 20, 200, 25);
        leftPanel.add(tfJudul);

        JLabel lblPenulis = new JLabel("Penulis:");
        lblPenulis.setBounds(20, 60, 100, 25);
        leftPanel.add(lblPenulis);

        tfPenulis = new JTextField();
        tfPenulis.setBounds(120, 60, 200, 25);
        leftPanel.add(tfPenulis);

        JLabel lblEmotion = new JLabel("Emotion:");
        lblEmotion.setBounds(20, 100, 100, 25);
        leftPanel.add(lblEmotion);

        cbEmotion = new JComboBox(s1);
        cbEmotion.setBounds(120, 100, 200, 25);
        leftPanel.add(cbEmotion);

        JLabel lblIsi = new JLabel("Isi:");
        lblIsi.setBounds(20, 140, 100, 25);
        leftPanel.add(lblIsi);

        taIsi = new JTextArea();
        JScrollPane scroll = new JScrollPane(taIsi);
        scroll.setBounds(120, 140, 200, 120);
        leftPanel.add(scroll);

        btnSave = new JButton("Simpan");
        btnSave.setBounds(120, 280, 100, 30);
        leftPanel.add(btnSave);
```

```
        // Tombol ke list
        btnLihatDaftar = new JButton("Lihat Daftar");
        btnLihatDaftar.setBounds(230, 280, 120, 30);
        leftPanel.add(btnLihatDaftar);

        btnSave.addActionListener(e -> {
            JournalModel journal = new JournalModel();
            journal.setJudul(tfJudul.getText());
            journal.setText(taIsi.getText());
            controller.addJournal(journal);

            JOptionPane.showMessageDialog(this, "Journal berhasil
disimpan!");
        });

        btnLihatDaftar.addActionListener(e -> {
            mainFrame.showPage("journalList");
        });

        // RIGHT PANEL
        JPanel rightPanel = new JPanel(new BorderLayout());
        add(rightPanel);

        JLabel lblKondisi = new JLabel("Saat ini kamu merasa:");
        rightPanel.add(lblKondisi, BorderLayout.NORTH);

        JLabel imageLabel = new JLabel();
        imageLabel.setHorizontalAlignment(JLabel.CENTER);
        rightPanel.add(imageLabel, BorderLayout.CENTER);

        updateImage(imageLabel);

        cbEmotion.addActionListener(e -> {
            emotionSelected = cbEmotion.getSelectedItem().toString();
            updateImage(imageLabel);
        });
    }

    private void updateImage(JLabel imageLabel) {
        String path = "/assets/happy.png";

        if ("Sad".equals(emotionSelected)) {
            path = "/assets/sad.png";
        } else if ("Angry".equals(emotionSelected)) {
            path = "/assets/angry.png";
        } else if ("Confused".equals(emotionSelected)) {
            path = "/assets/confused.png";
        }

        ImageIcon icon = new ImageIcon(getClass().getResource(path));
        Image scaled = icon.getImage().getScaledInstance(200, 200,
Image.SCALE_SMOOTH);
        imageLabel.setIcon(new ImageIcon(scaled));
    }
}
```

3. GUI Journal List

Kode GUI Journal list ini digunakan untuk menampilkan journal yang ada di database maupun jurnal yang baru di input melalui halaman utama, halaman ini merupakana halaman kedua. Untuk mengakses halaman ini, perlu klik tombol “Lihat Daftar” dan akan berpindah tempat. Setiap journal yang ditampilkan

sebagai card bar memiliki judul, preview isi, tanggal dan emotion serta memiliki button detail, edit dan hapus.

```
package com.kautsar.ujian_rpl.view;
/**
 *
 * @author kauts
 */
import com.kautsar.ujian_rpl.controller.JournalController;
import com.kautsar.ujian_rpl.model.JournalModel;
import java.util.List;
import java.awt.*;
import javax.swing.*;


public class JournalListGUI extends JPanel {

    private MainFrame mainFrame;
    private JournalController controller;
    private JPanel listPanel;

    public JournalListGUI(MainFrame mainFrame, JournalController
controller) {
        this.mainFrame = mainFrame;
        this.controller = controller;

        setLayout(new BorderLayout());

        JLabel title = new JLabel("Daftar Jurnal", SwingConstants.CENTER);
        add(title, BorderLayout.NORTH);

        listPanel = new JPanel();
        listPanel.setLayout(new BoxLayout(listPanel, BoxLayout.Y_AXIS));
        add(new JScrollPane(listPanel), BorderLayout.CENTER);

        JButton btnBack = new JButton("Kembali");
        btnBack.addActionListener(e ->
mainFrame.showPage("journalInput"));
        add(btnBack, BorderLayout.SOUTH);

        loadData();
    }

    public void loadData() {
        listPanel.removeAll();

        List<JournalModel> journals = controller.getAllJournal();

        for (JournalModel j : journals) {
            listPanel.add(makeItem(j));
        }

        listPanel.revalidate();
        listPanel.repaint();
    }

    private JPanel makeItem(JournalModel j) {
        JPanel panel = new JPanel(new BorderLayout());
        panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        JPanel left = new JPanel();
        left.setLayout(new BoxLayout(left, BoxLayout.Y_AXIS));
        left.add(new JLabel("Judul: " + j.getJudul()));
        left.add(new JLabel("Preview: " + j.getText().substring(0,
```

```
Math.min(20, j.getText().length())) + "..."));
        panel.add(left, BorderLayout.CENTER);

        JPanel right = new JPanel();
        right.setLayout(new BoxLayout(right, BoxLayout.Y_AXIS));

        JButton btnDetail = new JButton("Detail");
        JButton btnEdit = new JButton("Edit");
        JButton btnHapus = new JButton("Hapus");



        btnHapus.addActionListener(e -> {
            int confirm = JOptionPane.showConfirmDialog(this, "Hapus
jurnal ini?");
            if (confirm == JOptionPane.YES_OPTION) {
                controller.deleteJournal(j.getId());
                loadData();
            }
        });

        right.add(btnDetail);
        right.add(btnEdit);
        right.add(btnHapus);

        panel.add(right, BorderLayout.EAST);

        return panel;
    }

     public void refreshList() {
        listPanel.removeAll();

        List<JournalModel> journals = controller.getAllJournal();

        for (JournalModel j : journals) {
            JPanel item = new JPanel();
            item.setLayout(new BorderLayout());

            JLabel lbl = new JLabel(j.getJudul());
            item.add(lbl, BorderLayout.CENTER);

            JButton btnDetail = new JButton("Detail");
            item.add(btnDetail, BorderLayout.EAST);

            listPanel.add(item);
        }

        listPanel.revalidate();
        listPanel.repaint();
    }
}
```

4. GUI Journal Detail

Kode Journal Detail berfungsi untuk menampilkan detail dari journal menggunakan Jpanel sebagai wadah dari windowsnya.

```
package com.kautsar.ujian_rpl.view;

/**
 *
 * @author kauts
 */
```

```
import com.kautsar.ujian_rpl.model.JournalModel;
import java.awt.BorderLayout;
import javax.swing.*;

public class JournalDetailGUI extends JPanel {

    private MainFrame mainFrame;
    private JLabel lblJudul;
    private JLabel lblPenulis;
    private JTextArea taIsi;

    public JournalDetailGUI(MainFrame mainFrame) {
        this.mainFrame = mainFrame;

        setLayout(new BorderLayout());

        lblJudul = new JLabel("Judul: ");
        lblPenulis = new JLabel("Penulis: ");

        taIsi = new JTextArea();
        taIsi.setEditable(false);

        add(lblJudul, BorderLayout.NORTH);
        add(new JScrollPane(taIsi), BorderLayout.CENTER);

        JButton btnBack = new JButton("Kembali");
        btnBack.addActionListener(e -> mainFrame.showPage("journalList"));
        add(btnBack, BorderLayout.SOUTH);
    }


    public void setDetail(JournalModel j) {
        lblJudul.setText("Judul: " + j.getJudul());
        lblPenulis.setText("Penulis: " + j.getPenulis());
        taIsi.setText(j.getText());
    }
}
```

5. GUI Journal Edit

Kode Journal Edit berfungsi untuk menampilkan halaman edit journal menggunakan Jpanel sebagai wadah dari windowsnya.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-
default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to
edit this template
 */
package com.kautsar.ujian_rpl.view;


/**
 *
 * @author kauts
 */
import javax.swing.*;
import java.awt.*;
```

```
import com.kautsar.ujian_rpl.controller.JournalController;
import com.kautsar.ujian_rpl.model.JournalModel;
import java.time.format.DateTimeFormatter;

public class JournalEditGUI extends JPanel {

    private JTextField judulField;
    private JTextField penulisField;
    private JComboBox cbEmotion;
    private JTextArea textArea;

    private JLabel createdAtLabel;

    private JButton updateButton;
    private JButton cancelButton;

    private JournalModel journal;

    private MainFrame mainFrame;
    private JournalController controller;
    private String s1[] = {"Happy", "Sad", "Confused", "Angry"};

    public JournalEditGUI(MainFrame mainFrame, JournalController
controller) {

        this.mainFrame = mainFrame;
        this.controller = controller;

        setLayout(new BorderLayout());
        setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        JLabel title = new JLabel("Edit Journal", SwingConstants.CENTER);
        title.setFont(new Font("Arial", Font.BOLD, 20));
        add(title, BorderLayout.NORTH);

        JPanel form = new JPanel(new GridLayout(10, 1, 5, 5));

        form.add(new JLabel("Judul:"));
        judulField = new JTextField();
        form.add(judulField);

        form.add(new JLabel("Penulis:"));
```

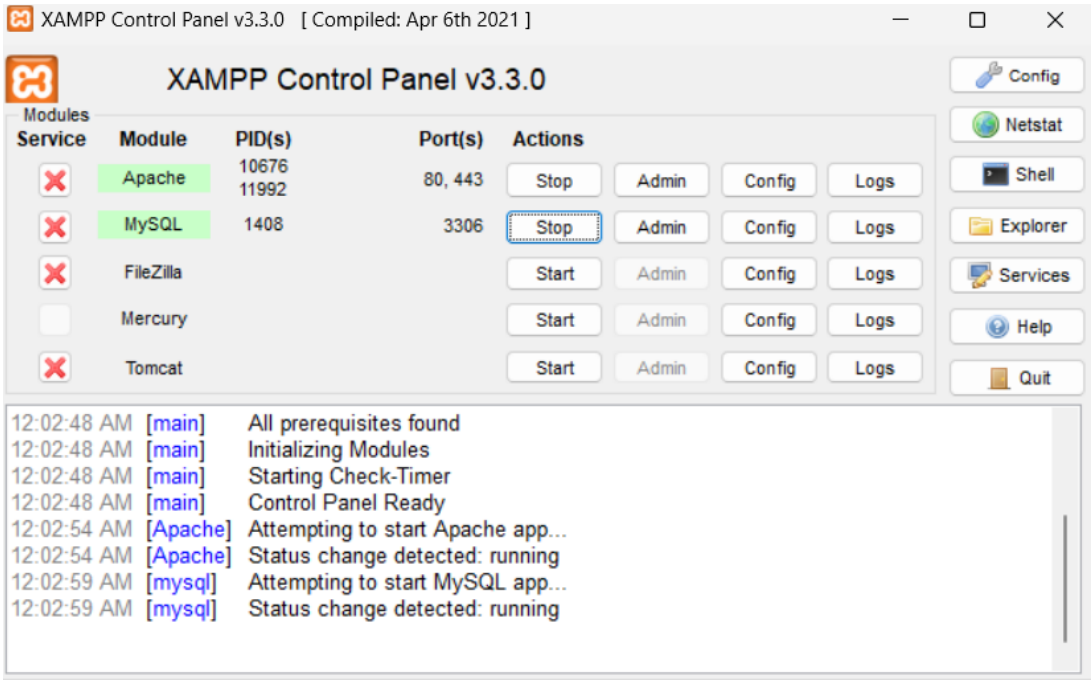
```
        penulisField = new JTextField();
        form.add(penulisField);

        form.add(new JLabel("Emotion"));
        cbEmotion = new JComboBox(s1);
        form.add(cbEmotion);

        form.add(new JLabel("Isi Journal:"));
        textArea = new JTextArea(5, 20);
        form.add(new JScrollPane(textArea));

        form.add(new JLabel("Tanggal Dibuat:"));
        createdAtLabel = new JLabel("-");
        form.add(createdAtLabel);

        add(form, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        updateButton = new JButton("Update");
        cancelButton = new JButton("Cancel");
        buttonPanel.add(updateButton);
        buttonPanel.add(cancelButton);

        add(buttonPanel, BorderLayout.SOUTH);

        updateButton.addActionListener(e -> {
            if (journal == null) {
                JOptionPane.showMessageDialog(this, "Tidak ada data yang
dipilih!");
                return;
            }

            journal.setJudul(getInputJudul());
            journal.setPenulis(getInputPenulis());
            journal.setEmotion(getInputEmotion());
            journal.setText(getInputText());

            controller.updateJournal(journal);

            JOptionPane.showMessageDialog(this, "Journal berhasil
diupdate!");

            // refresh list
```

```
            JournalListGUI listPage = (JournalListGUI)
mainFrame.getPage("journalList");
            listPage.loadData();

            // pindah ke halaman list
            mainFrame.showPage("journalList");
        });

        cancelButton.addActionListener(e -> {
            mainFrame.showPage("journalList");
        });
    }

    public void setJournal(JournalModel j) {
        this.journal = j;

        if (j != null) {
            judulField.setText(j.getJudul());
            penulisField.setText(j.getPenulis());
            cbEmotion.setSelectedItem(j.getEmotion());
            textArea.setText(j.getText());
            createdAtLabel.setText(j.getCreatedAt().format(
                    DateTimeFormatter.ofPattern(
                            "d MMMM yyyy HH:mm",
                            new java.util.Locale("id")
                    )
            ));
        }
    }

    public String getInputJudul() {
        return judulField.getText();
    }

    public String getInputPenulis() {
        return penulisField.getText();
    }

    public String getInputEmotion() {
        return cbEmotion.getSelectedItem().toString();
    }

    public String getInputText() {
```

```
            return textArea.getText();
        }
}
```

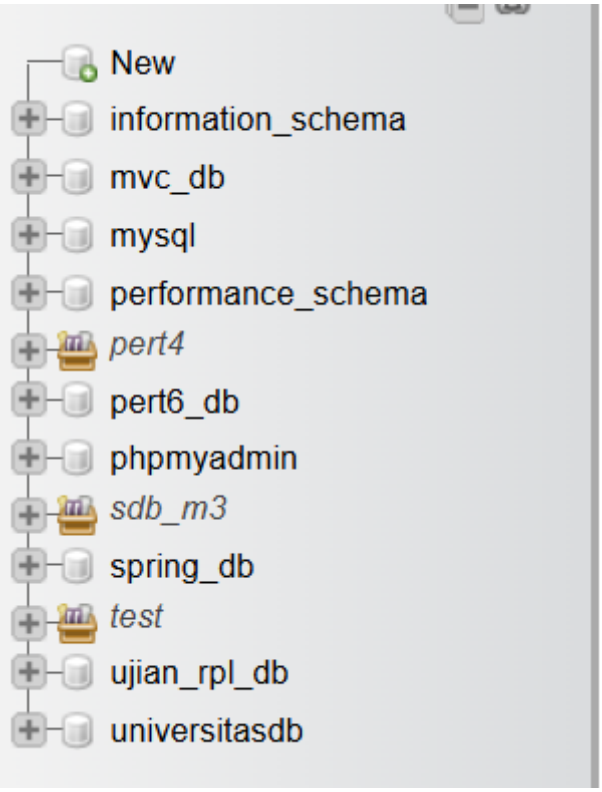
f. Menyalakan Server Database

Project ini menggunakan MySQL sebagai database, yang dimana database tersebut terhubung dengan XAMPP untuk control panel dari database. XAMPP adalah sebuah stack perangkat lunak open-source dan cross-platform yang sangat populer, dirancang khusus untuk menciptakan lingkungan server web lokal yang lengkap dan siap pakai di komputer pribadi. Untuk menyalakan server dan database, klik tombol *action* pada Apache dan MySQL seperti gambar dibawah ini, dan setelah berwarna hijau , klik tombol *admin* untuk membuka *phpmyadmin* dan konfigurasi database.



g. Pembuatan Database

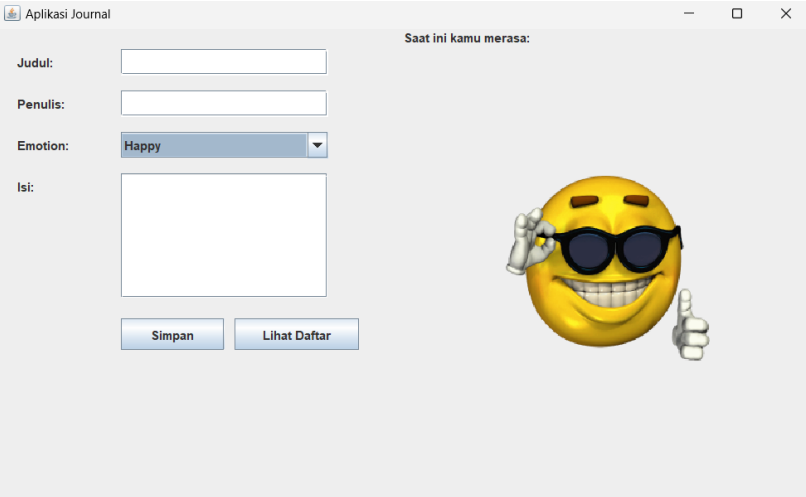
Database dibuat dengan nama ujian_rpl_db , database ini cukup dibuat saja, dikarenakan table dalam database akan dibuat otomatis melalui jakarta persistance yaitu library API dari Java.

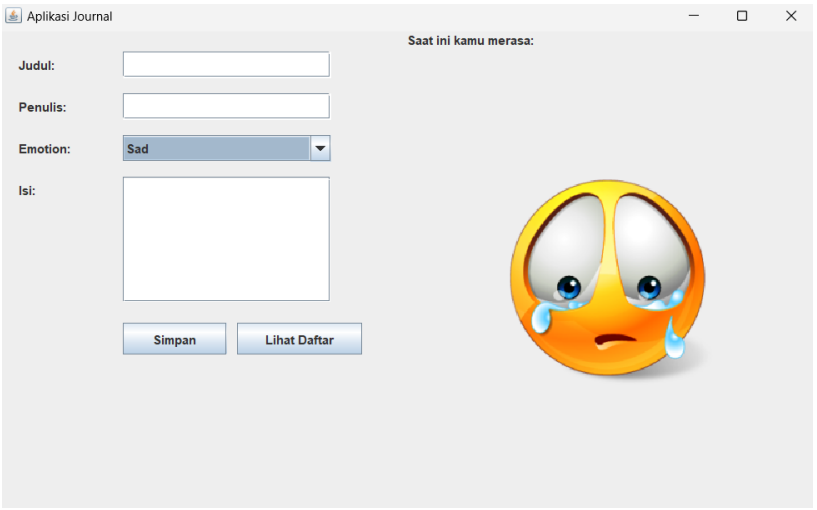


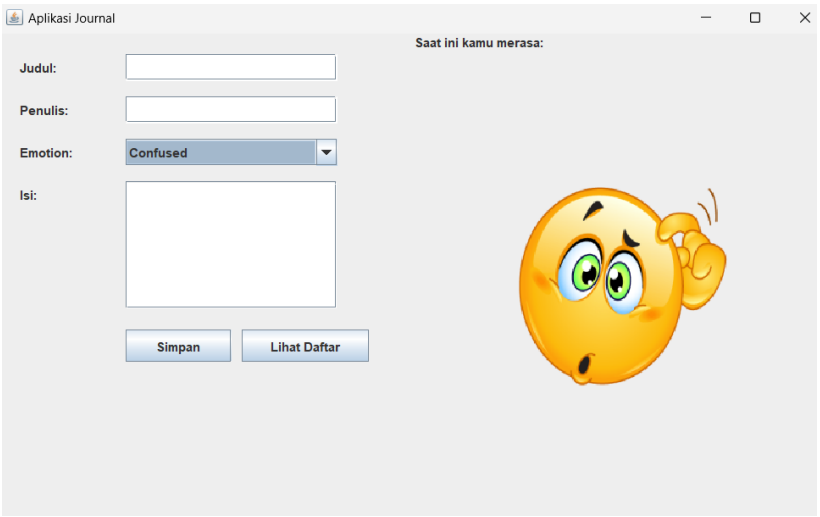
h. Output

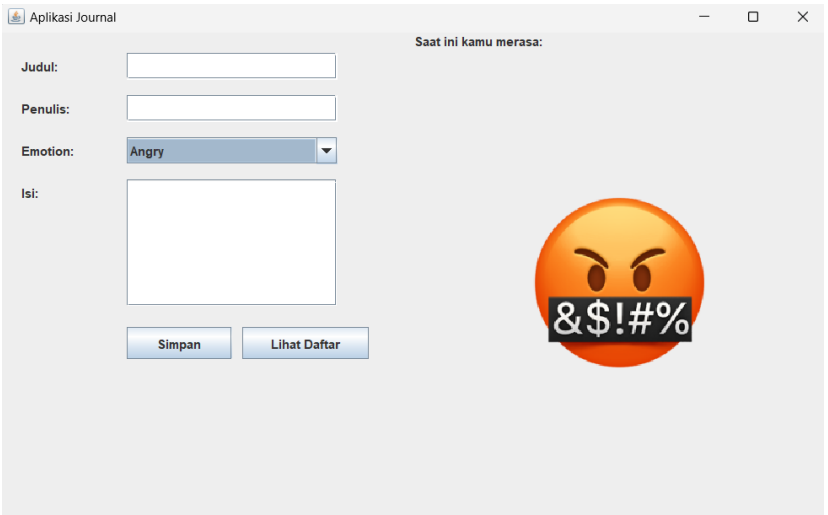
Berikut output halaman dari aplikasi desktop *daily journal*

1. Halaman input journal (Halaman Utama)

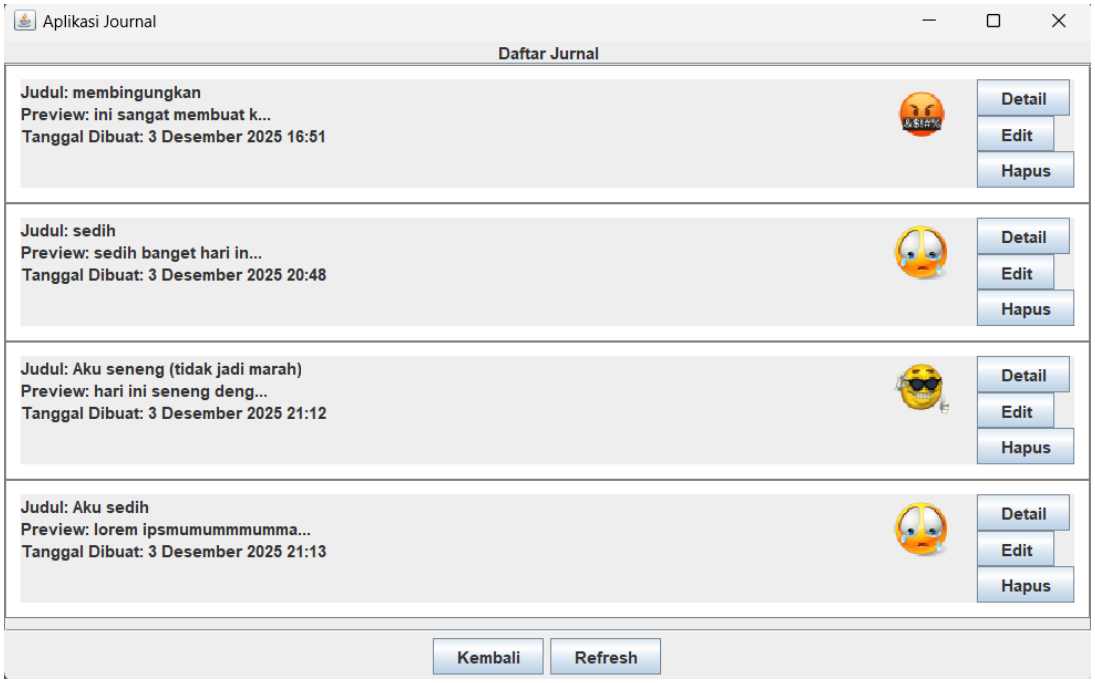



Aplikasi Journal
Saat ini kamu merasa:
Judul:
Penulis:
Emotion:
Sad
Isi:
Simpan
Lihat Daftar

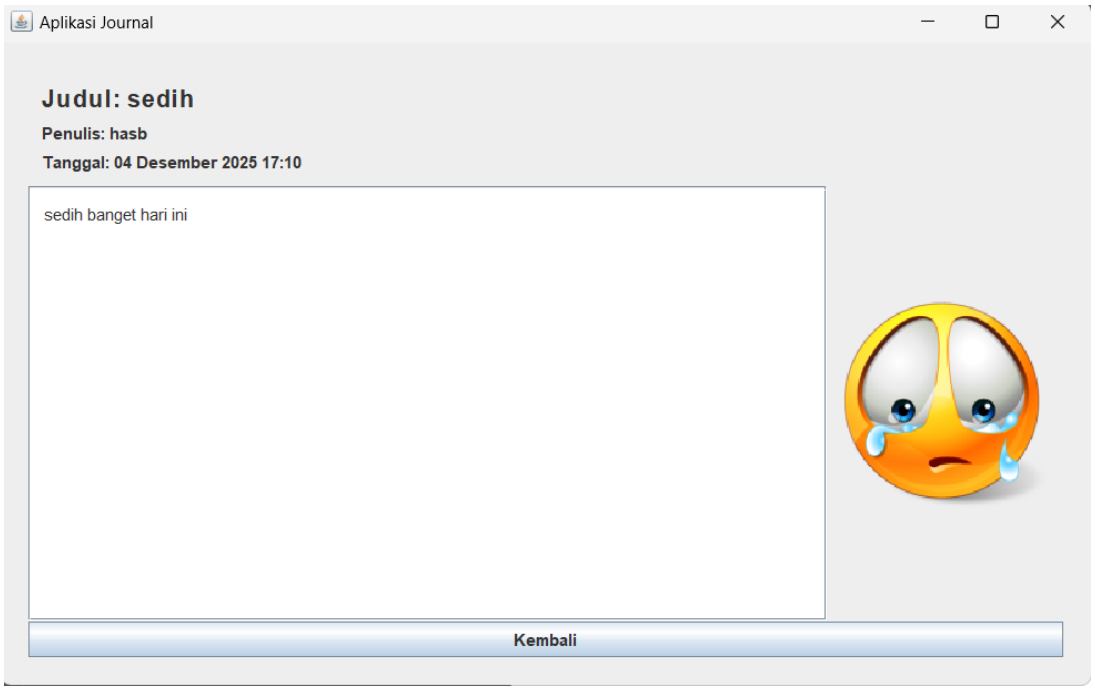

Aplikasi Journal
Saat ini kamu merasa:
Judul:
Penulis:
Emotion:
Confused
Isi:
Simpan
Lihat Daftar


Aplikasi Journal
Saat ini kamu merasa:
Judul:
Penulis:
Emotion:
Angry
Isi:
Simpan
Lihat Daftar
&$!#%

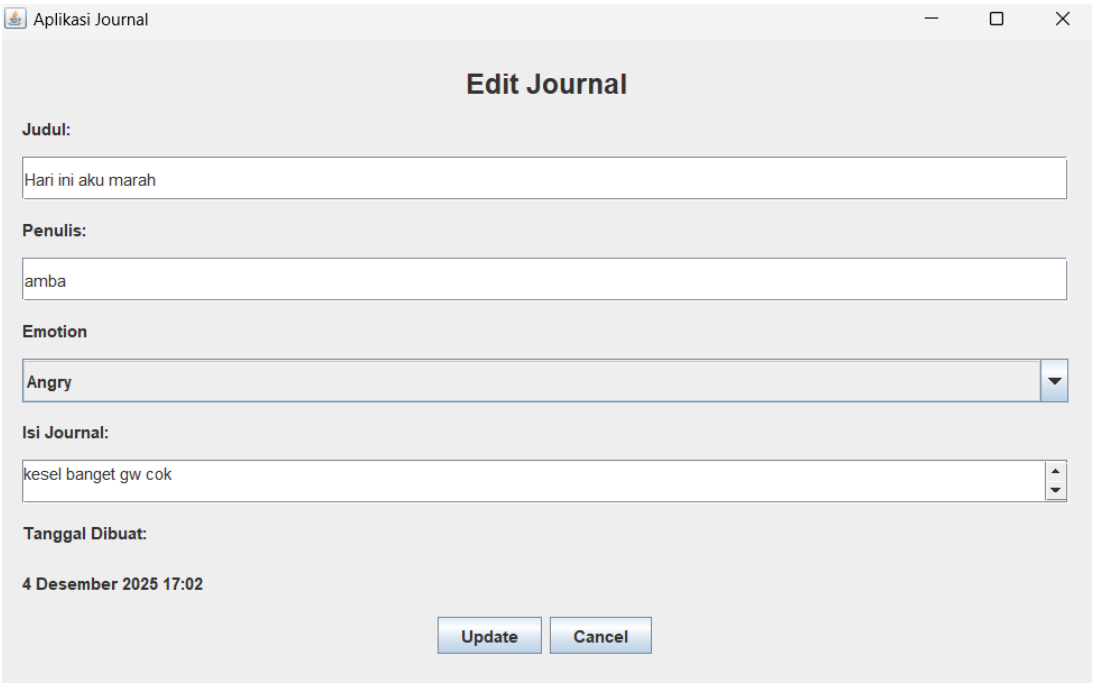
2. Halaman list journal



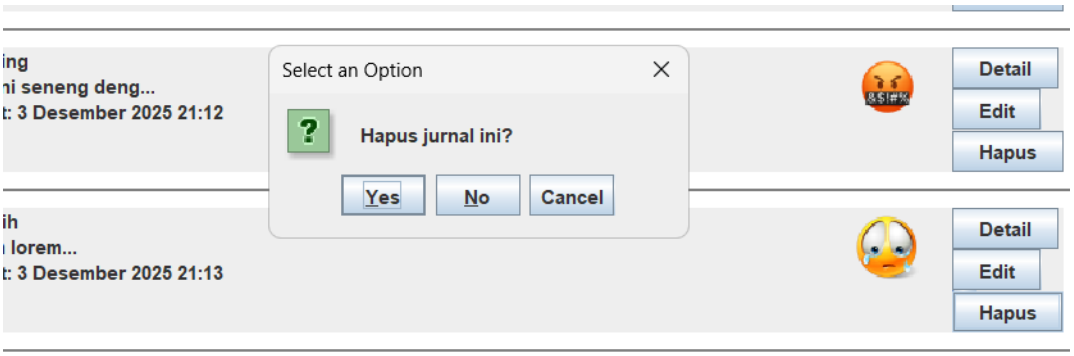
3. Halaman Detail journal



4. Halaman Edit Journal



5. Dialog Hapus



BAB IV
PENUTUP

4.1 Kesimpulan

Proyek *Daily Journal* adalah implementasi aplikasi Java Desktop yang kuat dan modern, dibangun menggunakan kombinasi teknologi enterprise dan desktop untuk mengelola data jurnal. Inti dari aplikasi ini adalah memanfaatkan Spring Boot sebagai framework utama untuk mengorganisir kode dan mengelola ketergantungan (Dependency Injection), sebuah praktik yang biasanya terlihat pada aplikasi enterprise, namun diadaptasi untuk lingkungan desktop.

Aplikasi jurnal desktop ini menunjukkan implementasi arsitektur berlapis (layered architecture) yang solid, di mana Java Swing berfungsi sebagai lapisan antarmuka pengguna grafis (View), bertanggung jawab atas semua interaksi visual dengan pengguna, mulai dari formulir input hingga tampilan data. Di balik layar, proyek ini mengandalkan Spring Boot sebagai wadah framework utama yang mengatur semua komponen dan mengelola dependency injection, memastikan kode modularitas tinggi. Untuk persistensi data, proyek ini terhubung ke database MySQL lokal yang dijalankan melalui XAMPP, dengan MySQL Driver bertindak sebagai jembatan teknis. Lapisan data ini dimediasi oleh JournalRepository yang memanfaatkan Spring Data JPA dan Hibernate untuk menghilangkan kebutuhan menulis kode SQL manual, memungkinkan developer untuk fokus pada objek Java. Selanjutnya, JournalService menampung semua logika dan aturan bisnis aplikasi, memisahkan pemrosesan data dari antarmuka pengguna. Terakhir, JournalController berfungsi sebagai perantara penting, yang menerima aksi dari GUI Swing (seperti klik tombol) dan mendelegasikannya ke lapisan Service, memastikan pemisahan tanggung jawab yang rapi antara tampilan (Swing) dan logika aplikasi (Service).

4.2 Saran

Project masih belum memiliki validasi input yang memadai pada lapisan Controller maupun Service. Hal ini dapat menimbulkan risiko seperti data tidak konsisten, kesalahan proses bisnis, atau bahkan error saat penyimpanan ke database. Oleh karena itu, perlu dilakukan penambahan mekanisme validasi yang lebih ketat, seperti pengecekan nilai kosong, format tanggal yang benar, panjang teks, serta batasan-batasan khusus sesuai kebutuhan domain. Dengan adanya validasi pada lapisan logika aplikasi, sistem akan menjadi lebih robust dan mampu menangani input yang tidak valid sebelum diproses lebih lanjut atau disimpan ke database.