



UNTAR
Universitas Tarumanagara

**LAPORAN PROYEK AKHIR
OBJECT BASED PROGRAMMING**

**PEMBUATAN SISTEM PENDAFTARAN
DAN PENJADWALAN KURSUS
BAHASA INGGRIS**

DISUSUN OLEH KELOMPOK 5

**Michelle (825230048)
Melfanny Uredja (825230037)
Michael Emmanuel (825230057)
Julius Kristover (825230056)**

**PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS TARUMANAGARA
DESEMBER 2024**

DAFTAR ISI

Kata Pengantar.....	2
BAB I.....	3
PENDAHULUAN.....	3
1.1. Latar Belakang.....	3
1.2. Batasan Sistem.....	4
1.3. Tujuan dan Kegunaan.....	4
BAB II.....	6
RANCANGAN SISTEM.....	6
2.1. Sistem yang Dirancang.....	6
2.2. Class Diagram.....	6
2.3. Use Case Diagram.....	10
2.4. Activity Diagram.....	11
2.5. Sequence Diagram.....	16
BAB III.....	20
PEMBUATAN SISTEM.....	20
3.1. Implementasi Sistem yang Dirancang.....	20
3.2. Implementasi Abstract Class User.....	23
3.3. Implementasi Class.....	24
3.4. Implementasi Interface.....	42
BAB IV.....	46
PENGUJIAN PROGRAM APLIKASI.....	46
4.1. Skenario Pengujian.....	46
4.2. Pengujian Skenario SignUp.....	46
4.3. Pengujian Skenario Login.....	49
4.4. Pengujian Skenario Tes.....	51
4.5. Pengujian Skenario Reset Password.....	54
4.6. Pengujian Skenario Login Admin.....	56
4.7. Pengujian Skenario Melihat Info User.....	57
4.8. Pengujian Skenario Kelola Instructor.....	58
4.9. Pengujian Skenario Menambah Instructor.....	59
4.10. Pengujian Skenario Tambah Jadwal.....	60
4.11. Pengujian Skenario Menghapus Jadwal.....	61
4.12. Pengujian Skenario Kelola Enrollment.....	63
BAB V.....	64
PENGUJIAN PROGRAM APLIKASI.....	64
DAFTAR PUSTAKA.....	66

Kata Pengantar

Puji dan syukur kami panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan karunia-Nya, penulisan laporan ini yang berjudul “Sistem Pendaftaran dan Penjadwalan Kursus Bahasa Inggris” dapat diselesaikan dengan baik. Laporan ini disusun untuk memberikan gambaran mengenai pengembangan sebuah sistem yang bertujuan untuk mempermudah proses pendaftaran dan penjadwalan bagi peserta kursus bahasa inggris secara efisien dan efektif.

Dalam era digital yang terus berkembang, kebutuhan akan sistem informasi yang terintegrasi semakin meningkat, khususnya dalam bidang pendidikan. Sistem yang dikembangkan ini diharapkan mampu memberikan solusi praktis bagi lembaga kursus bahasa inggris dalam mengelola data pendaftaran peserta, menyusun jadwal kursus secara terstruktur, serta memberikan kemudahan akses bagi peserta kursus dalam memperoleh informasi yang mereka butuhkan.

Penulisan laporan ini tidak terlepas dari bantuan dan dukungan dari berbagai pihak. Oleh karena itu, kami ucapkan terima kasih kepada semua pihak yang telah turut memberikan kontribusi selama proses pengembangan dan penulisan laporan ini.

Tentunya, kami menyadari bahwa masih terdapat kekurangan, baik dari penyusunan maupun tata bahasa penyampaian dalam laporan ini. Maka dari itu, kami dengan rendah hati, kami menerima segala saran dan kritik dari pembaca demi penyempurnaan di masa mendatang. Kami juga berharap semoga laporan yang kami susun ini dapat memberikan manfaat dan juga inspirasi untuk pembaca. Terima kasih.

Jakarta, 12 November 2024

Kelompok 5

BAB I

PENDAHULUAN

1.1. Latar Belakang

Di era globalisasi saat ini, kemampuan berbahasa Inggris menjadi keterampilan yang sangat dibutuhkan. Semakin hari, permintaan untuk kursus bahasa Inggris terus meningkat di masyarakat. Namun, banyak lembaga kursus bahasa Inggris yang masih menggunakan metode manual dalam pendaftaran dan penjadwalan kelas, seperti pencatatan melalui formulir fisik dan penggunaan jadwal manual yang rentan terhadap kesalahan.

Proses pendaftaran dan penjadwalan yang dilakukan secara manual ini memiliki beberapa kelemahan. Pertama, proses pendaftaran yang lambat dapat menghambat calon peserta dalam proses pembelajaran dan data yang tersimpan pun sering kali sulit dikelola secara efektif. Kedua, penjadwalan jadwal manual memiliki resiko bentrok yang dapat mengganggu pembelajaran peserta.

Dengan memanfaatkan teknologi informasi, sebuah sistem pendaftaran dan penjadwalan kursus bahasa Inggris dapat dikembangkan untuk mengatasi permasalahan ini. Sistem ini akan memungkinkan calon peserta untuk mendaftar secara online, memilih jadwal yang tersedia, serta memudahkan pengelola dalam mengatur dan memantau pendaftaran dan jadwal secara *real-time*. Selain itu, sistem ini akan menyediakan fitur yang mampu menyesuaikan jadwal dengan ketersediaan pengajar dan fasilitas sehingga dapat memaksimalkan efektivitas proses belajar mengajar.

1.2. Batasan Sistem

Batasan sistem adalah kondisi yang membatasi fitur atau fungsi dari suatu sistem. Batasan ini mencangkup keterbatasan yang ada dalam penggunaan sistem yang dapat mempengaruhi efektivitas dan efisiensi untuk memenuhi tujuan yang diinginkan.

Kelompok kami menentukan batasan sistem yang belum dapat kami buat. Alasan kelompok kami belum dapat membuatnya dikarenakan keterbatasan waktu pengumpulan.

- **Tidak ada fitur yang mendukung multi-bahasa**

Sistem yang kami buat beserta tampilan di websitenya untuk saat ini hanya tersedia dalam bahasa inggris dan semua informasi harus diakses dalam bahasa inggris.

- **Tidak ada fitur chat**

Sistem yang kami buat tidak ada fitur *chat* yang dapat membantu pengguna untuk berkomunikasi secara online dengan admin.

- **Tidak ada fitur notifikasi**

Sistem yang kami buat tidak ada fitur notifikasi yang dapat membantu pengguna untuk mengingatkan pengguna mengenai jadwal kursus yang diambil.

1.3. Tujuan dan Kegunaan

Pembuatan sistem pendaftaran dan penjadwalan lembaga kursus kami buat untuk membantu meningkatkan efisiensi pendaftaran dan penjadwalan. Jadi, member kami tidak perlu datang ke tempat kursus hanya untuk mendaftar dan mendapatkan jadwal.

Kelompok kami membuat sistem tersebut untuk tujuan tertentu. Berikut adalah tujuan dari sistem pendaftaran dan penjadwalan kursus bahasa inggris yang kami buat.

- **Sistem kami menyediakan pendaftaran online**

Sistem yang kami buat dapat mempermudah proses pendaftaran member secara online. Dengan adanya fitur ini, member dapat mendaftar darimana saja dan kapan saja tanpa harus datang ke tempat kursus kami secara langsung.

- **Sistem kami dapat menyediakan pilihan jadwal**

Sistem yang kami buat menyediakan pilihan jadwal kepada member sehingga member dapat mengikuti kursus kami di waktu luang.

Berikut adalah kegunaan dari sistem pendaftaran dan penjadwalan kursus bahasa inggris yang kami buat :

- **Sistem kami dapat meningkatkan efisiensi operasional**

Sistem yang kami buat dapat meningkatkan efisiensi operasional karena proses pendaftaran dan penjadwalan yang sudah otomatis, waktu dan tenaga yang dibutuhkan untuk mengelola kursus dapat dihemat secara signifikan. Jadi memungkinkan lembaga kursus kami untuk fokus pada peningkatan kualitas pengajaran.

- **Sistem kami dapat memudahkan akses bagi member dan instruktur**

Sistem yang kami buat dapat memudahkan akses bagi member untuk mendaftar dan melihat jadwal kursus mereka dan instruktur dapat melihat informasi mengenai jadwal mengajar mereka

- **Sistem kami dapat mendukung pertumbuhan lembaga kursus**

Sistem yang kami buat dapat mendukung pertumbuhan lembaga kursus agar lebih handal dan mudah digunakan, jadi lembaga kursus dapat melayani lebih banyak member dan memperluas cakupan layanan mereka. Hal ini dapat meningkatkan jumlah member dan mendukung pertumbuhan lembaga kursus secara keseluruhan.

BAB II

RANCANGAN SISTEM

2.1. Sistem yang Dirancang

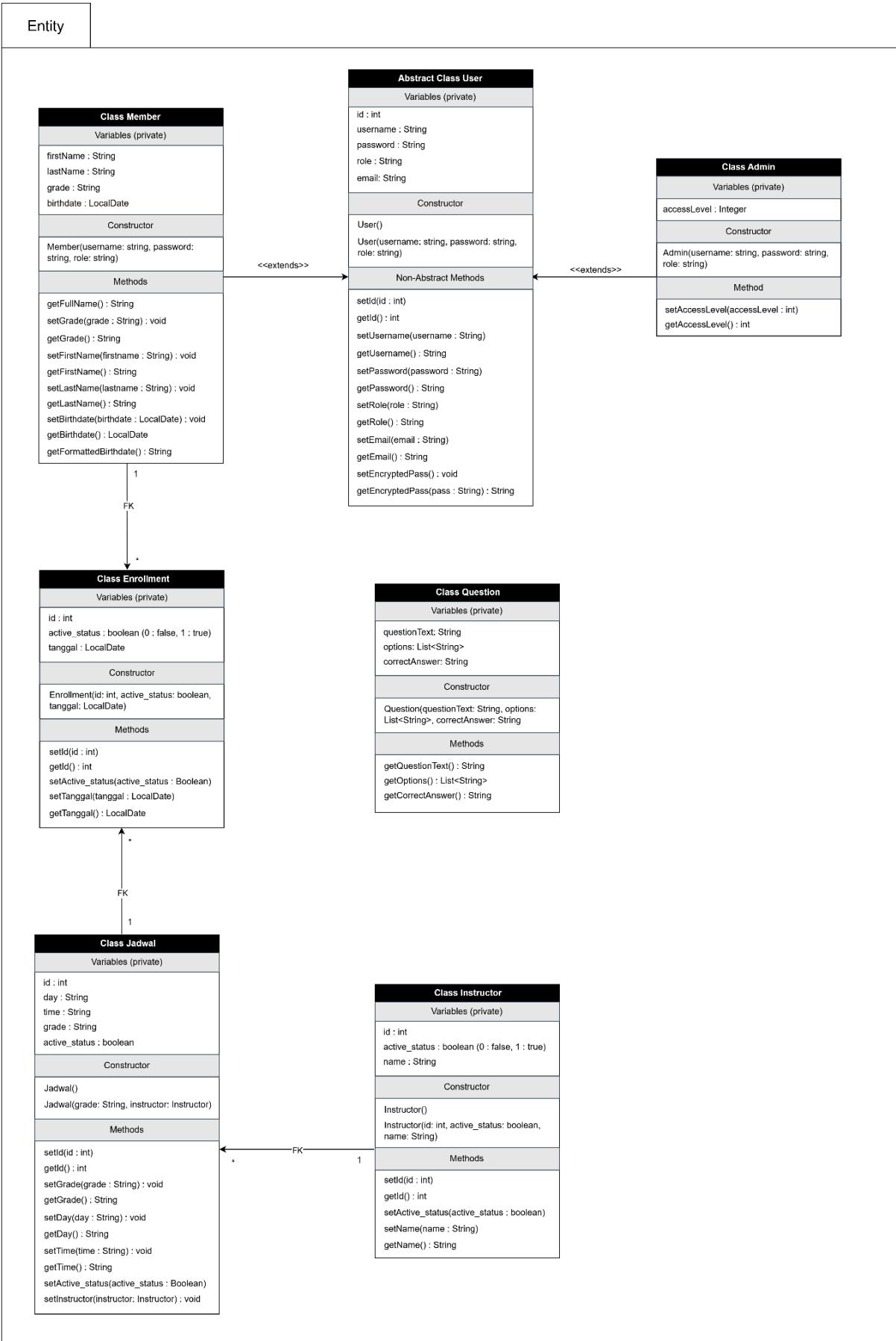
Pada bagian ini, kelompok kami akan menjelaskan secara garis besar mengenai UML diagram yang kami gunakan untuk menggambarkan rancangan sistem pendaftaran dan penjadwalan kursus kami. UML (*Unified Modelling Language*) adalah suatu metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan sistem berorientasi objek.

Kelompok kami membuat empat diagram, yaitu : *class diagram*, *use case diagram*, *activity diagram*, dan *sequence diagram*. Untuk penjelasan lebih lanjut, akan kami jelaskan di subbab selanjutnya.

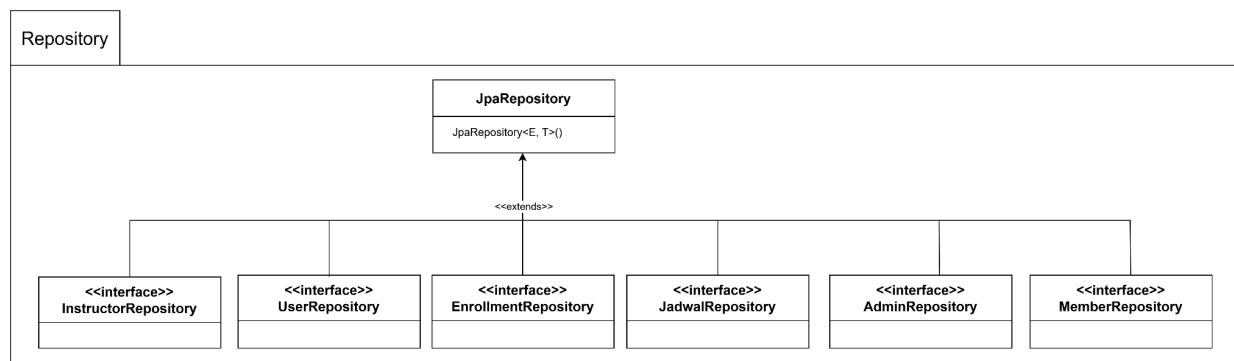
2.2. Class Diagram

Pada bagian ini, kelompok kami membuat class diagram untuk sistem pendaftaran dan penjadwalan lembaga kursus yang kami buat. Class diagram atau diagram kelas merupakan suatu diagram yang digunakan untuk menampilkan kelas-kelas berupa paket-paket untuk memenuhi salah satu kebutuhan paket yang akan digunakan nantinya.

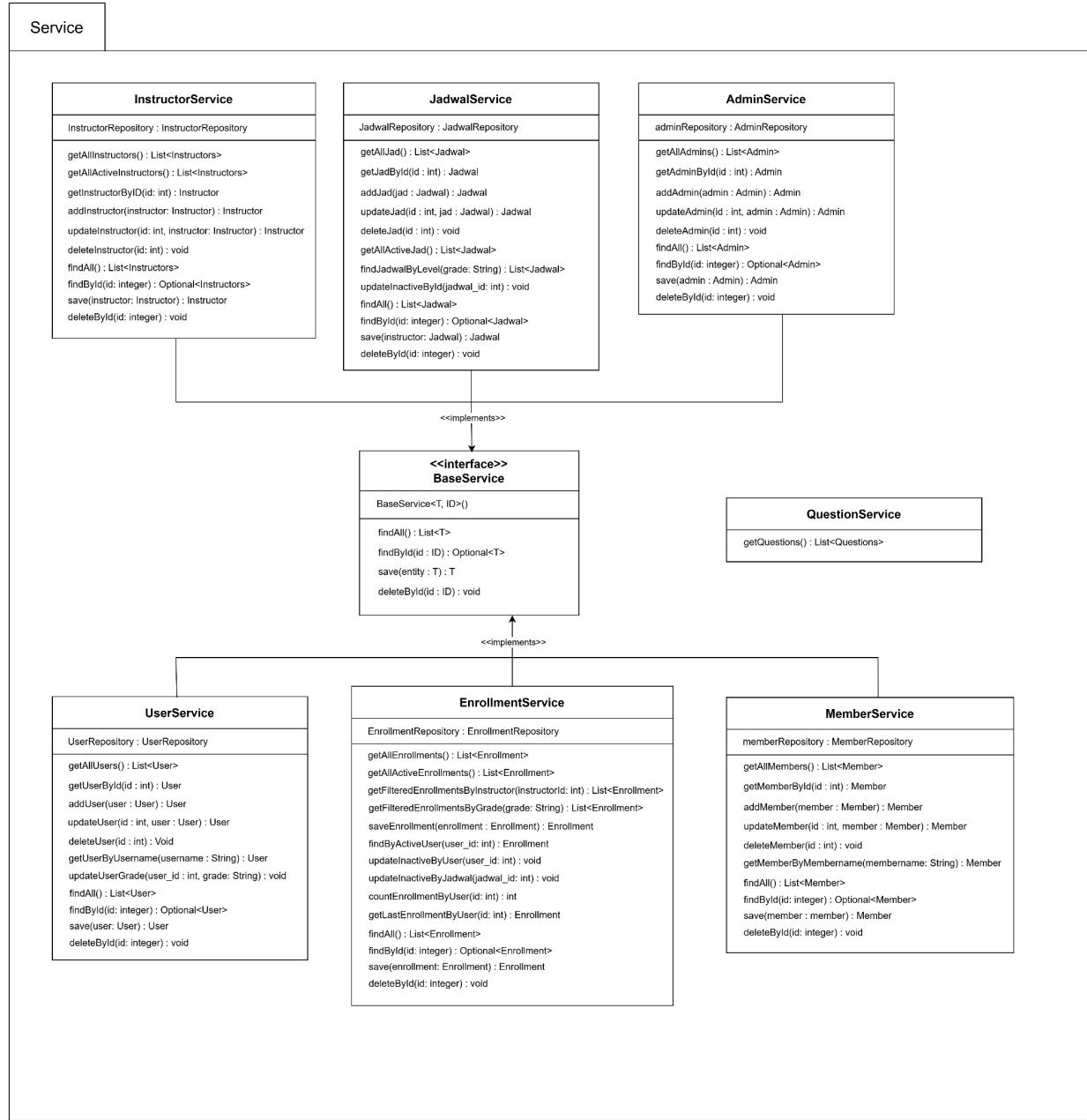
Class diagram pada kelompok kami dibagi menjadi 4 bagian, yaitu : *entity*, *repository*, *service*, dan *controller* sesuai dengan hasil codingan kami. Berikut adalah gambar dari class diagram yang kami buat. *Entity* adalah representasi dari tabel yang ada di *database*. *Repository* adalah lapisan yang bertanggung jawab untuk berkomunikasi dengan *database*. *Service* bertujuan untuk menghubungkan *repository* dan *controller*. *Controller* bertujuan untuk menghubungkan *service* dengan *view*. *Controller* dapat digunakan untuk menampilkan data ke halaman *html*.



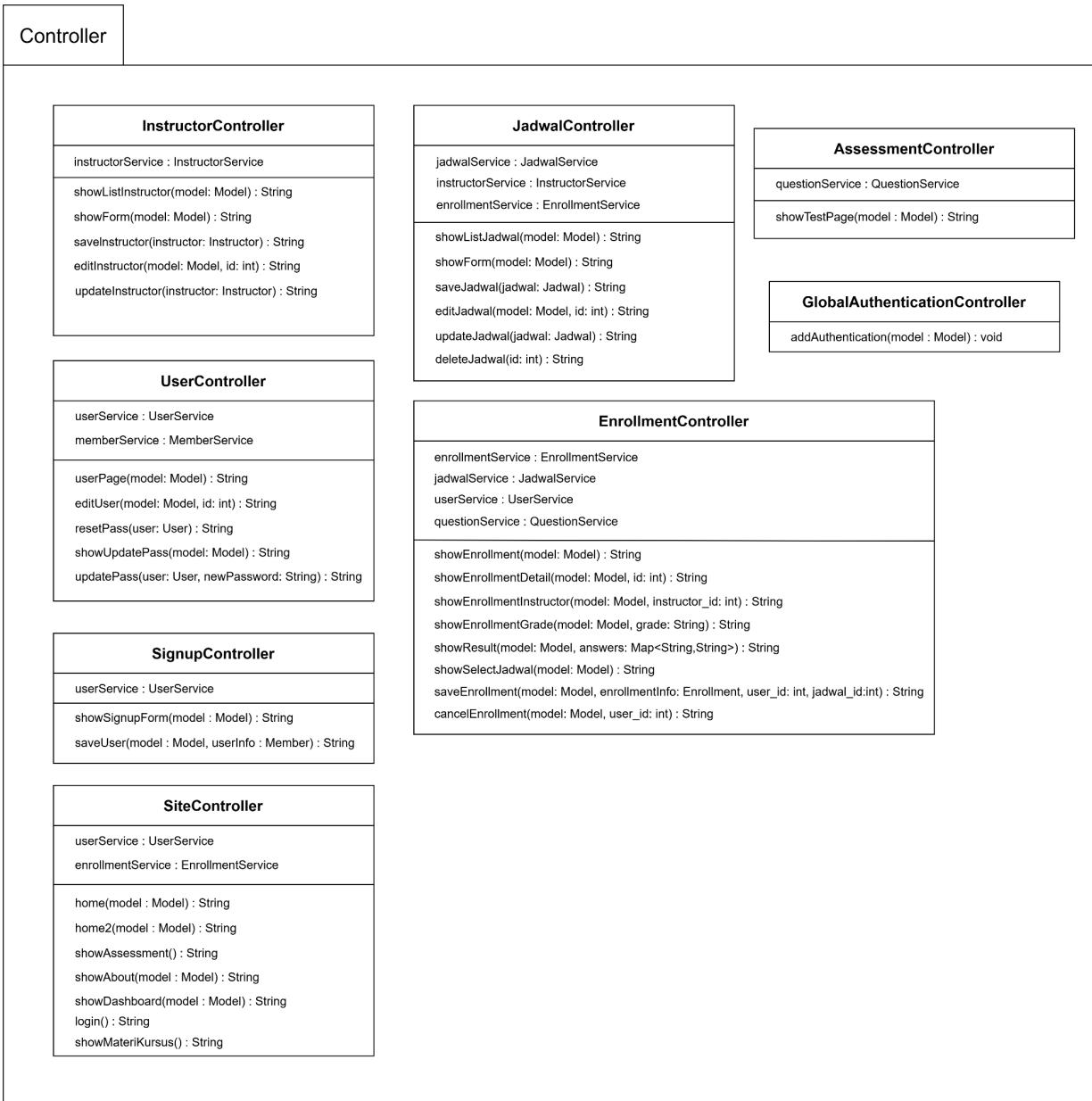
Gambar 2.2.1 Class Diagram Entity untuk Sistem Pendaftaran dan Penjadwalan Lembaga Kursus



Gambar 2.2.2 Class Diagram Repository untuk Sistem Pendaftaran dan Penjadwalan Lembaga Kursus



Gambar 2.2.3 Class Diagram Service untuk Sistem Pendaftaran dan Penjadwalan Lembaga Kursus



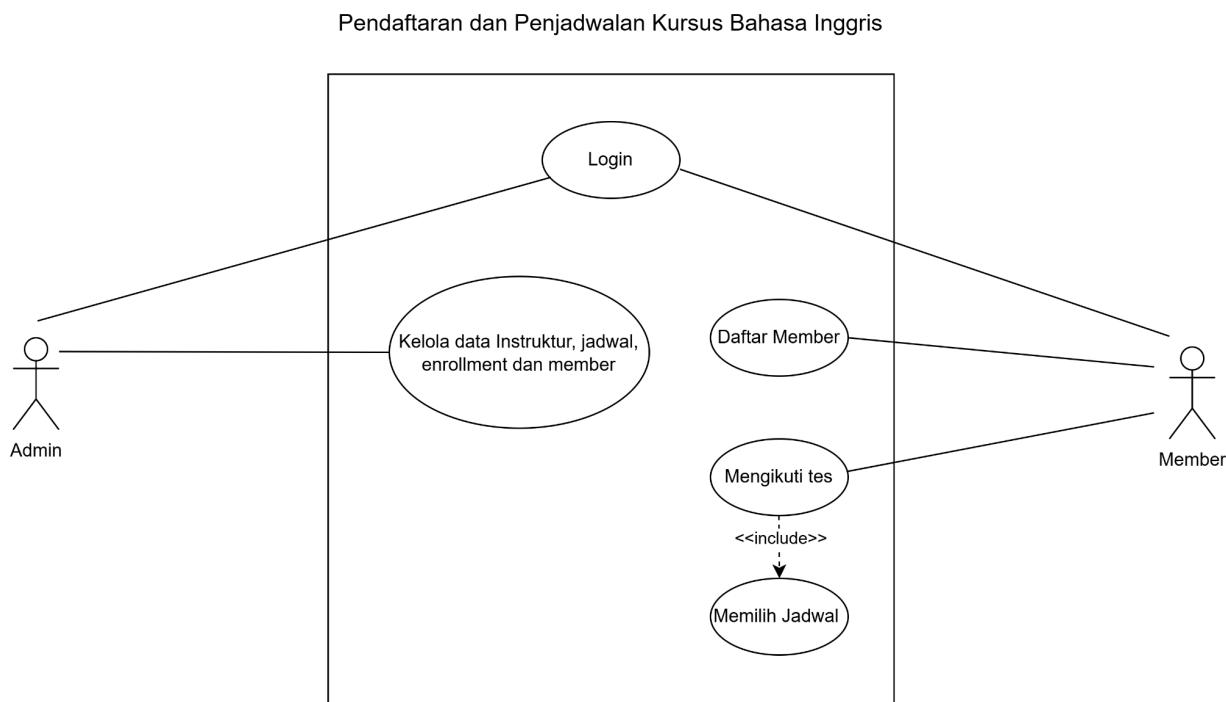
Gambar 2.2.4 Class Diagram Controller untuk Sistem Pendaftaran dan Penjadwalan Lembaga Kursus

2.3. Use Case Diagram

Sebelum kami membuat codingan untuk sistem pendaftaran dan penjadwalan lembaga kursus kami, untuk tahap pertama kami membuat *use case diagram*. *Use case*

adalah satu jenis dari diagram UML yang menggambarkan hubungan interaksi antara sistem dan aktor.

Use case diagram yang kami buat meliputi dua aktor, yaitu admin dan member. Admin dapat mengelola data instruktur, jadwal, member, dan enrollment. Member dapat melakukan pendaftaran, mengikuti tes, dan memilih jadwal sesuai dengan hasil tesnya.



Gambar 2.3.1 Use Case Diagram untuk Sistem Pendaftaran dan Penjadwalan Lembaga Kursus

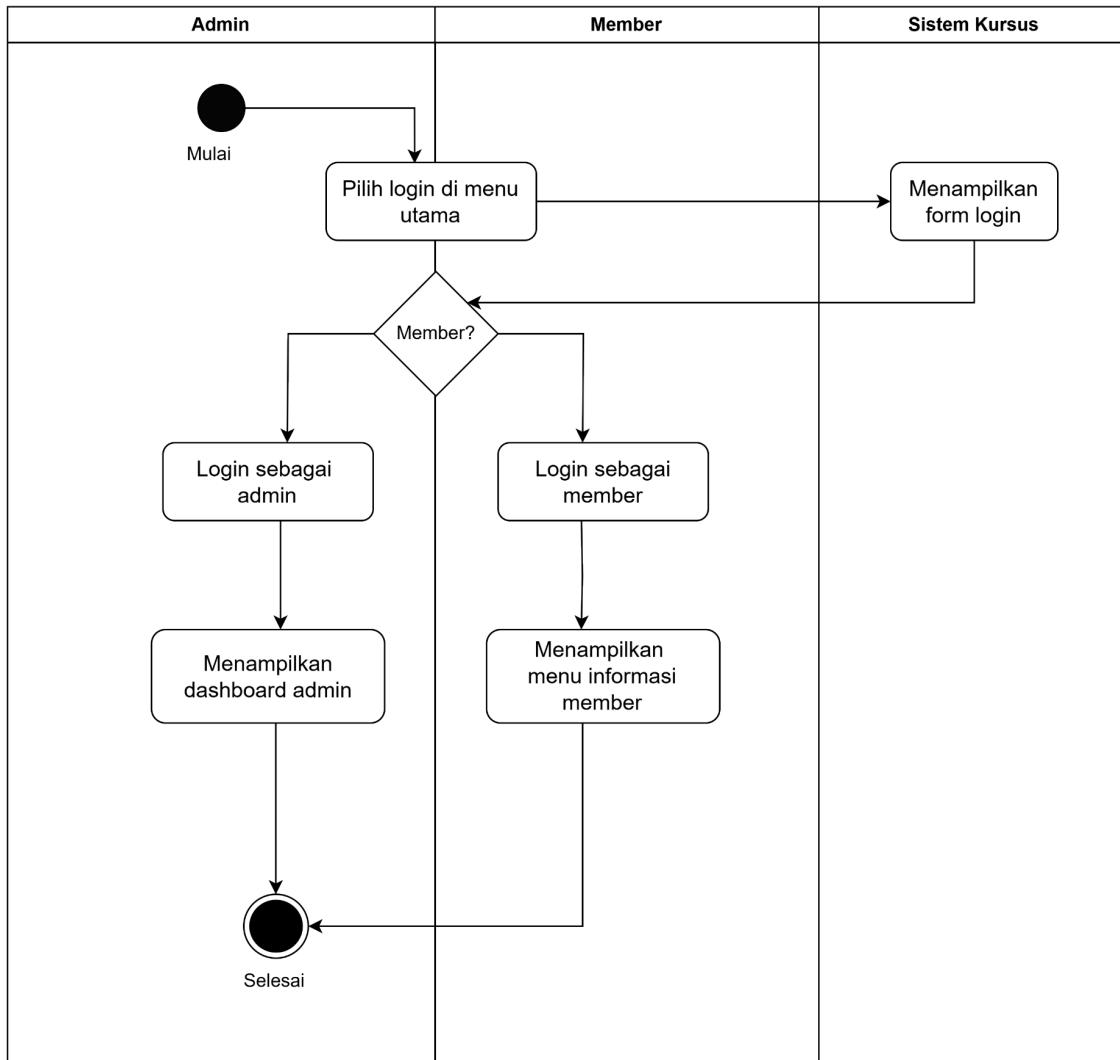
2.4. Activity Diagram

Setelah kami membuat *use case* diagram, tahap selanjutnya kami membuat *activity* diagram. *Activity* diagram adalah sebuah diagram yang dapat memodelkan berbagai proses yang terjadi pada sistem. *Activity* diagram adalah salah satu contoh diagram dari UML dalam pengembangan dari *use case*.

Kelompok kami membuat empat *activity* diagram, yaitu *activity* diagram login, *activity* diagram kelola data instruktur, jadwal, member, dan enrollment, *activity* diagram daftar member, dan *activity* diagram mengikuti tes dan memilih jadwal. Untuk login,

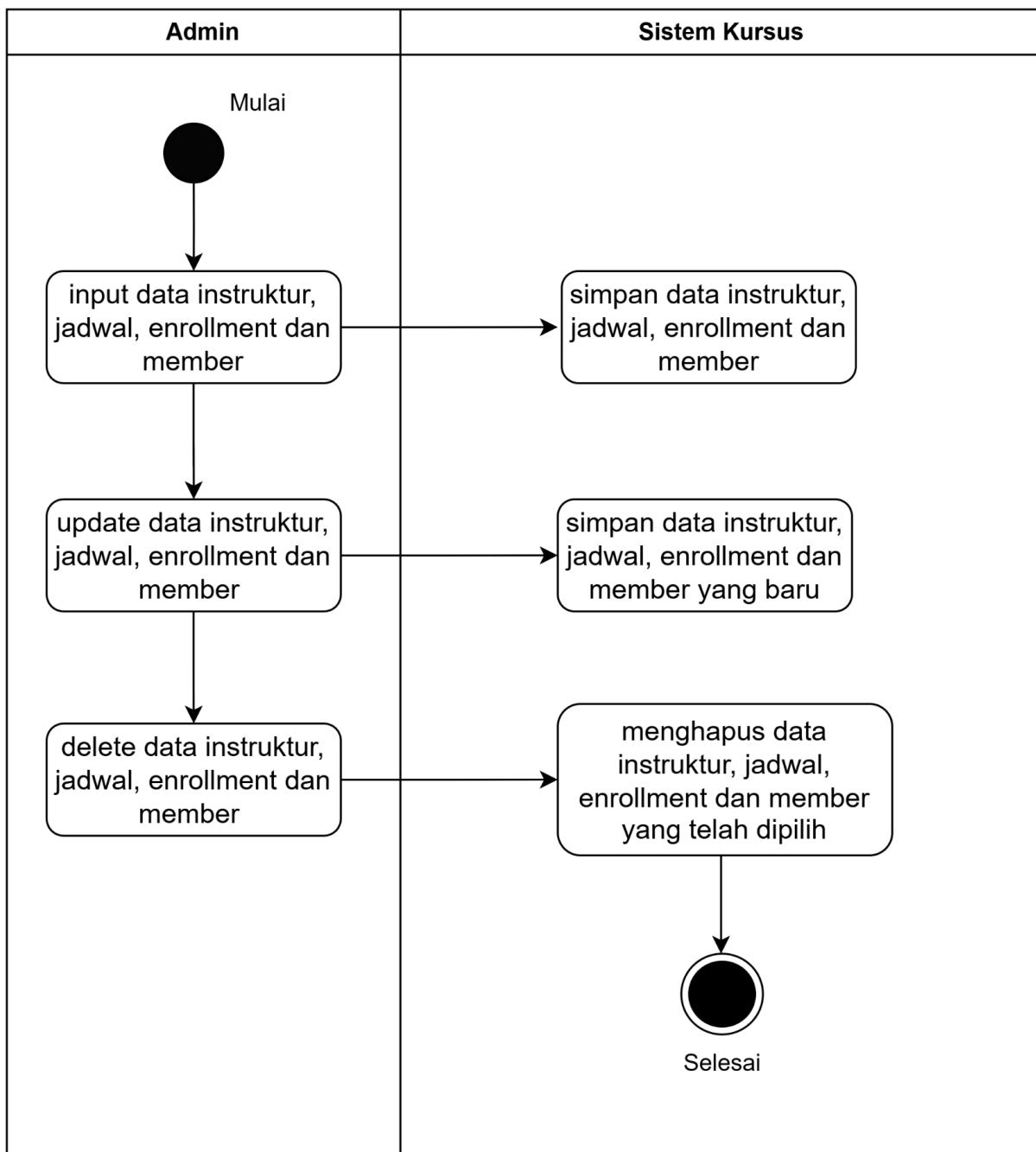
member dan admin melakukan login di satu form yang sama lalu setelah login jika login sebagai admin, maka akan masuk ke menu tampilan admin. Sebaliknya, jika login sebagai member, akan masuk ke menu tampilan member.

Activity Diagram : Login



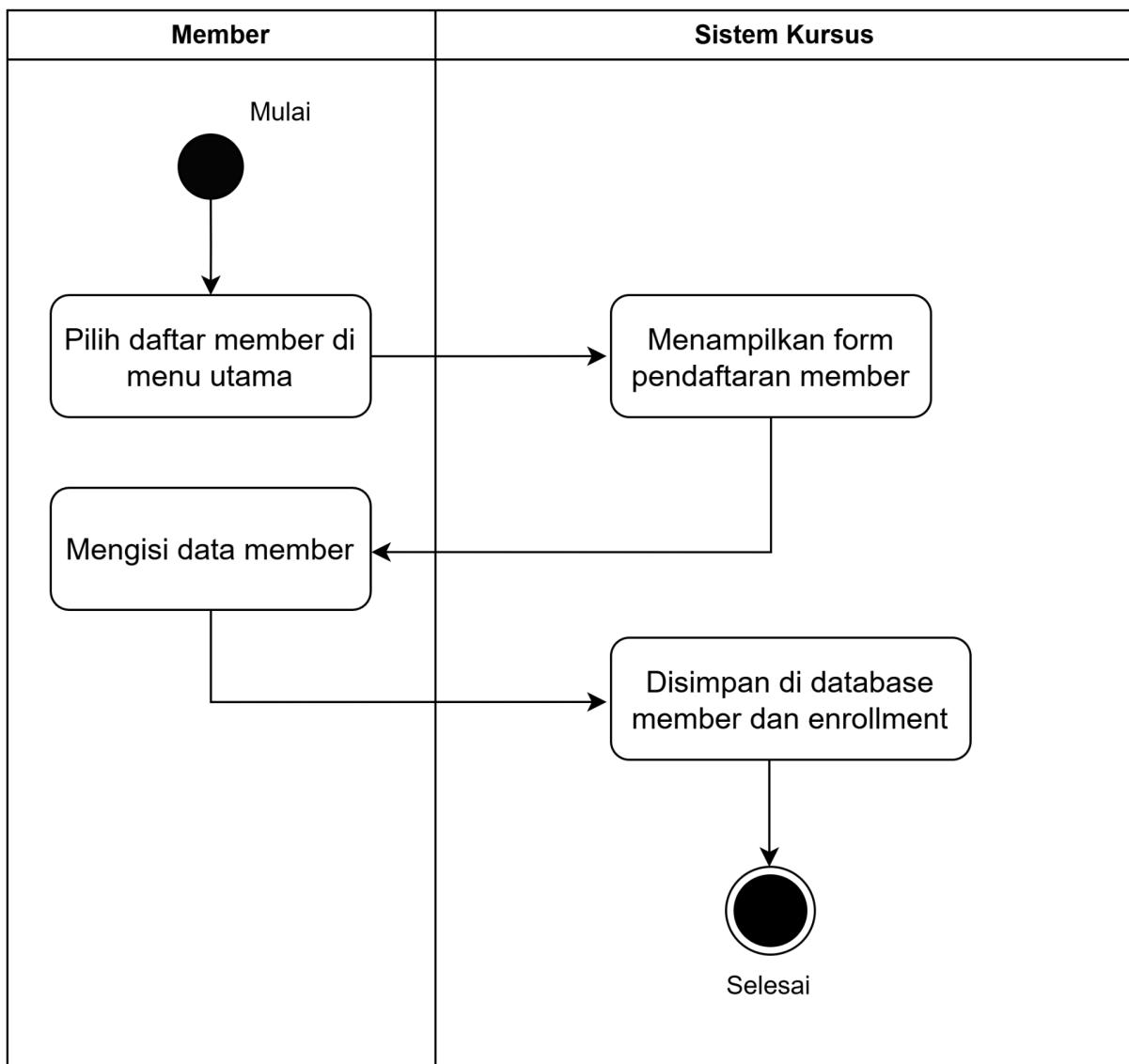
Gambar 2.4.1 Activity Diagram untuk Login

Activity Diagram Kelola data Instruktur, jadwal, enrollment dan member



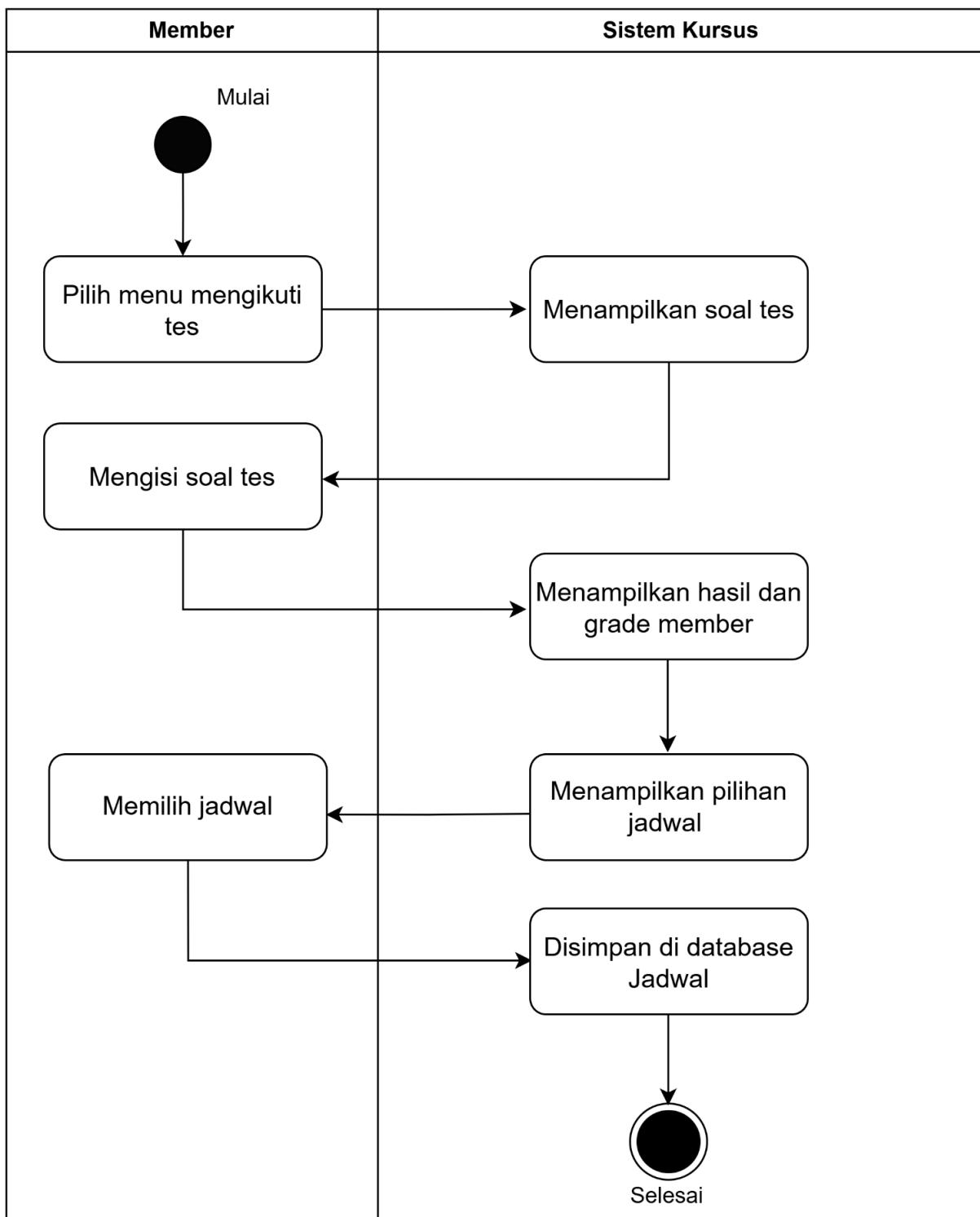
Gambar 2.4.2 Activity Diagram untuk Kelola data instruktur, jadwal, enrollment dan member

Activity Diagram Daftar Member



Gambar 2.4.3 Activity Diagram untuk daftar member

Activity Diagram Mengikuti Tes & mengikuti Jadwal Kelas



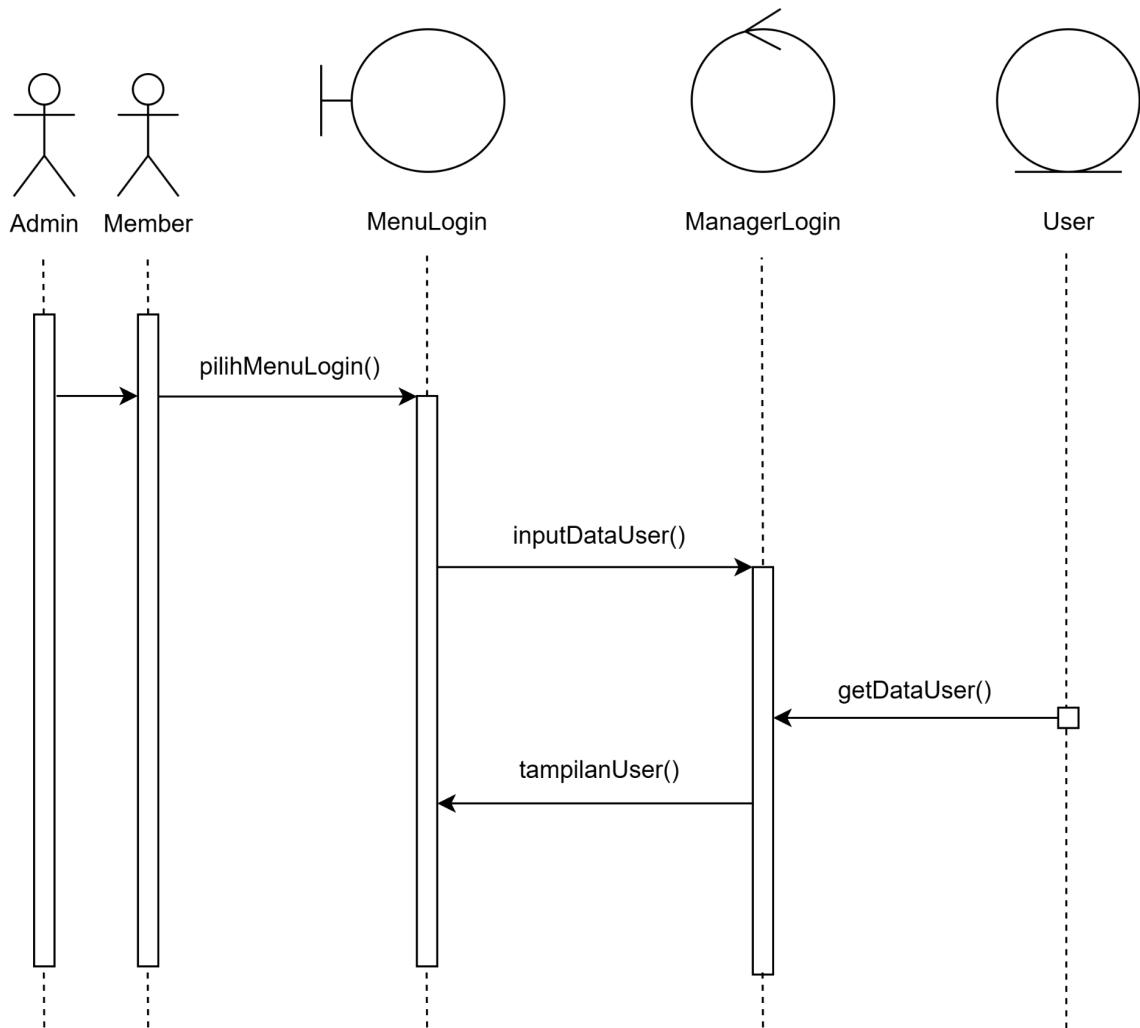
Gambar 2.4.4 Activity Diagram untuk mengikuti tes & mengikuti jadwal kelas

2.5. Sequence Diagram

Setelah membuat activity diagram, kita menuju tahap selanjutnya yaitu membuat *sequence diagram*. *Sequence diagram* adalah diagram yang menjelaskan interaksi objek berdasarkan urutan waktu. Sequence dapat menggambarkan urutan atau tahapan yang harus dilakukan untuk dapat menghasilkan sesuatu.

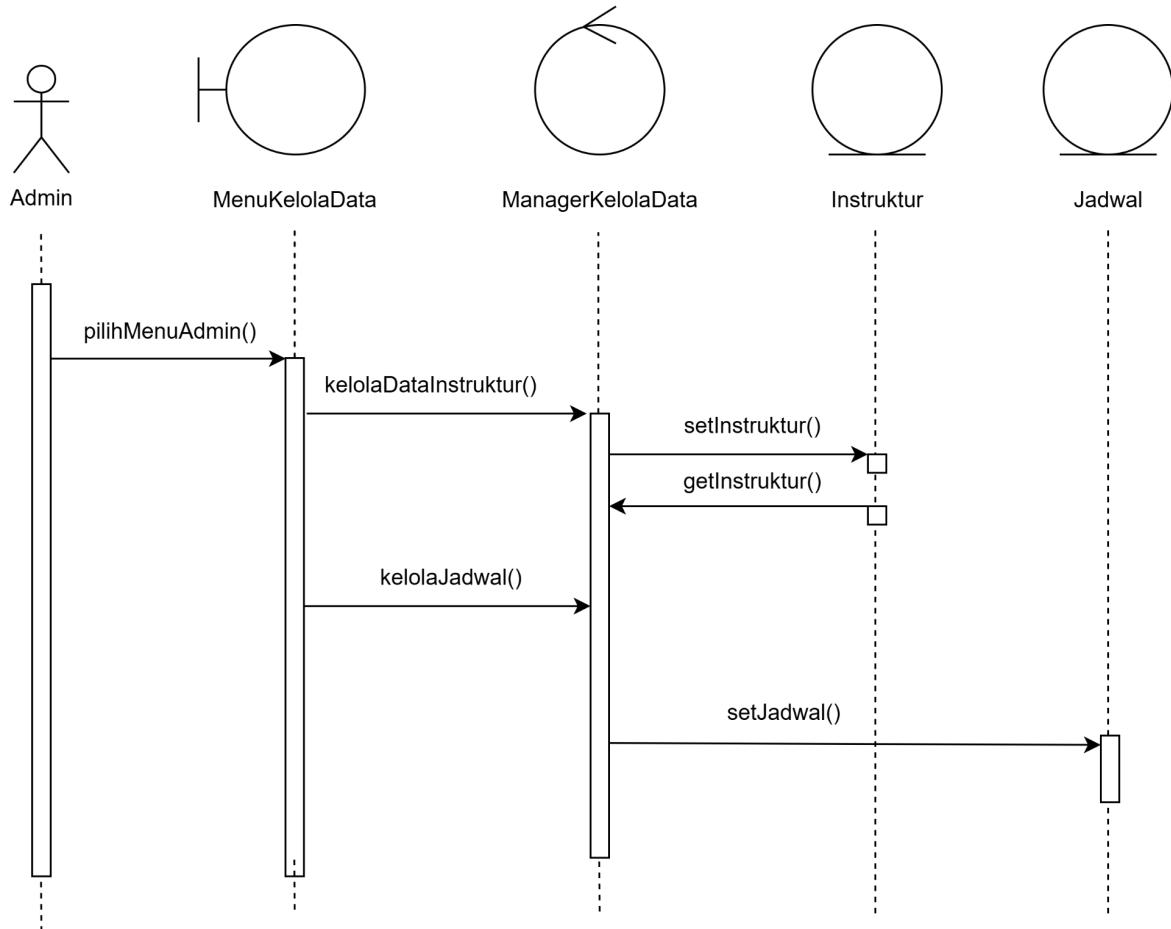
Sequence diagram yang kami buat berdasarkan *activity diagram* kami. Jadi, kami juga membuat empat sequence diagram, yaitu login, kelola data instruktur, jadwal, member, dan *enrollment*, daftar member, dan mengikuti tes dan memilih jadwal.

Sequence Diagram : Login



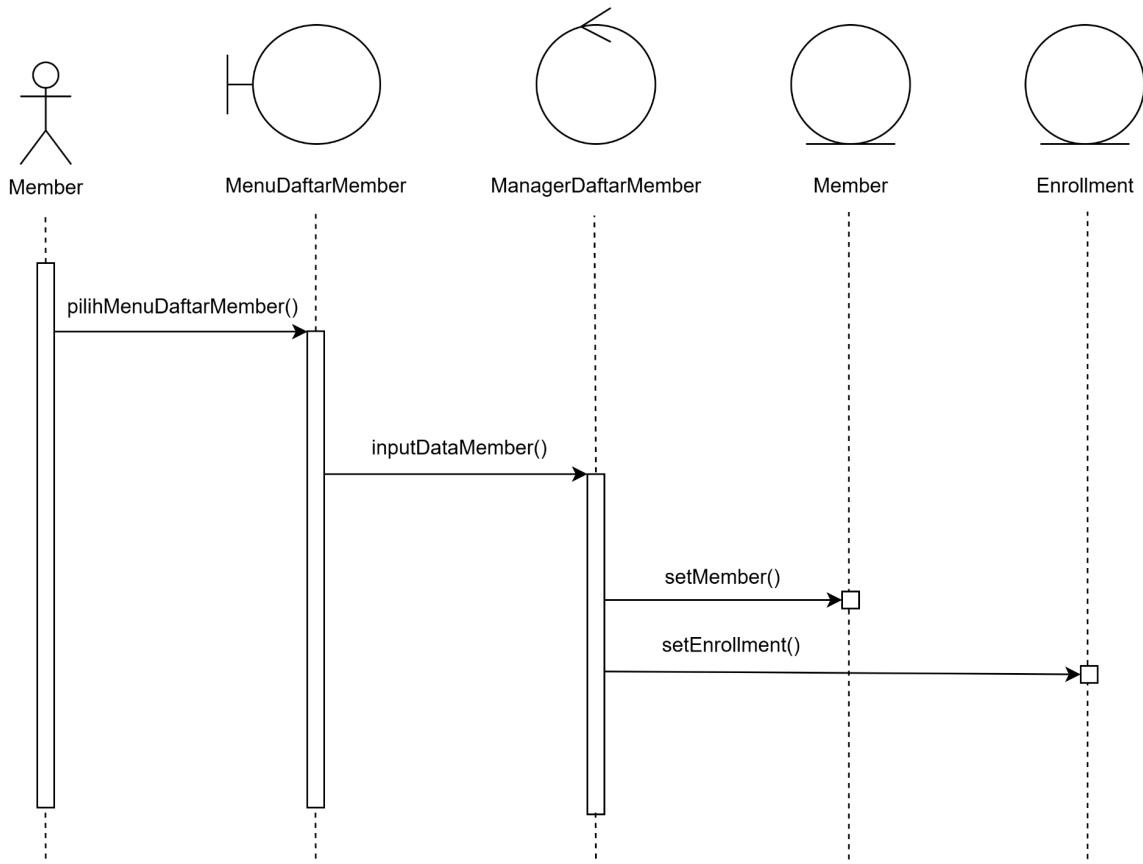
Gambar 2.5.1 Sequence Diagram untuk Login

Sequence Diagram : Kelola Data Instruktur, Jadwal dan Member



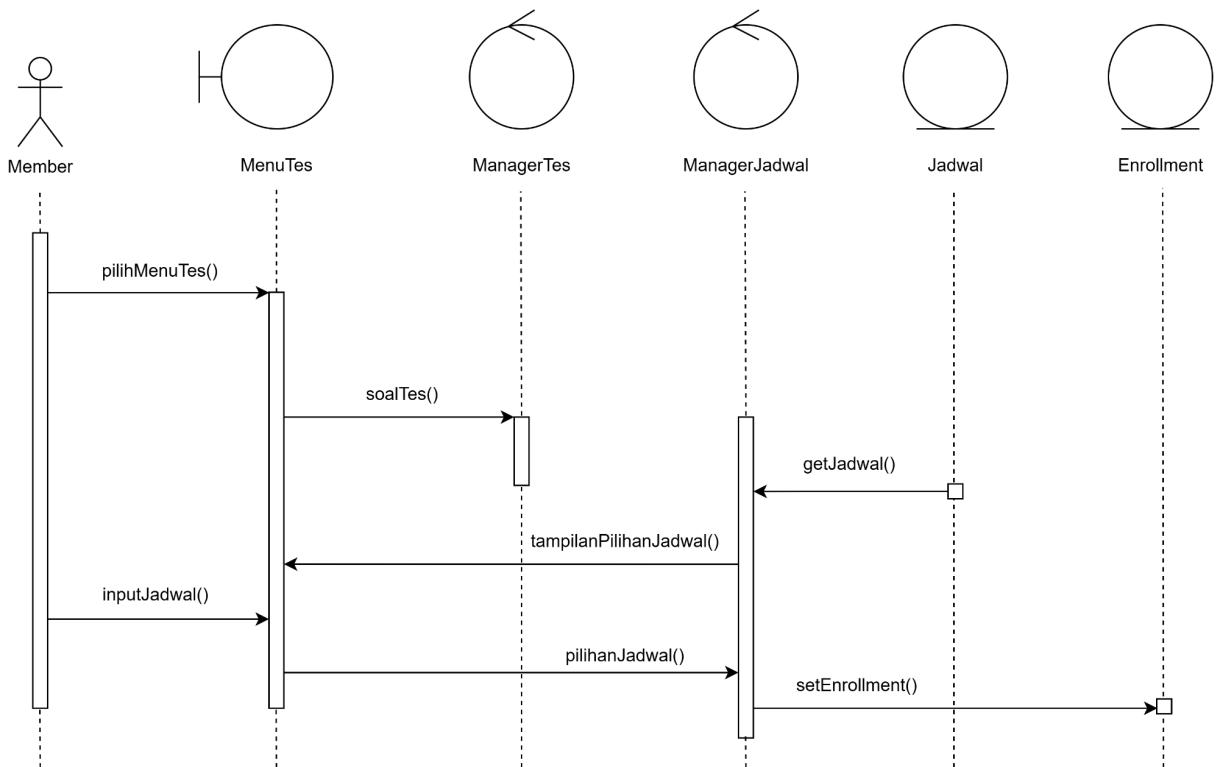
Gambar 2.5.2 Sequence Diagram untuk Kelola data instruktur, jadwal dan member

Sequence Diagram : Daftar Member



Gambar 2.5.3 Sequence Diagram untuk Daftar member

Sequence Diagram : Mengikuti Tes dan Memilih Jadwal



Gambar 2.5.4 Sequence Diagram untuk Mengikuti tes dan memilih jadwal

BAB III

PEMBUATAN SISTEM

3.1. Implementasi Sistem yang Dirancang

Pada tahap ini, kelompok kami menggunakan bahasa pemrograman *java* dan framework *Springboot* untuk membuat sistem penjadwalan dan pendaftaran lembaga kursus. Menurut microsoft *Java SpringBoot* adalah alat sumber terbuka yang memudahkan penggunaan kerangka kerja berbasis *Java* untuk membuat layanan mikro dan aplikasi web.

Untuk membuat sistem pendaftaran dan penjadwalan lembaga kursus pertama-tama kami perlu *import* beberapa *library* dari *java* dan *Springboot*. Berikut adalah beberapa *library* yang kami *import* dari *java* dan *Springboot* beserta dengan penjelasan fungsinya.

- **import jakarta.persistence.Column;**
Untuk memetakan atribut kelas ke kolom dalam tabel
- **import jakarta.persistence.Entity;**
Untuk menandakan bahwa kelas ini adalah entitas yang akan dipetakan ke tabel database
- **import jakarta.persistence.GeneratedValue;**
Untuk mengatur strategi pengisian nilai primary key secara otomatis
- **import jakarta.persistence.GenerationType;**
Untuk memilih strategi secara otomatis berdasarkan konfigurasi penyedia database yang digunakan
- **import jakarta.persistence.Id;**
Untuk menandakan atribut sebagai primary key
- **import jakarta.persistence.JoinColumn;**
Untuk menyatakan hubungan antar entitas dalam JPA
- **import jakarta.persistence.ManyToOne;**
Untuk menyatakan hubungan antar entitas *many to one* dalam JPA
- **import jakarta.persistence.Table;**
Untuk menentukan nama tabel database
- **import java.time.format.DateTimeFormatter;**
Untuk memformat tanggal menjadi string atau sebaliknya

- **import java.time.LocalDate;**
Untuk merepresentasikan tanggal tanpa waktu
- **import java.util.ArrayList;**
Untuk menyimpan struktur data yang umum digunakan
- **import java.util.List;**
Untuk menyimpan struktur data yang umum digunakan
- **import java.util.Map;**
Untuk menyimpan struktur data yang umum digunakan
- **import java.util.Optional;**
Untuk menyimpan struktur data yang umum digunakan
- **import java.util.stream.Collectors;**
Untuk memproses *stream* data dengan cara tertentu (misalnya mengelompokan atau menggabungkan)
- **import org.springframework.beans.factory.annotation.Autowired;**
Untuk menyuntikkan *bean* ke dalam aplikasi kami
- **import org.springframework.boot.autoconfigure.SpringBootApplication;**
Untuk mengaktifkan fitur-fitur dasar springboot
- **import org.springframework.boot.SpringApplication;**
Untuk menjalankan aplikasi springboot
- **import org.springframework.context.annotation.Bean;**
Untuk menandakan metode sebagai sumber *bean* untuk manajemen konteks aplikasi
- **import org.springframework.context.annotation.Configuration;**
Untuk menandakan kelas sebagai tempat konfigurasi spring
- **import org.springframework.data.jpa.repository.JpaRepository;**
Untuk antarmuka yang menyediakan CRUD untuk entitas
- **import org.springframework.data.jpa.repository.Modifying;**
Untuk menandakan bahwa query akan memodifikasi data
- **import org.springframework.data.jpa.repository.Query;**
Untuk mendefinisikan query SQL secara *custom*
- **import
org.springframework.security.authentication.dao.DaoAuthenticationProvider;**
Untuk penyedia autentikasi menggunakan data pengguna dari database
- **import
org.springframework.security.config.annotation.web.builders.HttpSecurity;**
Untuk mengkonfigurasi keamanan aplikasi web
- **import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;**

Untuk mengaktifkan konfigurasi keamanan berbasis *web* dalam aplikasi *springboot*

- **import org.springframework.security.core.Authentication;**
Untuk memeriksa status autentikasi pengguna
- **import org.springframework.security.core.context.SecurityContextHolder;**
Untuk memegang informasi autentikasi dan keamanan pengguna saat ini
- **import org.springframework.security.core.userdetails.UserDetails;**
Untuk merepresentasikan data pengguna yang diperlukan untuk autentikasi dan otorisasi
- **import org.springframework.security.core.userdetails.UserDetailsService;**
Untuk memuat data pengguna berdasarkan nama pengguna
- **import org.springframework.security.core.userdetails.UsernameNotFoundException;**
Untuk error yang sering digunakan di dalam Spring security
- **import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;**
Untuk mengenkripsi atau mengacak kata sandi
- **import org.springframework.security.crypto.password.PasswordEncoder;**
Untuk mengenkripsi kata sandi dan memeriksa kata sandi apakah sudah cocok dengan database atau belum
- **import org.springframework.security.web.SecurityFilterChain;**
Untuk memeriksa apakah pengguna sudah login dan apakah pengguna punya izin untuk mengakses halaman tertentu
- **import org.springframework.stereotype.Controller;**
Untuk menandai kelas sebagai *controller*
- **import org.springframework.stereotype.Repository;**
Untuk menandai kelas sebagai *repository*
- **import org.springframework.stereotype.Service;**
Untuk menandai kelas sebagai *service*
- **import org.springframework.transaction.annotation.Transactional;**
Untuk aplikasi berinteraksi dengan database
- **import org.springframework.ui.Model;**
Untuk mengirim data dari *controller* ke tampilan atau *view*
- **import org.springframework.web.bind.annotation.*;**
Untuk memasukan semua anotasi yang disediakan oleh *spring web* di dalam paket
- **import org.springframework.web.bind.annotation.ControllerAdvice;**
Untuk menangani *error* atau pengecualian yang muncul di aplikasi
- **import org.springframework.web.bind.annotation.GetMapping;**

Untuk menangani permintaan HTTP GET, biasanya saat pengguna mengunjungi URL

- **import org.springframework.web.bind.annotation.ModelAttribute;**
Untuk memproses data dari formulir dan mengikatnya ke objek java
- **import org.springframework.web.bind.annotation.PathVariable;**
Untuk mengambil data langsung dari URL
- **import org.springframework.web.bind.annotation.PostMapping;**
Untuk menangani permintaan HTTP POST, biasanya saat pengguna mengirimkan data melalui formulir
- **import org.springframework.web.bind.annotation.RequestParam;**
Untuk mengambil data dari parameter query di URL

3.2. Implementasi Abstract Class User

Abstract class user merepresentasikan entitas umum dari pengguna dalam sistem yang kami buat. Class ini mendefinisikan atribut-atribut yang dimiliki oleh semua jenis pengguna.

Atribut di abstract class user kami mencakup id, username, password, role, dan email. Selain itu, ada juga dependency yang kami tambahkan di pom.xml untuk login kami.

```
19
20  @Entity
21  @Inheritance(strategy = InheritanceType.SINGLE_TABLE)
22  @Table(name = "user")
23
24  public abstract class User {
25      @Id
26      @GeneratedValue(strategy = GenerationType.IDENTITY)
27      private int id;
28
29      @Column(name = "username")
30      private String username;
31
32      @Column(name = "password")
33      private String password;
34
35      @Column(name = "role")
36      private String role;
37
38      @Column(name = "email")
39      private String email;
40
```

Gambar 3.2.1 Abstract Class User

```
<!-- BUAT LOGIN -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Gambar 3.2.2 dependency tambahan

3.3. Implementasi Class

Implementasi class dilakukan berdasarkan class diagram yang kami buat di bab sebelumnya. Sistem pendaftaran dan penjadwalan kursus kami terdiri dari berbagai class yang memiliki tanggung jawab spesifik untuk merepresentasikan pengguna, jadwal, dan pendaftaran.

Class yang kami buat meliputi class Enrollment, class Instructor, class Jadwal, class Member, class Question, class UserDetailServiceImpl, class WebSecurityConfig, class AdminService, EnrollmentService, InstructorService, JadwalService, MemberService, QuestionService, UserService, AssessmentController, EnrollmentController, GlobalAuthenticationController, InstructorController, JadwalController, SignupController, SiteController, dan UserController. Class yang kami buat sudah mencakup semua entitas, controller, repository, service, dan security.

- Implementasi Class Enrollment

```
@Entity
@Table(name = "enrollment")
public class Enrollment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "tanggal")
    private LocalDate tanggal;

    @ManyToOne // Anotasi untuk membuat foreign key
    @JoinColumn(name = "user_id", nullable = false) // Foreign key ke tabel Member
    private User user;

    @ManyToOne // Anotasi untuk membuat foreign key
    @JoinColumn(name = "jadwal_id", nullable = false) // Foreign key ke tabel Member
    private Jadwal jadwal;

    @Column(name = "active_status")
    private boolean active_status;

    //SETTER GETTER
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

Gambar 3.3.1 Class Enrollment

- Implementasi Class Instructor

```
8  @Entity
9  @Table(name = "instructor")
10 |
11 public class Instructor {
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private int id;
15
16     @Column(name = "name")
17     private String name;
18
19     @Column(name = "active_status")
20     private boolean active_status;
21
22     public Instructor() {}
23
24     public Instructor(String name, boolean active_status) {
25         this.name = name;
26         this.active_status = active_status;
27     }
28
29     public int getId() {
30         return id;
31     }
32
33     public void setId(int id) {
34         this.id = id;
35     }
```

Gambar 3.3.2 Class Instructor

- Implementasi Class Jadwal

```
@Entity
@Table(name = "jadwal")
public class Jadwal {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

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

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

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

    @Column(name = "active_status")
    private boolean active_status;

    @ManyToOne // Anotasi untuk membuat foreign key
    @JoinColumn(name = "instructor_id", nullable = false) // Foreign key ke tabel Member
    private Instructor instructor;

    public Jadwal() {}
    public Jadwal(String grade, Instructor instructor) {
        this.grade = grade;
    }
}
```

Gambar 3.3.3 Class Jadwal

- Implementasi Class Member

```
@Entity
public class Member extends User {

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

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

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

    @Column(name="birthdate")
    private LocalDate birthdate;

    public String getFullName() {
        return firstname + " " + lastname;
    }

    public String getGrade() {
        return grade;
    }

    public void setGrade(String grade) {
        this.grade = grade;
    }

    public void setFirstname(String firstname) {
```

Gambar 3.3.4 Class Member

- Implementasi Class Question

```
package com.example.demospringboot.entity;

import java.util.List;

public class Question {
    private String questionText;
    private List<String> options;
    private String correctAnswer;

    public Question(String questionText, List<String> options, String correctAnswer) {
        this.questionText = questionText;
        this.options = options;
        this.correctAnswer = correctAnswer;
    }

    public String getQuestionText() {
        return questionText;
    }

    public List<String> getOptions() {
        return options;
    }

    public String getCorrectAnswer() {
        return correctAnswer;
    }
}
```

Gambar 3.3.5 Class Question

- Implementasi Class UserDetailServiceImpl

```

@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    private final UserRepository userRepository;

    @Autowired
    public UserDetailsServiceImpl(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepository.findByUsername(username);
        if (user == null) {
            throw new UsernameNotFoundException(msg:"User not found");
        }
        return new org.springframework.security.core.userdetails.User(
            user.getUsername(),
            user.getPassword(),
            user.getAuthorities());
    }
}

```

Gambar 3.3.6 Class UserDetailServiceImpl

- Implementasi Class WebSecurityConfig

```

@Configuration
@EnableWebSecurity
public class WebSecurityConfig {

    private final UserDetailsServiceImpl userDetailsService;

    public WebSecurityConfig(UserDetailsServiceImpl userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
    // source:https://docs.spring.io/spring-security/reference/servlet/configuration/java.html

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/*").permitAll()
                .requestMatchers("/home", "/about", "/signup/**", "/result").permitAll()
                .anyRequest().authenticated());
    }
}

```

Gambar 3.3.7 Class WebSecurityConfig

- Implementasi Class AdminService

```
@Service
public class AdminService implements BaseService<Admin, Integer>{
    @Autowired
    private AdminRepository adminRepository;
    public List<Admin> getAllAdmins() {
        return adminRepository.findAll();
    }
    public Admin getAdminById(int id) {
        return adminRepository.findById(id).orElse(null);
    }
    public Admin addAdmin(Admin admin) {
        String pass = admin.getPassword();
        String hashedpass = admin.getEncryptedPass(pass);
        admin.setPassword(hashedpass);
        return adminRepository.save(admin);
    }
    public Admin updateAdmin(int id, Admin admin) {
        admin.setId(id);
        return adminRepository.save(admin);
    }
    public void deleteAdmin(int id) {
        adminRepository.deleteById(id);
    }
}
```

Gambar 3.3.8 Class AdminService

- Implementasi Class EnrollmentService

```

@Service
public class EnrollmentService implements BaseService<Enrollment, Integer>{
    @Autowired
    private EnrollmentRepository enrollmentRepository;

    public List<Enrollment> getAllEnrollments() {
        return enrollmentRepository.findAll();
    }

    public List<Enrollment> getAllActiveEnrollments() {
        List<Enrollment> enrollments = enrollmentRepository.findAllActive();

        //urutkan dulu dengan sorted() kemudian ditampung ke collector
        return enrollments.stream()
            .sorted((e1, e2) -> e2.getJadwal().getGrade().compareTo(e1.getJadwal().getGrade())) // Urutkan Descending Pemula
            .collect(Collectors.toList());

        //return enrollmentRepository.findAllActive();
    }

    public List<Enrollment> getFilteredEnrollmentsByInstructor(int instructorId) {
        List<Enrollment> enrollments = enrollmentRepository.findAllActive(); // Ambil semua data Enrollment

        // Filter berdasarkan instructor_id dari entitas Jadwal
        return enrollments.stream()
            .filter(enrollment -> enrollment.getJadwal().getInstructor().getId() == instructorId)
            .collect(Collectors.toList());
    }
}

```

Gambar 3.3.9 Class EnrollmentService

- Implementasi Class InstructorService

```
@Service
public class InstructorService implements BaseService<Instructor, Integer> {
    @Autowired
    private InstructorRepository instructorRepository;
    public List<Instructor> getAllInstructors() {
        return instructorRepository.findAll();
    }

    public List<Instructor> getAllActiveInstructors() {
        return instructorRepository.findAllActive();
    }

    public Instructor getInstructorById(int id) {
        return instructorRepository.findById(id).orElse(null);
    }

    public Instructor addInstructor(Instructor instructor) {
        return instructorRepository.save(instructor);
    }
    public Instructor updateInstructor(int id, Instructor instructor) {
        instructor.setId(id);
        return instructorRepository.save(instructor);
    }
    public void deleteInstructor(int id) {
        instructorRepository.deleteById(id);
    }
}
```

Gambar 3.3.10 Class InstructorService

- Implementasi Class JadwalService

```
@Service
public class JadwalService implements BaseService<Jadwal, Integer>{
    @Autowired
    private JadwalRepository jadwalRepository;
    public List<Jadwal> getAllJad() {
        return jadwalRepository.findAll();
    }
    public Jadwal getJadById(int id) {
        return jadwalRepository.findById(id).orElse(null);
    }
    public Jadwal addJad(Jadwal jad) {
        return jadwalRepository.save(jad);
    }
    public Jadwal updateJad(int id, Jadwal jad) {
        jad.setId(id);
        return jadwalRepository.save(jad);
    }
    public void deleteJad(int Id) {
        jadwalRepository.deleteById(Id);
    }

    public List<Jadwal> getAllActiveJad()
    {
        List<Jadwal> jadwals = jadwalRepository.findAllActive();
    }
}
```

Gambar 3.3.11 Class JadwalService

- Implementasi Class MemberService

```
@Service
public class MemberService implements BaseService<Member, Integer>{
    @Autowired
    private MemberRepository memberRepository;
    public List<Member> getAllMembers() {
        return memberRepository.findAll();
    }
    public Member getMemberById(int id) {
        return memberRepository.findById(id).orElse(null);
    }
    public Member addMember(Member member) {
        String pass = member.getPassword();
        String hashedpass = member.getEncryptedPass(pass);
        member.setPassword(hashedpass);
        return memberRepository.save(member);
    }
    public Member updateMember(int id, Member member) {
        member.setId(id);
        return memberRepository.save(member);
    }
    public void deleteMember(int id) {
        memberRepository.deleteById(id);
    }
}
public Member getMemberByMembername(String membername) {
    return memberRepository.findByUsername(membername);
}
```

Gambar 3.3.12 Class MemberService

- Implementasi Class QuestionService

```
package com.example.demospringboot.service;

import com.example.demospringboot.entity.Question;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
public class QuestionService {
    public List<Question> getQuestions() {
        List<Question> questions = new ArrayList<>();
        questions.add(new Question(questionText:"What is the correct past tense of 'go'?", correctAnswer:"B"));
        questions.add(new Question(questionText:"Which one is the synonym of 'happy'?", correctAnswer:"B"));
        questions.add(new Question(questionText:"What is the opposite of 'fast'?", correctAnswer:"A"));
        // Tambahkan soal lain sesuai kebutuhan
        return questions;
    }
}
```

Gambar 3.3.13 Class QuestionService

- Implementasi Class UserService

```
@Service
public class UserService implements BaseService<User, Integer>{
    @Autowired
    private UserRepository userRepository;
    public List<User> getAllUsers() {
        return userRepository.findAll();
    }
    public User getUserById(int id) {
        return userRepository.findById(id).orElse(null);
    }
    public User addUser(User user) {
        String pass = user.getPassword();
        String hashedpass = user.getEncryptedPass(pass);
        user.setPassword(hashedpass);
        return userRepository.save(user);
    }
    public User updateUser(int id, User user) {
        user.setId(id);
        return userRepository.save(user);
    }
    public void deleteUser(int id) {
        userRepository.deleteById(id);
    }

    public User getUserByUsername(String username) {
        return userRepository.findByUsername(username);
    }
}
```

Gambar 3.3.14 Class UserService

- Implementasi Class AssessmentController

```

@Controller
public class AssessmentController {

    @Autowired
    private QuestionService questionService;

    @GetMapping("/test")
    public String showTestPage(Model model) {
        List<Question> questions = questionService.getQuestions();
        model.addAttribute(attributeName:"questions", questions);
        return "assessment/english-test";
    }
}

```

Gambar 3.3.15 Class AssessmentController

- Implementasi Class EnrollmentController

```

@Controller
public class EnrollmentController {
    @Autowired
    private EnrollmentService enrollmentService;

    @Autowired
    private JadwalService jadwalService;

    @Autowired
    private UserService userService;

    @Autowired
    private QuestionService questionService;

    @GetMapping("/enrollment")
    public String showEnrollment(Model model){
        model.addAttribute(attributeName:"enrollmentList", enrollmentService.getAllActiveEnrollments());
        return "enrollment/enrollment-list.html";
    }

    // menampilkan detail enrollment

    @GetMapping("/enrollment/{id}")
    public String showEnrollmentDetail(Model model,@PathVariable("id") int id){
        //model.addAttribute("enrollmentList", enrollmentService.getAllActiveEnrollments());
        return "enrollment/enrollment-list.html";
    }
}

```

Gambar 3.3.16 Class EnrollmentController

- Implementasi Class GlobalAuthenticationController

```

@ControllerAdvice
public class GlobalAuthenticationController {

    @ModelAttribute
    public void addAuthentication(Model model) {
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

        if (authentication != null && authentication.isAuthenticated()) {
            // global variable, supaya bisa diakses di semua halaman
            // var username buat cek di view (thymeleaf) apakah sudah login belum
            System.out.println("hello saya ada di global controller");
            model.addAttribute(attributeName:"username", authentication.getName());
        }
    }
}

```

Gambar 3.3.17 Class GlobalAuthenticationController

- Implementasi Class InstructorController

```

@Controller
public class InstructorController {
    @Autowired
    private InstructorService instructorService;

    // menampilkan halaman dengan daftar instruktur
    @GetMapping("/instructor")
    public String showListInstructor(Model model){
        model.addAttribute(attributeName:"instructorList", instructorService.getAllInstructors());
        // List <Instructor> mylist = instructorService.getAllInstructors();
        // for (Instructor i : mylist) {
        //     System.out.println(i.getId());
        // }
        // System.out.println("ini /ins");
        //model.addAttribute("instructorInfo", new Instructor());
        return "instructor/instructor-list.html";
    }

    @GetMapping("/instructor/new")
    public String showForm(Model model) {
        model.addAttribute(attributeName:"instructor", new Instructor());
        return "instructor/instructor-form"; // Nama file template (Thymeleaf)
    }

    // Proses tambah instruktur
    @PostMapping("/instructor/save")
    public String saveInstructor(Instructor instructor) {
        instructorService.addInstructor(instructor);
    }
}

```

Gambar 3.3.18 Class InstructorController

- Implementasi Class JadwalController

```

@Controller
public class JadwalController {
    @Autowired
    private JadwalService jadwalService;

    @Autowired
    private InstructorService instructorService;

    @Autowired
    private EnrollmentService enrollmentService;

    @GetMapping("/jadwal")
    public String showListJadwal(Model model) {
        model.addAttribute(attributeName:"jadwallist", jadwalService.getAllActiveJad());
        return "jadwal/jadwal-list.html";
    }

    @GetMapping("/jadwal/new")
    public String showForm(Model model) {
        model.addAttribute(attributeName:"jadwal", new Jadwal());
        model.addAttribute(attributeName:"instructors", instructorService.getAllActiveInstructors());
        String[] arrDay = new String[0];
        model.addAttribute(attributeName:"arrDay", arrDay);
        return "jadwal/jadwal-form";
    }
}

```

Gambar 3.3.19 Class JadwalController

- Implementasi Class SignupController

```

@Controller
public class SignupController {
    @Autowired
    private UserService userService;

    @GetMapping("/signup")
    public String showSignUpForm(Model model) {
        model.addAttribute(attributeName:"userInfo", new Member());
        return "signup";
    }

    @PostMapping(value="/signup")
    public String saveUser(Model model, Member userInfo)
    {
        userService.addUser(userInfo);
    }
}

```

Gambar 3.3.20 Class SignupController

- Implementasi Class SiteController

```
@Controller
public class SiteController {
    @Autowired
    private UserService userService;
    @Autowired
    private EnrollmentService enrollmentService;

    @GetMapping("/")
    public String home(Model model) {
        return "redirect:/home"; // Halaman yang dapat diakses tanpa login
    }

    @GetMapping("/home")
    public String home2(Model model) {
        return "home";
    }

    @GetMapping("/assessment")
    public String showAssessment() {
        return "assessment";
    }

    @GetMapping("/about")
    public String showAbout(Model model) {
```

Gambar 3.3.21 Class SiteController

- Implementasi Class UserController

```

@Controller
public class UserController {
    @Autowired
    private UserService userService;
    @Autowired
    private MemberService memberService;

    @GetMapping("/user")
    public String userPage(Model model){
        @SuppressWarnings("unused")
        List<Member> userList;
        model.addAttribute(attributeName:"userList", memberService.getAllMembers());
        return "user/user-list";
    }

    // Menampilkan halaman add, edit user
    @GetMapping("/user/{id}")
    public String editUser(Model model, @PathVariable("id") int id) {
        //System.out.println("User id(form):"+id);

        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        model.addAttribute(attributeName:"roles", authentication.getAuthorities().toString());
        model.addAttribute(attributeName:"user", userService.getUserById(id));
        return "user/user-detail";
    }
}

```

Gambar 3.3.22 Class UserController

3.4. Implementasi Interface

Interface adalah sebuah kontrak yang mendeklarasikan metode-metode yang harus diimplementasikan oleh class yang mengimplementasikannya. Interface ditandai dengan kata kunci interface.

Pada tahap ini, kelompok kami mengimplementasikan beberapa interface untuk sistem pendaftaran dan penjadwalan lembaga kursus yang kami buat. Berikut adalah beberapa interface yang kami buat.

- Implementasi Interface AdminRepository

```

package com.example.demospringboot.repository;
import com.example.demospringboot.entity.Admin;
import org.springframework.data.jpa.repository.JpaRepository;
// import org.springframework.stereotype.Repository;
// import org.springframework.data.jpa.repository.Query;
// import java.util.List;

public interface AdminRepository extends JpaRepository<Admin, Integer> {
}

```

Gambar 3.3.23 Interface AdminRepository

- Implementasi Interface EnrollmentRepository

```

@Repository
public interface EnrollmentRepository extends JpaRepository<Enrollment, Integer> {
    // custome native query tidak berdasarkan JPA

    @Query(value = "SELECT * FROM enrollment WHERE user_id = :user_id AND active_status = 1", nativeQuery = true)
    Enrollment findByUserId(int user_id);

    @Query(value = "SELECT * FROM enrollment WHERE active_status = 1", nativeQuery = true)
    List<Enrollment> findAllActive();

    // update dengan tiga anotasi

    @Modifying
    @Transactional
    @Query(value="UPDATE enrollment SET active_status = 0 WHERE user_id = :user_id", nativeQuery = true)
    int updateInactiveByUser(int user_id);

    @Modifying
    @Transactional
    @Query(value="UPDATE enrollment SET active_status = 0 WHERE jadwal_id = :jadwal_id", nativeQuery = true)
    int updateInactiveByJadwal(int jadwal_id);

    @Query(value = "SELECT COUNT(*) FROM enrollment WHERE user_id = :user_id", nativeQuery = true)
    Integer countByUser(int user_id);

    @Query(value="SELECT * FROM enrollment WHERE user_id = :user_id ORDER BY id DESC LIMIT 1", nativeQuery = true)
    Enrollment findLastByUser(int user_id);
}

```

Gambar 3.3.24 Interface EnrollmentRepository

- Implementasi Interface InstructorRepository

```

@Repository
// Interface

public interface InstructorRepository
extends JpaRepository<Instructor, Integer> {
    @Query(value = "SELECT * FROM instructor WHERE active_status = 1", nativeQuery = true)
    List<Instructor> findAllActive();
}

```

Gambar 3.3.24 Interface InstructorRepository

- Implementasi Interface JadwalRepository

```
@Repository
public interface JadwalRepository extends JpaRepository<Jadwal, Integer> {
    //tambahan, supaya hanya memilih jadwal yang aktif
    @Query(value = "SELECT * FROM jadwal WHERE active_status = 1 AND grade= :grade", nativeQuery = true)
    public List<Jadwal> findBygrade(String grade);

    @Query(value = "SELECT * FROM jadwal WHERE active_status = 1", nativeQuery = true)
    List<Jadwal> findAllActive();

    @Modifying
    @Transactional
    @Query(value="UPDATE jadwal SET active_status = 0 WHERE id = :jadwal_id", nativeQuery = true)
    int updateInactiveById(int jadwal_id);

}
```

Gambar 3.3.25 Interface JadwalRepository

- Implementasi Interface MemberRepository

```
public interface MemberRepository extends JpaRepository<Member, Integer> {
    @Query(value = "SELECT * FROM user WHERE username = :username", nativeQuery = true)
    Member findByUsername(String username);

    @Modifying
    @Transactional
    @Query(value="UPDATE user SET grade = :grade WHERE id = :user_id", nativeQuery = true)
    int updateMemberGrade(int user_id, String grade);
}
```

Gambar 3.3.26 Interface MemberRepository

- Implementasi Interface UserRepository

```
@Repository
// Interface
|
public interface UserRepository
extends JpaRepository<User, Integer> {
    @Query(value = "SELECT * FROM user WHERE username = :username", nativeQuery = true)
    User findByUsername(String username);

    @Modifying
    @Transactional
    @Query(value="UPDATE user SET grade = :grade WHERE id = :user_id", nativeQuery = true)
    int updateUserGrade(int user_id, String grade);
}
```

Gambar 3.3.27 Interface UserRepository

- Implementasi Interface BaseService

```
public interface BaseService<T, ID> {

    // karena hampir semua service yang dibuat terdapat findAll, find by id, delete, save
    // maka dibuat interface yang bisa mewakili fungsi2 tersebut, nantinya tiap class akan override
    // getAll -> findAll
    // add -> save
    // delete -> delete
    // getById -> findById

    List<T> findAll();

    Optional<T> findById(ID id);

    T save(T entity);

    void deleteById(ID id);
}
```

Gambar 3.3.28 Interface BaseService

BAB IV

PENGUJIAN PROGRAM APLIKASI

4.1. Skenario Pengujian

Pada tahap ini, kelompok kami mencoba untuk menguji fungsionalitas, performa dan pengalaman pengguna pada website penjadwalan dan pendaftaran lembaga kursus yang kami buat. Fokus utama kami pada tahap ini adalah memastikan setiap fitur berjalan sesuai dengan seharusnya.

Website kami memiliki beberapa fitur seperti Signup, Login, tes, *reset password*, info *user*, kelola *instructor*, kelola jadwal, dan kelola *enrollment*. Maka dari setiap fitur-fitur ini kami mencoba menguji beberapa skenario yang terjadi untuk memastikan bahwa sistem yang kami buat dapat menangani berbagai kondisi.

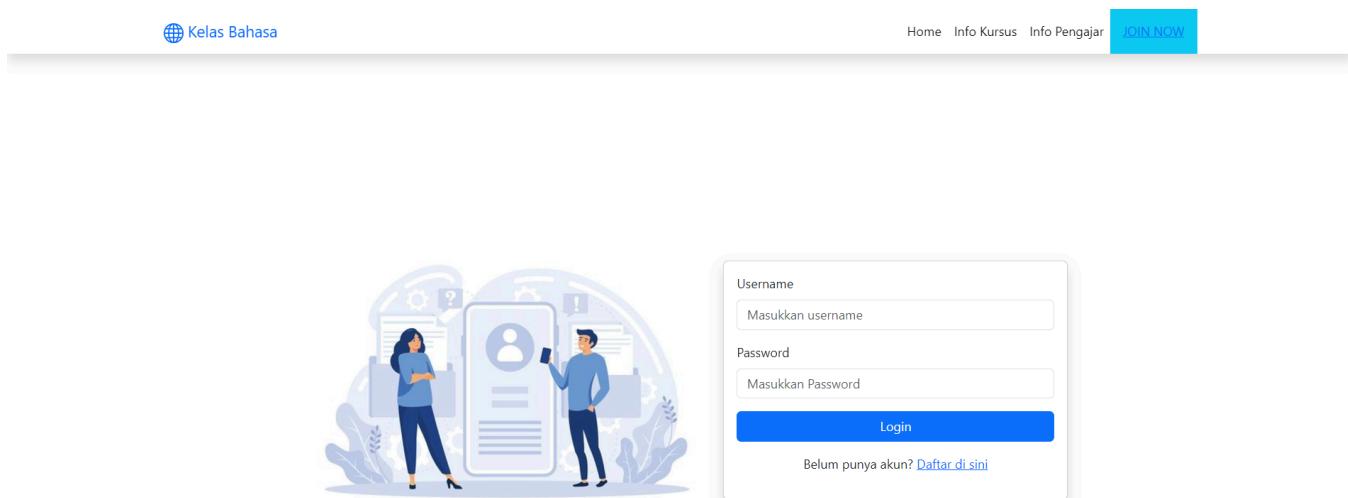
No	Jenis Pengujian	Skenario
1	Pengujian SignUp	Skenario SignUp
2	Pengujian Login	Skenario Login
3	Pengujian Tes	Skenario tes
4	Pengujian Reset Password	Skenario Reset Password
5	Pengujian Login Admin	Skenario Login Admin
6	Pengujian Melihat Informasi User	Skenario Info User
7	Pengujian Kelola Instructor	Skenario Kelola Instructor
8	Pengujian Tambah instructor	Skenario Tambah Instructor
9	Pengujian tambah jadwal	Skenario tambah Jadwal
10	Pengujian menghapus jadwal	Skenario menghapus jadwal
11	Pengujian kelola enrollment	Skenario kelola enrollment

4.2. Pengujian Skenario SignUp

Ini adalah pengujian dari Skenario Signup yang kelompok kami buat. Signup ini dapat digunakan untuk *member* yang ingin mendaftar di lembaga kursus. *Member* dapat melakukan pendaftaran secara online dengan mengisi data diri mereka pada form yang telah kami buat. Setelah itu, maka data-data member tersebut akan masuk ke dalam *database* sistem kami.

Pertama-tama member dapat masuk ke dalam menu *home* website kami. Lalu, klik *join now* maka akan muncul tampilan seperti gambar 4.2.1. Setelah itu pilih link dengan tulisan daftar disini maka akan muncul tampilan menu signup kami seperti gambar 4.2.2. Berikut adalah skenario yang akan terjadi pada fitur signup yang kami buat.

- Klik Join Now menuju menu login



Gambar 4.2.1 menu login

- Klik daftar disini menuju menu signup

Kelas Bahasa

Home Info Kursus Info Pengajar JOIN NOW

Form Pendaftaran

Firstname
Masukkan Firstname

Lastname
Masukkan Lastname

Tanggal Lahir
dd/mm/yyyy

Username
Masukkan Username

Email
Masukkan Email

Password
Masukkan Password

Saya menyetujui syarat dan ketentuan

Daftar

Gambar 4.2.2 menu SignUp

- isi data diri dengan lengkap

Kelas Bahasa

Home Info Kursus Info Pengajar JOIN NOW

Form Pendaftaran

Firstname
michael

Lastname
emmanuel

Tanggal Lahir
16/10/2005

Username
michael

Email
me402709@gmail.com

Password
.....

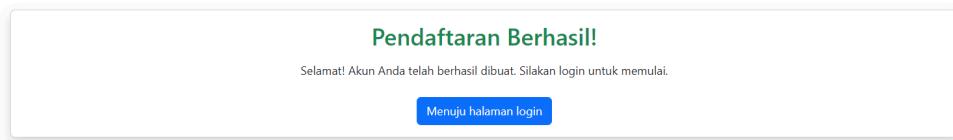
Saya menyetujui syarat dan ketentuan

Daftar

Show desktop

Gambar 4.2.3 pengisian menu SignUp

- tampilan apabila berhasil daftar



Gambar 4.2.4 tampilan berhasil signup

- tampilan apabila form tidak diisi lengkap

A screenshot of a web page showing an incomplete sign-up form. On the left, there is a cartoon illustration of a person making an 'OK' hand gesture. The main area shows a sign-up form titled "Form Pendaftaran". The fields filled are: Firstname (michael), Lastname (emmanuel), Tanggal Lahir (16/10/2005), Username (michael), Email (meh402709@gmail.com). The Password field is empty, indicated by a placeholder "Masukkan Password" and a validation message "Please fill out this field." below it. A checkbox for accepting terms and conditions is also empty. A blue "Daftar" button is at the bottom right of the form.

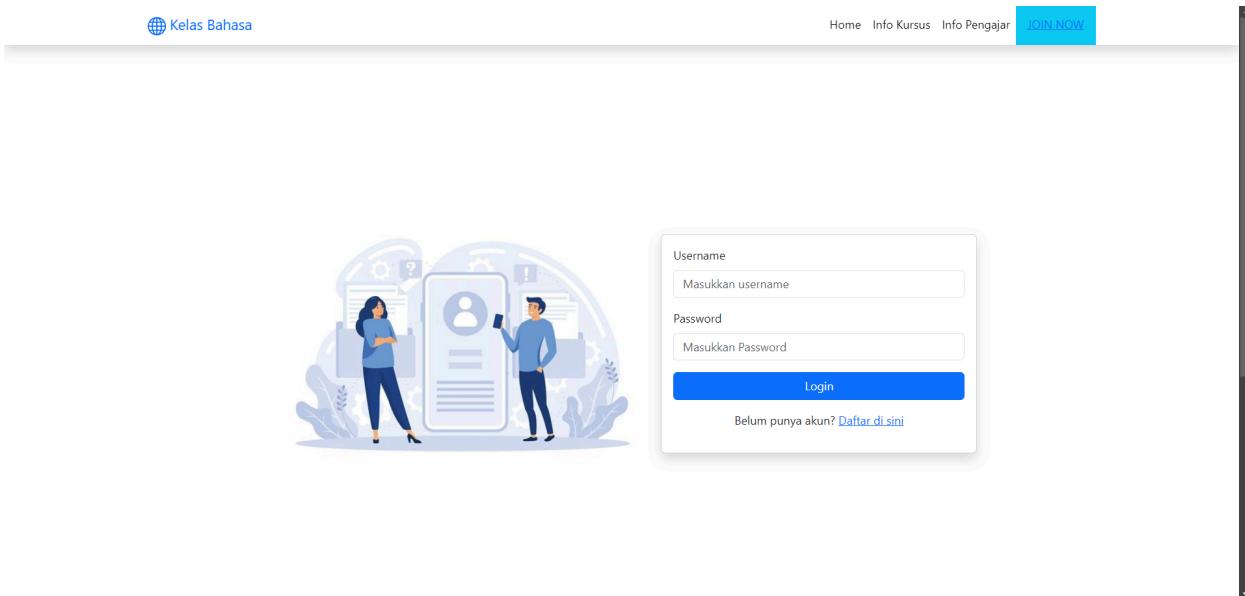
Gambar 4.2.5 tampilan pengisian form signup tidak lengkap

4.3. Pengujian Skenario Login

Ini adalah pengujian dari Skenario Login yang kelompok kami buat. Login ini dapat digunakan untuk member yang ingin mengikuti pelajaran dari kursus bahasa inggris.

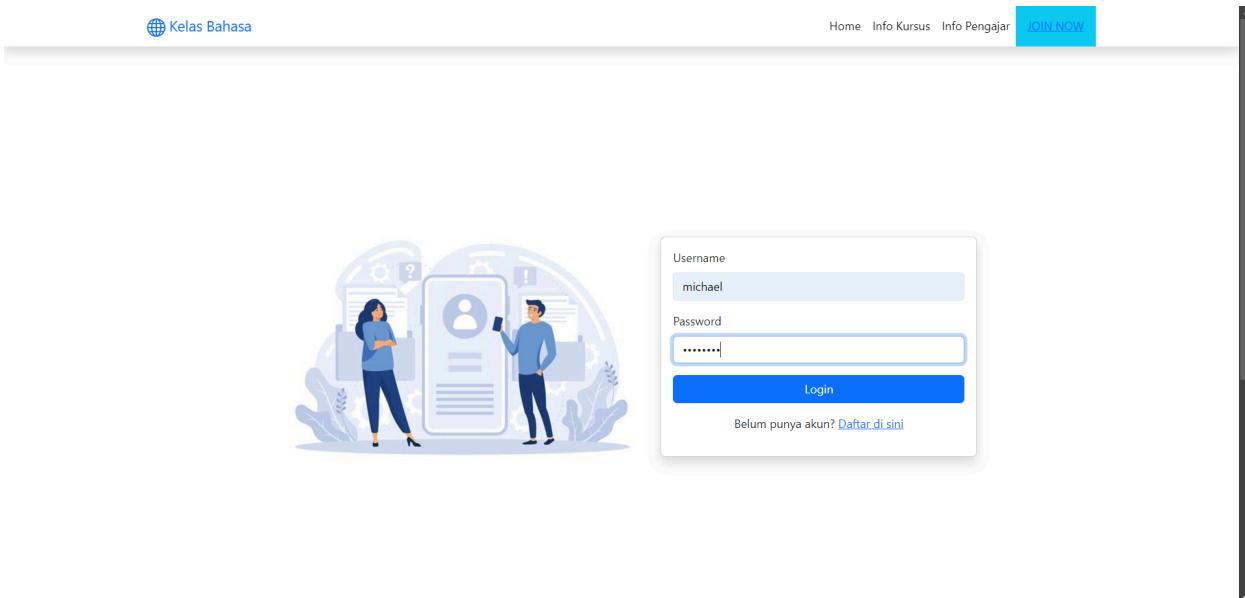
Pertama-tama member dapat klik *Join Now* dan mereka akan masuk ke dalam menu login. *Member* dapat memasukan *username* dan *password* yang mereka input sewaktu signup dan sistem kami akan melakukan validasi dengan database yang ada. Setelah itu, maka member dapat melihat dashboard mereka yang berisi profil mereka, materi kursus, ubah *password*, dan logout. Berikut adalah beberapa skenario pengujian dari fitur login yang kami buat.

- Masuk ke menu Login



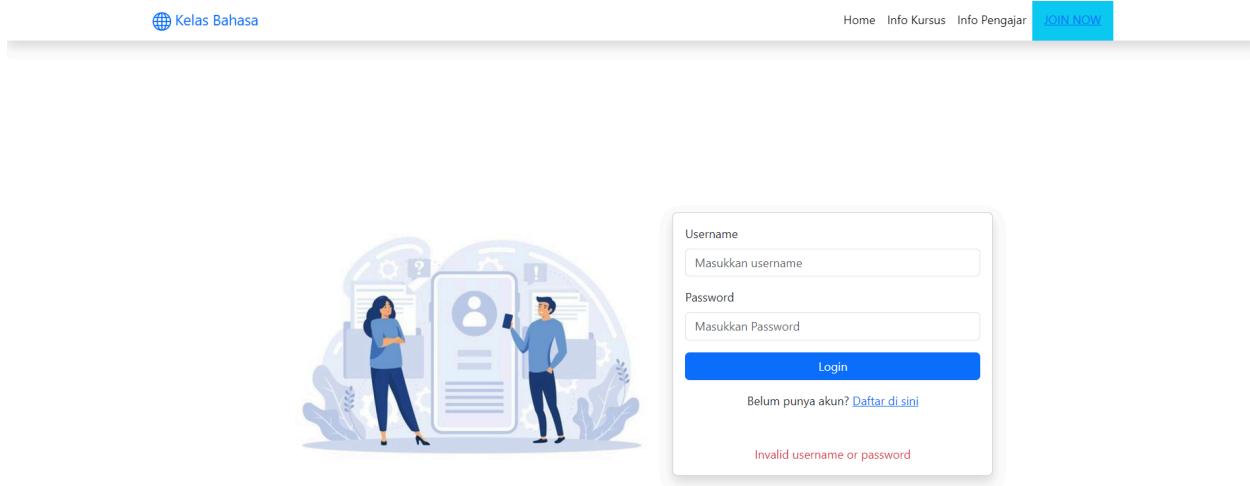
Gambar 4.3.1 Menu Login

- Masukan *username* dan *password* yang sudah didaftarkan



Gambar 4.3.2 Pengisian Menu Login

- Tampilan apabila username/password salah



Gambar 4.3.3 Tampilan username atau password salah

- Tampilan apabila login berhasil



Gambar 4.3.4 Dashboard User

4.4. Pengujian Skenario Tes

Ini adalah pengujian dari Skenario tes yang kelompok kami buat. Tes ini dapat digunakan untuk menguji kemampuan bahasa inggris member dan menentukan jadwal yang dapat diikuti oleh member.

Pertama-tama member dapat klik Mulai Sekarang pada *dashboard member*. Setelah itu, akan ada soal-soal bahasa inggris yang harus diisi oleh *member*. Lalu, *Member* akan mendapatkan hasil tes nya segera dan mendapatkan pilihan jadwal sesuai dengan kemampuan mereka. Berikut adalah pengujian skenario tes website kami.

- Pilih menu “Mulai Sekarang”
-



Gambar 4.4.1 Dashboard Member

- Kerjakan Hasil Tes lalu *submit*

Tes Bahasa Inggris

Pilih jawaban yang benar untuk setiap pertanyaan di bawah ini.

1. What is the correct past tense of 'go'?

- A) Goed
- B) Went
- C) Going

2. Which one is the synonym of 'happy'?

- A) Sad
- B) Joyful
- C) Angry

3. What is the opposite of 'fast'?

- A) Slow
- B) Quick
- C) Swift

4. Choose the correct sentence:

- A) She don't like apples
- B) She doesn't like apples.
- C) She doesn't likes apples.

5. I ___ to the park every weekend.

- A) goes
- B) Go
- C) gone

Submit

Gambar 4.4.2 Soal Tes

- Hasil dari pengeraan soal

Hasil Tes Bahasa Inggris Anda

Berikut adalah hasil tes Bahasa Inggris yang telah Anda selesaikan:

Skor Anda: 100/100

Grade: Lanjutan

Berdasarkan skor Anda, kami merekomendasikan grade kursus **Lanjutan** untuk melanjutkan pengembangan keterampilan Bahasa Inggris Anda.

Pilih Jadwal Kursus

Silakan pilih jadwal yang sesuai untuk memulai kursus Bahasa Inggris Anda:

Sabtu
13:00 - 15:00

Pilih Jadwal

Gambar 4.4.3 Hasil dan Pilihan Jadwal

- Tampilan setelah memilih jadwal



Gambar 4.4.4 Tampilan Pendaftaran Berhasil

- Tampilan setelah kembali ke menu utama

Menu Pilihan

The dashboard includes a greeting "Selamat Datang, michael" and a "User Section" sidebar with links for "Lihat Profile", "Lihat Materi Kursus", "Ubah Password", and "Logout". The main area displays "Informasi Kursus yang diikuti" with details: Grade: Lanjut, Hari: Sabtu, Waktu: 13:00 - 15:00, Instruktur: Dr. John Doe, M.Ed. Action buttons include "Ulangi Assessment Tes", "Ubah Jadwal", and "Berhenti Kursus".

Gambar 4.4.5 Dashboard Member

4.5. Pengujian Skenario Reset Password

Ini adalah pengujian dari Skenario *reset password* yang kelompok kami buat. *Reset password* ini dapat digunakan oleh *member* untuk mengubah *password* akun mereka sendiri.

Pertama-tama member dapat klik Ubah *Password* pada *dashboard member*. Lalu, memasukan *password* baru mereka dan konfirmasi *password* baru mereka. Lalu, setelah mereka login kembali maka *password* yang baru sudah dapat digunakan. Berikut adalah pengujian skenario *reset password* website kami.

- Pilih menu Ubah *Password*

A horizontal line with a central blue link labeled "Ubah Password".

Gambar 4.5.1 Link Ubah Password

- Masukan *password* baru dan konfirmasi

Reset Password

Masukkan password baru dan konfirmasi password Anda.

Password Baru

Konfirmasi Password Baru

Reset Password

Gambar 4.5.2 Form *Reset Password*

- Tampilan jika *password* tidak sesuai

Reset Password

Masukkan password baru dan konfirmasi password Anda.

Password Baru

Konfirmasi Password Baru

Password tidak cocok. Silakan coba lagi.

Reset Password

Gambar 4.5.3 Tampilan *password* tidak sesuai

4.6. Pengujian Skenario Login Admin

Ini adalah pengujian dari Skenario Login Admin yang kelompok kami buat. Login Admin ini dapat digunakan oleh admin untuk Kelola member, kelola instructor, kelola jadwal dan kelola *enrollment*.

Pertama-tama admin dapat masuk ke menu login dan masuk sebagai admin. Lalu, admin akan masuk ke *dashboard* admin. Berikut adalah pengujian skenario login admin website kami.

- Masuk Ke login *page* dan masukan *username* dan id admin



Username

Password

Belum punya akun? [Daftar di sini](#)

You have been logged out successfully.

Gambar 4.6.1 Menu Login

- Tampilan pada *page* admin

Menu Pilihan

Selamat Datang, admin

Admin Menu

[Kelola User](#)

[Kelola Instructor](#)

[Kelola Jadwal](#)

[Kelola Keikutsertaan \(enrollment\)](#)

[Logout](#)

Gambar 4.6.2 Dashboard Admin

4.7. Pengujian Skenario Melihat Info User

Ini adalah pengujian dari skenario melihat info *user* yang kelompok kami buat. Info *user* ini dapat digunakan untuk admin yang ingin melihat informasi dari *user*.

Pertama-tama, *admin* dapat klik kelola *user* di *dashboard* admin. Lalu, muncul informasi *user-user* beserta dengan idnya. Id usernya dapat diklik oleh *admin* untuk melihat informasi dari *user* tersebut. Setelah itu, admin dapat melihat informasi *user* tersebut. Berikut adalah skenario melihat info *user* dari *website* kami.

- Pilih menu kelola *user*

Kelola User

Gambar 4.7.1 Kelola User

- Klik id *user* yang ingin dilihat infonya

No.	ID	Username	Firstname	Last Name	Birthdate	e-mail	Role
1	2	user1	First	User	2004-05-05	user1@gmail.com	USER
2	3	user2	Bryan	Johnson	2008-06-12	user3@gmail.com	USER
3	4	user3	Michael	Johnson	2009-12-12	michael.johnson@example.com	USER
4	5	user4	Emily	Davis	2001-09-20	emily.davis@example.com	USER
5	6	user5	Robert	Brown	2007-06-22	robert.brown@example.com	USER
6	7	user6	Sarah	Miller	2002-08-09	sarah.miller@example.com	USER
7	8	user7	David	Wilson	2008-02-17	david.wilson@example.com	USER
8	9	user8	Laura	Taylor	2010-10-19	laura.taylor@example.com	USER
9	10	user9	Daniel	Moore	2008-09-21	daniel.moore@example.com	USER
10	11	user10	Sophia	Anderson	2005-03-15	sophia.anderson@example.com	USER
11	12	michael	michael	emmanuel	2005-10-16	me402709@gmail.com	USER

Gambar 4.7.2 List Member

- Tampilkan Informasi *User* yang dilihat admin

Detail Data

Return

Information	
Username	user1
Grade	Menengah
Email	user1@gmail.com
Tanggal Lahir	05-May-2004

First User
Member

Gambar 4.7.3 Profil Member

4.8. Pengujian Skenario Kelola Instructor

Ini adalah pengujian dari skenario kelola *instructor* yang kelompok kami buat. Kelola *instructor* ini dapat digunakan untuk admin yang ingin mengelola instructor (dapat mengubah nama instructor maupun *active* statusnya)

Pertama-tama, *admin* dapat klik kelola *instructor* di *dashboard* admin. Lalu, muncul daftar *instructors* beserta dengan idnya. Id usernya dapat diklik oleh admin untuk dikelola datanya. Setelah itu, muncul tampilan edit *instructor* untuk diubah nama atau status aktifnya. Berikut adalah skenario kelola *instructor* dari *website* kami.

- Pilih menu kelola *instructor*

[Kelola Instructor](#)

Gambar 4.8.1 Kelola Instructor

- Klik ID *instructor* yang ingin di olah

No.	ID	Active Status	Instructor Name
1	1	Active	Dr. John Doe, M.Ed.
2	2	Active	Jisoo (Kim Ji-soo)
3	3	Active	Jennie (Jennie Kim)
4	4	Active	Rosé (Park Chae-young)
5	5	Active	Lisa (Lalisa Manoban)

Gambar 4.8.2 List Instructor

- Ubah nama atau mengubah status keaktifan

Edit Instructor

Name

Active?

Submit

Gambar 4.8.3 Form Edit Instructor

- apabila mengubah nama

No.	ID	Active Status	Instructor Name
1	1	Active	Dr. John Doe,

Gambar 4.8.4 Contoh Mengubah Nama

- apabila mengubah activity status

No.	ID	Active Status	Instructor Name
1	1	Inactive	Dr. John Doe, M.Ed.

Gambar 4.8.5 Contoh Mengubah Status

4.9. Pengujian Skenario Menambah Instructor

Ini adalah pengujian dari skenario menambah *instructor* yang kelompok kami buat. Skenario ini dapat digunakan untuk admin yang ingin menambahkan data *instructor*.

Pertama-tama, *admin* dapat klik menu *add data*. Lalu, admin dapat memasukkan nama *instructor* beserta dengan aktif statusnya. Setelah itu, admin dapat menyimpan datanya dengan klik *submit*. Berikut adalah skenario kelola *instructor* dari *website* kami.

- Klik menu *Add data*

All Instructors			
No.	ID	Active Status	Instructor Name
1	1	Active	Dr. John Doe, M.Ed.
2	2	Active	Jisoo (Kim Ji-soo)
3	3	Active	Jennie (Jennie Kim)
4	4	Active	Rosé (Park Chae-young)
5	5	Active	Lisa (Lalisa Manoban)

Gambar 4.9.1 *List Instructor*

- Masukan Data yang ingin di input dan status aktif nya,lalu klik *submit*

Add Instructor

Name

Michael Emmanuel

Active?

Submit

Gambar 4.9.2 Form Edit *Instructor*

- Tampilan ketika data berhasil masuk

6	6	Active	Michael Emmanuel
---	-------------------	--------	------------------

Gambar 4.9.3 Tampilan Setelah diedit

4.10. Pengujian Skenario Tambah Jadwal

Ini adalah pengujian dari skenario tambah jadwal yang kelompok kami buat. Skenario ini dapat digunakan untuk admin yang ingin menambahkan jadwal kursus.

Pertama-tama, *admin* dapat klik kelola jadwal di dashboard admin. Lalu, klik menu *add* data. Lalu, admin dapat memasukkan nama *instructor*, *grade*, jam, dan harinya. Setelah itu, admin dapat menyimpan datanya dengan klik *submit*. Berikut adalah skenario tambah jadwal dari *website* kami.

- Pilih menu jadwal

[Kelola Jadwal](#)

Gambar 4.10.1 Kelola Jadwal

- Pilih add data

All Jadwal						
No.	ID	Level	Hari	Waktu	Nama Instruktur	Action
1	3	Pemula	Senin,Selasa	08:00 - 10:00	Jisoo (Kim Ji-soo)	<button>Delete</button>
2	4	Pemula	Rabu,Kamis	08:00 - 10:00	Jisoo (Kim Ji-soo)	<button>Delete</button>
3	5	Pemula	Jumat,Sabtu	08:00 - 10:00	Jennie (Jennie Kim)	<button>Delete</button>
4	1	Menengah	Senin,Selasa	10:00 - 12:00	Rosé (Park Chae-young)	<button>Delete</button>
5	2	Menengah	Rabu,Kamis	10:00 - 12:00	Lisa (Lalisa Manoban)	<button>Delete</button>
6	6	Lanjutan	Sabtu	13:00 - 15:00	Dr. John Doe, M.Ed.	<button>Delete</button>

Gambar 4.10.2 List Jadwal

- Pilih *instructor, grade, jam dan hari*

Add Jadwal

Instructor:

Grade

Jam

Hari

Senin
 Selasa
 Rabu
 Kamis
 Jumat
 Sabtu
 Minggu

Gambar 4.10.3 Form Add Jadwal

- Tampilan apabila berhasil menambahkan jadwal

4	7	Pemula	Jumat	08:00 - 10:00	Dr. John Doe, M.Ed.	Delete
---	-------------------	--------	-------	---------------	---------------------	------------------------

Gambar 4.10.4 Contoh Tambahan Jadwal

4.11. Pengujian Skenario Menghapus Jadwal

Ini adalah pengujian dari skenario menghapus jadwal yang kelompok kami buat. Skenario ini dapat digunakan untuk admin yang ingin menghapus jadwal kursus.

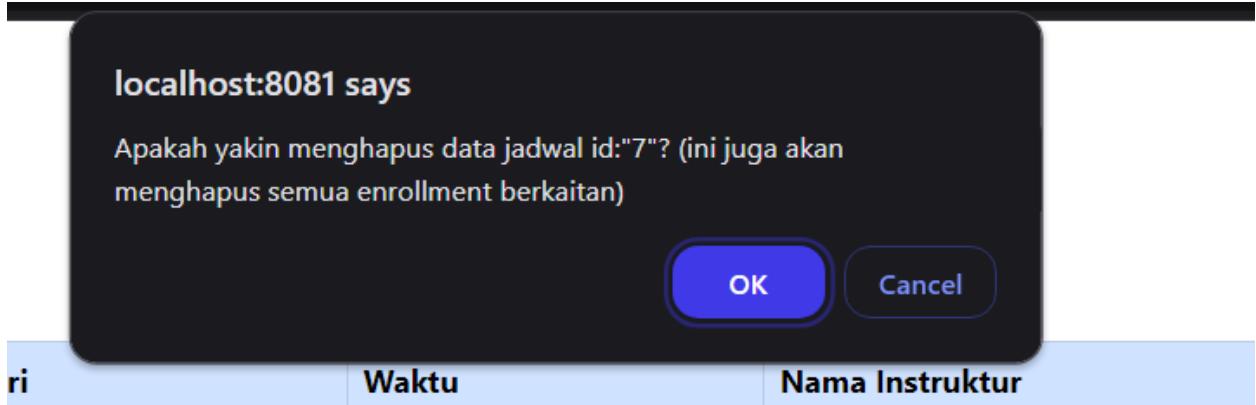
Pertama-tama, *admin* dapat klik kelola jadwal di dashboard admin. Lalu, admin dapat memilih jadwal yang ingin dihapus dan ada muncul tampilan ketika admin klik *delete*. Klik OK untuk menghapus data jadwalnya dan data berhasil dihapus. Berikut adalah skenario hapus jadwal dari *website* kami.

- Pilih jadwal yang ingin dihapus

All Jadwal						
No.	ID	Level	Hari	Waktu	Nama Instruktur	Action
1	3	Pemula	Senin,Selasa	08:00 - 10:00	Jisoo (Kim Ji-soo)	Delete
2	4	Pemula	Rabu,Kamis	08:00 - 10:00	Jisoo (Kim Ji-soo)	Delete
3	5	Pemula	Jumat,Sabtu	08:00 - 10:00	Jennie (Jennie Kim)	Delete
4	7	Pemula	Jumat	08:00 - 10:00	Dr. John Doe, M.Ed.	Delete
5	1	Menengah	Senin,Selasa	10:00 - 12:00	Rosé (Park Chae-young)	Delete
6	2	Menengah	Rabu,Kamis	10:00 - 12:00	Lisa (Lalisa Manoban)	Delete
7	6	Lanjutan	Sabtu	13:00 - 15:00	Dr. John Doe, M.Ed.	Delete

Gambar 4.11.1 List Jadwal

- Tampilan apabila klik delete



Gambar 4.11.2 Peringatan sebelum hapus data

- Tampilan sesudah dihapus

No.	ID	Level	Hari	Waktu	Nama Instruktur	Action
1	3	Pemula	Senin,Selasa	08:00 - 10:00	Jisoo (Kim Ji-soo)	<button>Delete</button>
2	4	Pemula	Rabu,Kamis	08:00 - 10:00	Jisoo (Kim Ji-soo)	<button>Delete</button>
3	5	Pemula	Jumat,Sabtu	08:00 - 10:00	Jennie (Jennie Kim)	<button>Delete</button>
4	1	Menengah	Senin,Selasa	10:00 - 12:00	Rosé (Park Chae-young)	<button>Delete</button>
5	2	Menengah	Rabu,Kamis	10:00 - 12:00	Lisa (Lalisa Manoban)	<button>Delete</button>
6	6	Lanjutan	Sabtu	13:00 - 15:00	Dr. John Doe, M.Ed.	<button>Delete</button>

Gambar 4.11.3 List Jadwal Setelah Dihapus

4.12. Pengujian Skenario Kelola Enrollment

Ini adalah pengujian dari skenario kelola *enrollment* yang kelompok kami buat. Skenario ini dapat digunakan untuk admin yang ingin mengelola *enrollment*.

Pertama-tama, *admin* dapat klik kelola keikutsertaan di *dashboard* admin. Lalu, muncul tampilan menu *enrollment* dan admin dapat klik nama salah satu *instructor*, salah satu *grade*, dan juga dapat klik salah satu nama *member*. Berikut adalah skenario kelola *enrollment* dari *website* kami.

- Pilih kelola keikutsertaan(*enrollment*)

Kelola Keikutsertaan (enrollment)

Gambar 4.12.1 Kelola Enrollment

- Tampilan menu *enrollment*

All Enrollments						
No.	ID	Instructor	Grade	Hari	Jam	Member
1	6	Jisoo (Kim Ji-soo)	Pemula	Senin,Selasa	08:00 - 10:00	Sarah Miller (user6)
2	7	Jisoo (Kim Ji-soo)	Pemula	Rabu,Kamis	08:00 - 10:00	David Wilson (user7)
3	8	Jisoo (Kim Ji-soo)	Pemula	Rabu,Kamis	08:00 - 10:00	Laura Taylor (user8)
4	9	Jennie (Jennie Kim)	Pemula	Jumat,Sabtu	08:00 - 10:00	Daniel Moore (user9)
5	2	Rosé (Park Chae-young)	Menengah	Senin,Selasa	10:00 - 12:00	Bryan Johnson (user2)
6	3	Lisa (Lalisa Manoban)	Menengah	Rabu,Kamis	10:00 - 12:00	Michael Johnson (user3)
7	4	Lisa (Lalisa Manoban)	Menengah	Rabu,Kamis	10:00 - 12:00	Emily Davis (user4)
8	5	Lisa (Lalisa Manoban)	Menengah	Rabu,Kamis	10:00 - 12:00	Robert Brown (user5)
9	11	Lisa (Lalisa Manoban)	Menengah	Rabu,Kamis	10:00 - 12:00	First User (user1)
10	10	Dr. John Doe, M.Ed.	Lanjutan	Sabtu	13:00 - 15:00	Sophia Anderson (user10)
11	13	Dr. John Doe, M.Ed.	Lanjutan	Sabtu	13:00 - 15:00	michael emanuel (michael)

Gambar 4.12.2 List Enrollment

- Tampilan apabila mengklik nama salah satu *instructor*

All Enrollments (filtered)						
No.	ID	Instructor	Grade	Hari	Jam	Member
1	6	Jisoo (Kim Ji-soo)	Pemula	Senin,Selasa	08:00 - 10:00	Sarah Miller (user6)
2	7	Jisoo (Kim Ji-soo)	Pemula	Rabu,Kamis	08:00 - 10:00	David Wilson (user7)
3	8	Jisoo (Kim Ji-soo)	Pemula	Rabu,Kamis	08:00 - 10:00	Laura Taylor (user8)

Gambar 4.12.3 Tampilan Jadwal dengan *Instructor* Jisoo

- Tampilan apabila mengklik salah satu dari *grade*

All Enrollments (filtered)

Back

No.	ID	Instructor	Grade	Hari	Jam	Member
1	6	Jisoo (Kim Ji-soo)	Pemula	Senin,Selasa	08:00 - 10:00	Sarah Miller (user6)
2	7	Jisoo (Kim Ji-soo)	Pemula	Rabu,Kamis	08:00 - 10:00	David Wilson (user7)
3	8	Jisoo (Kim Ji-soo)	Pemula	Rabu,Kamis	08:00 - 10:00	Laura Taylor (user8)
4	9	Jennie (Jennie Kim)	Pemula	Jumat,Sabtu	08:00 - 10:00	Daniel Moore (user9)

Gambar 4.13.4 Tampilan Grade Pemula

- Tampilan apabila mengklik salah satu nama *member*

Detail Data

Return

Information

	Username user6	Grade Pemula
Sarah Miller Member	Email sarah.miller@example.com	Tanggal Lahir 09-Aug-2002

Gambar 4.13.5 Tampilan Profil Member

BAB V

PENGUJIAN PROGRAM APLIKASI

Kelompok kami telah membuat sebuah sistem pendaftaran dan penjadwalan lembaga kursus dengan menggunakan *Java Springboot*. Sistem ini kami rancang untuk mempermudah proses pendaftaran member dan penjadwalan kelas secara terintegrasi. Kami berharap sistem yang kami buat dapat mengelola lembaga kursus menjadi lebih efisien dan terstruktur.

Sistem yang kami buat mempunyai beberapa fitur yang dapat membantu pengguna untuk pendaftaran dan penjadwalan lembaga kursus. Berikut adalah beberapa fitur utama yang kami buat.

- **Pendaftaran online**

Sistem yang kami buat dapat memungkinkan member untuk mendaftar kursus secara online.

- **Manajemen data**

Sistem yang kami buat dapat mengelola data pengguna, *instructor*, jadwal dan *Enrollment*.

- **Penjadwalan yang efisien**

Sistem yang kami buat dapat membantu member untuk menentukan jadwal sesuai dengan jam yang diinginkan oleh member dan hasil tes yang didapat oleh member.

Untuk membuat sistem ini pendekatan yang kami lakukan adalah dengan menggunakan pemrograman berbasis objek atau *Object Based Programming*. Kami juga telah membuat *abstract class* dan *interface* untuk membantu kami dalam membuat sistem penjadwalan dan pendaftaran lembaga kursus. Dalam pembuatan sistem ini, konsep pemrograman berbasis objek yang kami terapkan menggunakan enkapsulasi, polimorfisme dan *inheritance*.

Hasil pengujian kelompok kami menunjukan bahwa sistem berbasis objek ini memberikan struktur yang jelas dan fleksibilitas dalam pengembangan lebih lanjut. Namun, menurut kelompok kami terdapat beberapa kelemahan juga dalam sistem berbasis objek ini seperti kompleksitas kode yang meningkat seiring dengan bertambahnya fitur dan kebutuhan akan optimasi pada pengelolaan data dengan volume yang besar.

Karena keterbatasan waktu dalam pengumpulan maka hanya beberapa fitur saja yang dapat kami buat. Untuk pengembangan lebih lanjut, kelompok kami ada beberapa saran untuk penambahan fitur atau fungsi yang dapat dipertimbangkan. Berikut adalah saran dari kelompok kami.

- **Fitur Notifikasi**

Untuk mengingatkan member mengenai kelas yang akan dimulai sesuai dengan jadwal yang telah dipilih.

- **Fitur Chat**

Untuk membantu member agar dapat berkomunikasi secara online dengan admin kursus.

DAFTAR PUSTAKA

- [1] Microsoft, “Apa itu Java Spring Boot?”, [Online]. Available : <https://azure.microsoft.com/id-id/resources/cloud-computing-dictionary/what-is-java-spring-boot>. [Accessed 25 November 2024].
- [2] Dicoding, “Apa itu UML? Beserta Pengertian dan Contohnya”, [Online]. Available : [https://www.dicoding.com/blog/apa-itu-uml/#:~:text=UML%20\(Unified%20Modelling%20Language\)%20adalah,1.0%20pada%20bulan%20Januari%201997](https://www.dicoding.com/blog/apa-itu-uml/#:~:text=UML%20(Unified%20Modelling%20Language)%20adalah,1.0%20pada%20bulan%20Januari%201997). [Accessed 25 November 2024].
- [3] Spring, “Java Configuration”, [Online]. Available : <https://docs.spring.io/spring-security/reference/servlet/configuration/java.html>. [Accessed 18 November 2024].
- [4] Spring, “Authorize HttpServletRequest”, [Online]. Available : <https://docs.spring.io/spring-security/reference/servlet/authorization/authorize-http-requests.html>. [Accessed 18 November 2024].
- [5] Spring, “Form Login”, [Online]. Available : <https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/form.html>. [Accessed 18 November 2024].

