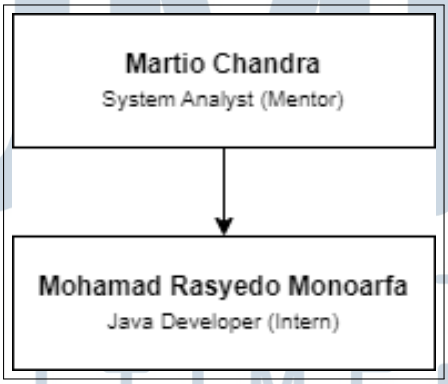
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama menjalani program magang di PT. Madani Intelsysdata, saya berkedudukan sebagai *Fullstack Java Developer Intern* dengan total beban kerja 640 jam yang dimulai pada 25 Agustus 2025. Saya tidak ditempatkan pada divisi formal tertentu sehingga seluruh aktivitas pengembangan dilakukan secara mandiri di bawah pengawasan langsung *System Analyst*, Martio Chandra. Penugasan utama berfokus pada pengembangan sistem pembelajaran internal yang ditujukan bagi siswa PKL (*Praktik Kerja Lapangan*) di lingkungan perusahaan, dengan tujuan membentuk ekosistem belajar yang terstruktur dan terarah sehingga materi, praktik teknis, serta evaluasi kemajuan dapat dikelola secara sistematis dan mudah ditinjau oleh pihak pembimbing. Penempatan ini sekaligus memberi ruang bagi saya untuk merancang alur teknis dan konten pembelajaran berdasarkan kebutuhan riil di lapangan, dengan orientasi pada konsistensi proses dan kemudahan pemanfaatan oleh peserta PKL.

Struktur organisasi yang berlaku bagi posisi saya selama magang digambarkan sebagai berikut pada Gambar 3.1 dan menjadi acuan jalur komunikasi, pengawasan, dan eskalasi masalah teknis serta fungsional dalam pelaksanaan tugas:



Gambar 3.1. Kedudukan di PT. Madani Intelsysdata

Komunikasi keseharian dilakukan melalui *WhatsApp* sebagai kanal yang ringkas dan responsif untuk konfirmasi instruksi, klarifikasi penugasan, dan tindak lanjut dari hasil *weekly update*. Karena bimbingan langsung tidak dilakukan setiap hari, konsultasi bersifat *targeted*—saya menghubungi supervisor ketika menemui

hambatan yang tidak dapat diselesaikan secara mandiri atau ketika keputusan teknis tertentu (misalnya penentuan prioritas fitur pembelajaran) membutuhkan validasi. Umpan balik dari supervisor biasanya diberikan segera setelah demo mingguan atau melalui pesan singkat, memastikan tidak ada jeda panjang antara evaluasi dan perbaikan. Dengan pola ini, saya mengelola ekspektasi secara jelas, mendokumentasikan poin keputusan, dan menutup celah komunikasi yang potensial di antara sesi tatap muka.

Untuk menjaga keterlacakan dan akuntabilitas kerja, saya membuat laporan harian melalui platform magang `prostep.umn.ac.id`. Laporan ini berisi ringkasan tugas yang dikerjakan, kendala yang dihadapi, serta rencana kegiatan berikutnya, sehingga seluruh aktivitas dapat diarsipkan dan ditinjau oleh pihak kampus maupun perusahaan. Di sisi teknis, pengelolaan kode dilakukan melalui *repository* pribadi di *GitHub*, mencakup pengaturan *version control*, pencatatan *commit* yang deskriptif, dan penyimpanan artefak pengembangan. Walau sistem belum memasuki tahap integrasi produksi dan masih bekerja di lingkungan lokal, disiplin pengarsipan ini penting untuk memudahkan audit progres, reproduksi perubahan, dan penyusunan dokumentasi teknis ketika nantinya masuk tahap integrasi yang lebih formal.

Karena proyek dikerjakan secara *solo*, saya memikul seluruh tanggung jawab mulai dari perencanaan alur fitur, eksekusi pengembangan, pengujian fungsional di lingkungan lokal, hingga penyusunan materi demo. Keterbatasan komunikasi dan minimnya pendampingan langsung menjadi kendala utama, terutama ketika dibutuhkan keputusan arsitektural secara cepat atau saat menemui hambatan konseptual yang memerlukan sudut pandang kedua. Dampak yang paling terasa adalah potensi tertundanya klarifikasi teknis hingga sesi *weekly update* berikutnya. Untuk memitigasi hal tersebut, saya menerapkan strategi kerja yang menekankan fokus dan kemandirian: menyiapkan daftar pertanyaan prioritas untuk dibahas di pertemuan mingguan, menyusun *backlog* fitur dengan urutan yang meminimalkan ketergantungan, serta mendokumentasikan alternatif solusi teknis yang siap diputuskan bersama supervisor. Pendekatan ini menekan risiko *bottleneck* dan memastikan setiap pertemuan mingguan menghasilkan keputusan yang *actionable*.

Format *weekly progress update* memiliki peran sentral dalam menjaga arah dan ritme proyek. Pada setiap sesi, saya mempresentasikan keluaran konkret berupa tampilan fitur yang sudah berjalan di *localhost*, memperlihatkan alur interaksi, dan menjelaskan keputusan desain yang diambil sejak pertemuan terakhir. Supervisor kemudian memberikan evaluasi yang terfokus pada kegunaan fitur untuk konteks

pembelajaran PKL, keselarasan dengan tujuan sistem, serta prioritas perbaikan yang perlu ditangani dalam satu minggu ke depan. Revisi biasanya berorientasi pada kejelasan alur pembelajaran, konsistensi tampilan, dan pengurangan friksi bagi pengguna akhir (siswa PKL). Setelah umpan balik diterima, saya menyusun rencana kerja pekanan yang spesifik, dengan target yang dapat diukur agar progres mudah dilacak pada sesi berikutnya.

Dalam keseharian, saya menggabungkan kedisiplinan administratif dan teknis. Secara administratif, seluruh pekerjaan dicatat pada laporan harian `prostep.umn.ac.id`, memastikan dokumentasi yang rapi dan akurat. Secara teknis, *repository GitHub* digunakan untuk mengatur versi dan memelihara riwayat perubahan. Meskipun integrasi dengan server produksi belum dilakukan, kedisiplinan *commit* (misalnya penamaan *commit* yang menjelaskan perubahan, pemisahan perubahan besar menjadi unit yang lebih kecil) memudahkan proses peninjauan dan mendorong praktik kerja profesional yang akan relevan saat sistem beranjak dari *localhost* menuju integrasi yang lebih luas. Keteraturan ini juga memudahkan saya menyusun bahan demo yang representatif dan siap dievaluasi.

Dari perspektif kedudukan dan koordinasi, saya bekerja tanpa struktur tim formal namun memiliki garis koordinasi yang jelas dengan *System Analyst*. Ketiadaan tim pendukung menghadirkan tantangan komunikasi, tetapi juga memberi ruang fokus yang dalam terhadap pekerjaan, memperkuat kemandirian dalam merencanakan dan mengeksekusi fitur, serta meningkatkan kecepatan iterasi di tingkat individu. Siklus *weekly update* berfungsi sebagai penyangga yang memastikan arah tetap pada tujuan pembelajaran PKL, sedangkan komunikasi *WhatsApp* menjadi penghubung antarsesi untuk memastikan keputusan minor tetap bergerak. Dengan kombinasi dokumentasi harian, pengarsipan kode yang tertib, dan evaluasi berkala, pelaksanaan magang berlangsung dalam struktur yang jelas, terukur, dan mendukung tercapainya sasaran sistem pembelajaran internal yang diharapkan perusahaan.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang, saya mendapatkan tanggung jawab utama dalam pengembangan proyek MID E-Learning, yaitu sebuah sistem pembelajaran internal yang dikembangkan oleh PT. Madani Intelsysdata untuk mendukung proses pembelajaran peserta PKL tingkat SMK. Sistem ini dirancang untuk menciptakan pembelajaran yang lebih teratur, terkoordinasi, dan mudah

diawasi, sehingga aktivitas peserta dapat dipantau secara efektif oleh mentor maupun supervisor. MID E-Learning berfungsi sebagai platform terintegrasi yang mencakup materi, penugasan, komunikasi, serta evaluasi peserta secara digital dalam ruang lingkup internal perusahaan.

Dalam pelaksanaan tugas, saya mengerjakan beberapa ruang lingkup pekerjaan yang meliputi:

- Perancangan *UI/UX*,
- Pengembangan *front-end*,
- Pengembangan *back-end*,
- Pengelolaan *database*,
- Integrasi sistem,
- *Maintenance* serta perbaikan fitur lama,
- *Testing*, dan dokumentasi.

Pada tahap desain, saya merancang antarmuka pengguna menggunakan *Figma*, dimulai dari pembuatan *wireframe*, penyusunan alur interaksi, hingga finalisasi desain yang modern dan sesuai kebutuhan pengguna internal. Selain itu, saya menggunakan *Plattext* untuk menyusun diagram desain sistem, memastikan arsitektur aplikasi terdefinisi dengan baik sebelum implementasi dimulai.

Tahap pengembangan *front-end* dilakukan menggunakan kombinasi *Spring Boot*, *Thymeleaf*, dan *Tailwind CSS*, yang menghasilkan tampilan antarmuka yang dinamis, responsif, serta mudah digunakan oleh berbagai *role* pengguna. Pada tahap *back-end*, saya membangun logika aplikasi menggunakan *Java Spring Boot*, mencakup implementasi autentikasi, manajemen *role*, pengelolaan data *course*, *assignment*, *announcement*, hingga forum internal. Saya juga mengembangkan *API* sebagai penghubung antara *front-end* dan server, sehingga pertukaran data dapat dilakukan secara efisien dan terstruktur.

Fitur-fitur inti yang berhasil dikembangkan meliputi:

- Tampilan antarmuka dinamis berbasis *Tailwind CSS*,
- Sistem multi-*role*: *admin*, *mentor*, dan *user*,

- Fitur *admin*: pembuatan akun, pengelolaan data *user*, manajemen *course*, penambahan dan pengelolaan *assignment*, penilaian tugas, komentar, pengelolaan *announcement*, forum internal, serta analisis performa peserta,
- Fitur *mentor*: seluruh fitur *admin* kecuali pengelolaan data *user*,
- Fitur *user*: mengganti *password*, mengumpulkan tugas, mengedit tugas, dan menghapus tugas.

Pengerjaan proyek dilakukan secara individu tanpa tim pengembang lain, sehingga saya berkoordinasi langsung dengan supervisor melalui pertemuan mingguan serta komunikasi melalui *WhatsApp*. Seluruh *repository* proyek dikelola menggunakan *GitHub* sebagai media *version control* dan *backup*. Bekerja secara mandiri memberikan tantangan tersendiri, terutama dalam hal validasi dan koordinasi teknis, namun juga memberikan ruang berpikir yang lebih luas, meningkatkan fokus, serta mendorong kreativitas dalam penyelesaian masalah.

Melalui pengalaman ini, saya memperoleh pemahaman menyeluruh mengenai proses pengembangan perangkat lunak mulai dari tahap desain hingga implementasi dan *deployment*. Selain itu, proyek ini memberikan kesempatan untuk mengasah kemampuan teknis maupun kemampuan bekerja secara mandiri dalam lingkungan profesional.

Sebagai pelengkap uraian tugas yang telah dijelaskan sebelumnya, pelaksanaan kerja magang juga dirangkum secara sistematis berdasarkan periode waktu pelaksanaan. Ringkasan ini menyajikan informasi mengenai minggu pelaksanaan, proyek yang dikerjakan, serta deskripsi aktivitas yang dilakukan. Penyajian dalam bentuk tabel bertujuan untuk memudahkan pembaca dalam menelusuri tahapan kegiatan magang, mengevaluasi kesesuaian antara rencana dan realisasi kegiatan, serta menunjukkan kontribusi penulis secara objektif dan terukur. Rincian pelaksanaan kerja magang disajikan pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu ke-	Pekerjaan yang Dilakukan
1	*Java Training*: Pengenalan lingkungan kerja, setup perangkat, serta memulai *Java Training* Bab 1 tentang konsep dasar komputer, internet, web, *JVM*, dan penulisan program *Java* pertama.
2	*Java Training*: Mempelajari dasar bahasa *Java* meliputi struktur program, variabel, operator, *control statement*, serta latihan program sederhana.
3	*Java Training*: Pendalaman konsep *Object Oriented Programming (OOP)* mencakup *class*, *object*, *encapsulation*, *inheritance*, *polymorphism*, dan penerapannya dalam studi kasus.
4	*Java Training*: Mempelajari *exception handling*, *array* dan *collection*, *generic*, *file handling*, serta implementasi proyek mini berbasis *console*.
5	*Java Training*: Mempelajari *GUI Java*, multimedia, *multithreading*, *networking*, serta koneksi *database* menggunakan *JDBC*.
6	*Java Training*: Mempelajari pengembangan aplikasi web *Java* (*JSF*, *AJAX*, *Web Services*) serta melakukan *review* keseluruhan materi *Java Training*.
7	MID E-learning: Memulai tahap *pre-development* proyek dengan penyusunan *SRS*, *use case diagram*, *activity diagram*, dan *system overview*.
8	MID E-learning: Penyusunan *class diagram*, *ERD*, desain *database*, serta evaluasi dan revisi seluruh dokumen perancangan sistem.
9	MID E-learning: Perancangan *UI/UX* untuk halaman *login*, *home*, *assignment*, serta desain awal untuk role mentor.
10	MID E-learning: Melanjutkan perancangan *UI/UX* untuk role mentor dan admin serta finalisasi desain antarmuka sistem.

Lanjut pada halaman berikutnya

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan)

Minggu ke-	Pekerjaan yang Dilakukan
11	MID E-learning: Memulai tahap *development* dengan pembuatan *login/register*, autentikasi *multi-role*, dan struktur hak akses pengguna.
12	MID E-learning: Pengembangan *dashboard* dan fitur utama untuk role admin dan mentor, termasuk manajemen user, *course*, dan tugas.
13	MID E-learning: Pengembangan fitur student meliputi akses *course*, materi, pengumpulan tugas, *forum* diskusi, dan notifikasi.
14	MID E-learning: Integrasi seluruh fitur antar role serta pengujian *end-to-end* sistem *e-learning*.
15	MID E-learning: Perbaikan *bug* hasil pengujian, optimasi performa *backend*, peningkatan keamanan sistem, serta penyempurnaan *UI/UX*.
16	MID E-learning: Validasi fungsional seluruh fitur, pengujian keamanan hak akses *multi-role*, serta optimasi *query database* dan performa aplikasi.
17	MID E-learning: *Final testing*, penyusunan dokumentasi teknis dan panduan pengguna, serta finalisasi sistem dan persiapan serah terima proyek.

3.3 Uraian Pelaksanaan Magang

Bagian ini menjelaskan secara rinci tahapan pelaksanaan magang yang saya lakukan selama terlibat dalam pengembangan sistem MID E-learning di PT Madani Intelsysdata. Uraian yang disajikan mencakup proses perancangan hingga implementasi sistem, dengan fokus utama pada pengembangan antarmuka pengguna dan penerapan fungsionalitas sistem sesuai kebutuhan pengguna. Pada subbagian ini akan ditampilkan desain antarmuka pengguna (*User Interface*) untuk dua peran utama dalam sistem, yaitu Admin/Mentor dan User (Student), yang masing-masing dirancang untuk mendukung aktivitas pengelolaan dan proses pembelajaran. Selain itu, subbagian ini juga memuat tampilan hasil

implementasi sistem yang telah dikembangkan, sebagai bentuk realisasi dari desain yang telah dirancang sebelumnya. Dengan penyajian ini, diharapkan pembaca dapat memperoleh gambaran yang utuh mengenai alur kerja, hasil desain, serta implementasi teknis yang dilakukan selama pelaksanaan magang.

3.3.1 Alur Sistem

Alur sistem disajikan untuk memberikan gambaran yang jelas mengenai hubungan antara aktor sistem, yaitu Admin, Mentor, dan Student, dengan fungsi-fungsi utama yang tersedia, mulai dari pengelolaan pengguna, pengelolaan *course* dan *assignment*, hingga aktivitas pembelajaran dan pengumpulan tugas. Pemodelan dilakukan menggunakan beberapa pendekatan, meliputi *use case diagram* untuk menggambarkan ruang lingkup dan kebutuhan fungsional sistem, *class diagram* untuk menunjukkan struktur data dan relasi antar entitas, serta *data flow diagram* untuk memvisualisasikan aliran data antar proses dan penyimpanan data. Selain itu, alur sistem juga dijelaskan secara operasional melalui *flowchart* pada fitur-fitur utama berdasarkan peran pengguna, sehingga setiap tahapan proses dapat dipahami secara runtut, terkontrol, dan mendukung pengembangan sistem *MID E-Learning* yang terstruktur dan terintegrasi.

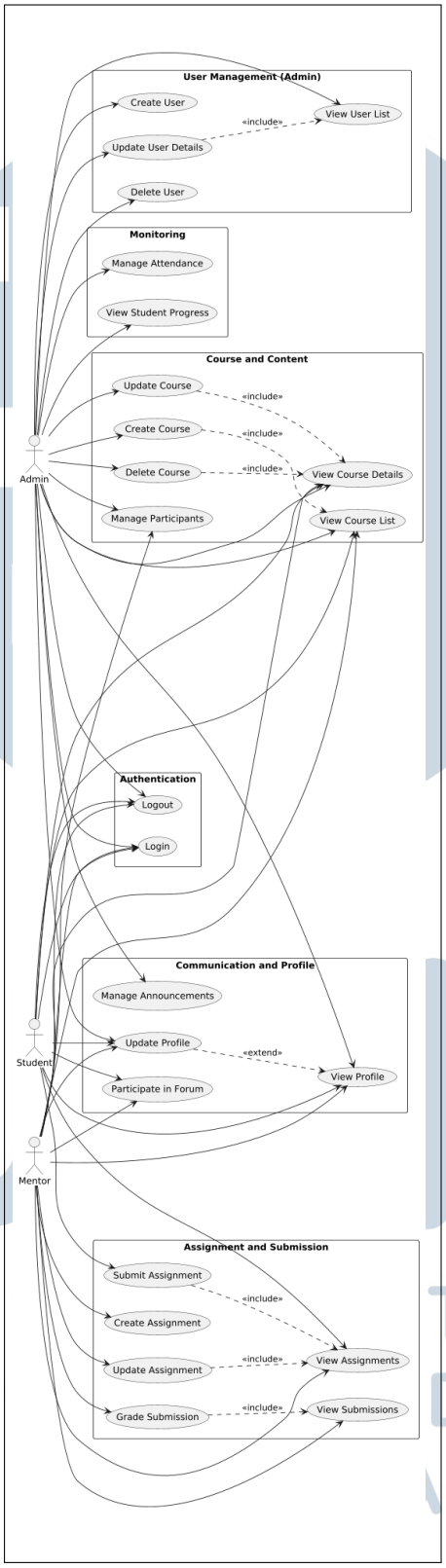
A Diagram Sistem

Diagram sistem pada aplikasi *MID E-Learning* digunakan untuk menggambarkan rancangan sistem dari berbagai sudut pandang, baik dari sisi interaksi pengguna, struktur data, maupun aliran data yang terjadi di dalam sistem. Diagram sistem ini terdiri dari *use case diagram*, *class diagram*, dan *data flow diagram*, yang masing-masing memiliki fungsi dan tujuan perancangan yang berbeda namun saling melengkapi. Penjelasan dari setiap diagram disajikan sebagai berikut.

1. **Use Case Diagram**

   Use case diagram pada sistem *MID E-Learning*, sebagaimana ditunjukkan pada Gambar 3.2, digunakan untuk menggambarkan pola interaksi antara aktor sistem, yaitu Admin, Mentor, dan Student, dengan fungsi-fungsi utama yang tersedia di dalam aplikasi. Diagram ini memberikan gambaran menyeluruh mengenai ruang lingkup sistem dari sudut pandang pengguna, sekaligus menjadi acuan dalam mengidentifikasi

kebutuhan fungsional berdasarkan peran masing-masing aktor. Melalui diagram ini, hubungan antar proses juga dapat dipahami dengan jelas melalui penggunaan relasi *include* dan *extend*, yang menunjukkan ketergantungan maupun perluasan fungsi dalam sistem. Secara visual, diagram ini membantu memastikan bahwa setiap fitur dirancang sesuai dengan tujuan dan alur pembelajaran yang diinginkan.



Gambar 3.2. Use case diagram sistem MID E-Learning

Aktor Admin memiliki peran utama dalam pengelolaan sistem secara keseluruhan. Admin bertanggung jawab terhadap manajemen pengguna, termasuk pengaturan akun Admin, Mentor, dan Student, serta melakukan monitoring terhadap progres pembelajaran siswa. Selain itu, Admin juga memiliki akses untuk mengelola *course* dan *assignment*, sehingga dapat memastikan bahwa struktur pembelajaran, materi, dan tugas tersusun dengan baik dan sesuai dengan kebutuhan sistem.

Aktor Mentor berperan dalam aspek akademik dan interaksi pembelajaran. Mentor memiliki akses untuk membuat *assignment*, melakukan penilaian terhadap tugas yang dikumpulkan oleh Student, serta berpartisipasi dalam forum diskusi sebagai sarana komunikasi dan pendampingan belajar. Melalui fitur forum diskusi, Mentor dapat memberikan arahan, menjawab pertanyaan, serta mendorong interaksi yang lebih aktif antara pengajar dan peserta.

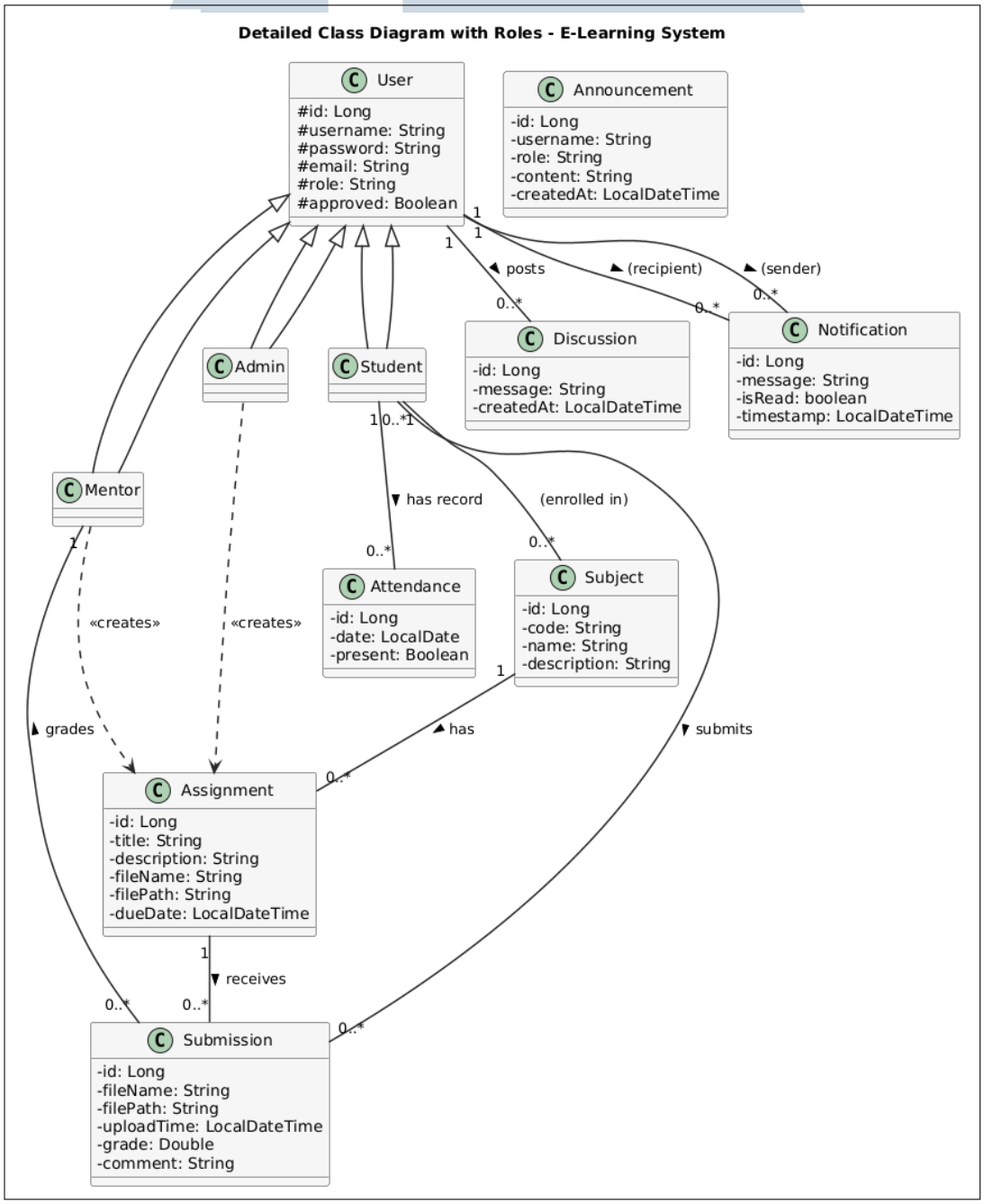
Sementara itu, aktor Student berperan sebagai pengguna yang mengikuti proses pembelajaran. Student dapat mengakses materi pembelajaran yang tersedia dalam *course*, melihat detail *course* yang diikuti, serta mengumpulkan *assignment* sesuai dengan ketentuan yang berlaku. Selain itu, Student juga dapat memantau progres pembelajaran dan hasil penilaian tugas, serta memperbarui data pribadi melalui fitur profil untuk menjaga keakuratan informasi akun.

Relasi *include*, seperti use case *View Course Details* yang menjadi bagian dari *Create Course*, menunjukkan bahwa proses tersebut merupakan langkah yang selalu terlibat dan tidak dapat dipisahkan dari proses utama. Sementara itu, relasi *extend*, seperti *Update Profile* yang memperluas use case *View Profile*, menggambarkan proses tambahan yang bersifat opsional dan hanya dijalankan ketika diperlukan oleh pengguna. Penggunaan relasi ini menegaskan bahwa sistem *MID E-Learning* dirancang secara modular dan fleksibel, sehingga setiap fitur dapat dikembangkan atau digunakan tanpa mengganggu alur utama pembelajaran.

2. **Class Diagram**

Class diagram pada sistem *MID E-Learning*, sebagaimana ditunjukkan pada Gambar 3.3, digunakan untuk menggambarkan struktur data serta hubungan antar kelas yang membentuk sistem

secara keseluruhan. Diagram ini berperan sebagai representasi statis dari sistem yang menunjukkan atribut, metode, serta relasi antar entitas. Dengan adanya *class diagram*, perancangan basis data dan implementasi pemrograman berorientasi objek dapat dilakukan secara lebih terstruktur dan konsisten, sehingga meminimalkan kesalahan dalam pengelolaan data dan pengembangan sistem.



Gambar 3.3. Class diagram struktur data sistem MID E-Learning

Kelas *User* berperan sebagai *superclass* yang menjadi dasar bagi kelas *Admin*, *Student*, dan *Mentor*. Kelas *User* menyimpan atribut umum seperti *id*, *username*, *email*, dan *role*, yang digunakan dalam proses autentikasi serta identifikasi hak akses pengguna. Penerapan konsep pewarisan (*inheritance*) pada struktur ini bertujuan untuk menghindari redundansi data dan menjaga konsistensi atribut yang dimiliki oleh setiap jenis pengguna.

Setiap turunan dari kelas *User* memiliki tanggung jawab dan relasi yang berbeda sesuai dengan perannya masing-masing. Kelas *Student* memiliki relasi dengan kelas *Attendance*, *Subject*, dan *Submission*, yang merepresentasikan aktivitas kehadiran, keikutsertaan dalam mata pelajaran, serta proses pengumpulan tugas. Relasi ini menunjukkan bahwa data akademik peserta dikelola secara terstruktur dan dapat ditelusuri berdasarkan setiap aktivitas pembelajaran yang dilakukan.

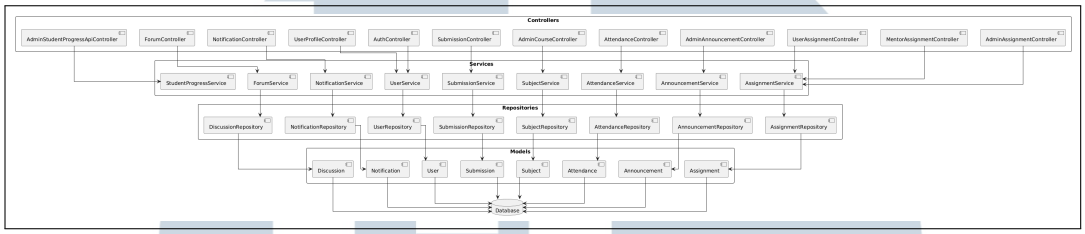
Kelas *Mentor* dan *Admin* memiliki relasi dengan kelas *Assignment*, yang menandakan bahwa kedua peran tersebut memiliki wewenang dalam pembuatan dan pengelolaan tugas. *Mentor* berfokus pada aspek akademik, seperti penyusunan dan penilaian tugas, sedangkan *Admin* memiliki peran tambahan dalam pengelolaan sistem secara menyeluruh. Kelas *Submission* menyimpan atribut penting seperti *grade* dan *comment*, yang digunakan sebagai media evaluasi dan umpan balik terhadap hasil pekerjaan *Student*.

Selain itu, setiap objek *User* dapat membuat *Announcement*, menerima *Notification*, serta berinteraksi melalui kelas *Discussion*. Relasi ini menunjukkan bahwa sistem menyediakan mekanisme komunikasi dan penyampaian informasi yang mendukung proses pembelajaran secara efektif. Secara keseluruhan, relasi antar kelas yang ditunjukkan melalui asosiasi dan pewarisan memperjelas alur data serta tanggung jawab masing-masing entitas, sekaligus mendukung pengembangan sistem *MID E-Learning* yang modular, terstruktur, dan mudah dikembangkan di masa mendatang.

3. **Data Flow Diagram**

Data flow diagram (*DFD*) pada sistem *MID E-Learning*, sebagaimana ditunjukkan pada Gambar 3.4, digunakan untuk menjelaskan aliran data antar entitas eksternal, proses internal, dan penyimpanan data yang terdapat dalam sistem. Diagram ini memberikan gambaran makro mengenai bagaimana informasi diterima, diproses, disimpan, dan didistribusikan kembali kepada

pengguna sesuai dengan peran dan kebutuhan masing-masing. Dengan menggunakan *DFD*, pengembang dapat memahami keterkaitan antar modul serta memastikan bahwa alur data dalam sistem berjalan secara logis dan konsisten.



Gambar 3.4. Data flow diagram sistem MID E-Learning

Entitas eksternal dalam sistem ini meliputi *Student*, *Mentor*, dan *Admin*, yang masing-masing berinteraksi dengan sistem melalui antarmuka pengguna. *Student* mengirimkan data berupa informasi kehadiran, pengumpulan tugas, serta permintaan akses materi pembelajaran. Data tersebut diterima oleh sistem dan diproses oleh modul-modul terkait, seperti modul *attendance* untuk pencatatan kehadiran, modul *assignment* untuk pengelolaan tugas, serta modul *subject* untuk pengelolaan materi dan *course*.

Data yang telah diproses kemudian disimpan ke dalam *database* sebagai pusat penyimpanan informasi sistem. Penyimpanan ini memungkinkan data akademik dan aktivitas pengguna terdokumentasi dengan baik dan dapat diakses kembali sesuai kebutuhan. *Mentor* dapat mengambil data dari sistem untuk melakukan penilaian tugas, memantau kehadiran, serta mengelola aktivitas pembelajaran. Sementara itu, *Admin* memanfaatkan aliran data tersebut untuk keperluan *monitoring* sistem, pengelolaan *course*, serta pengawasan progres pembelajaran secara keseluruhan.

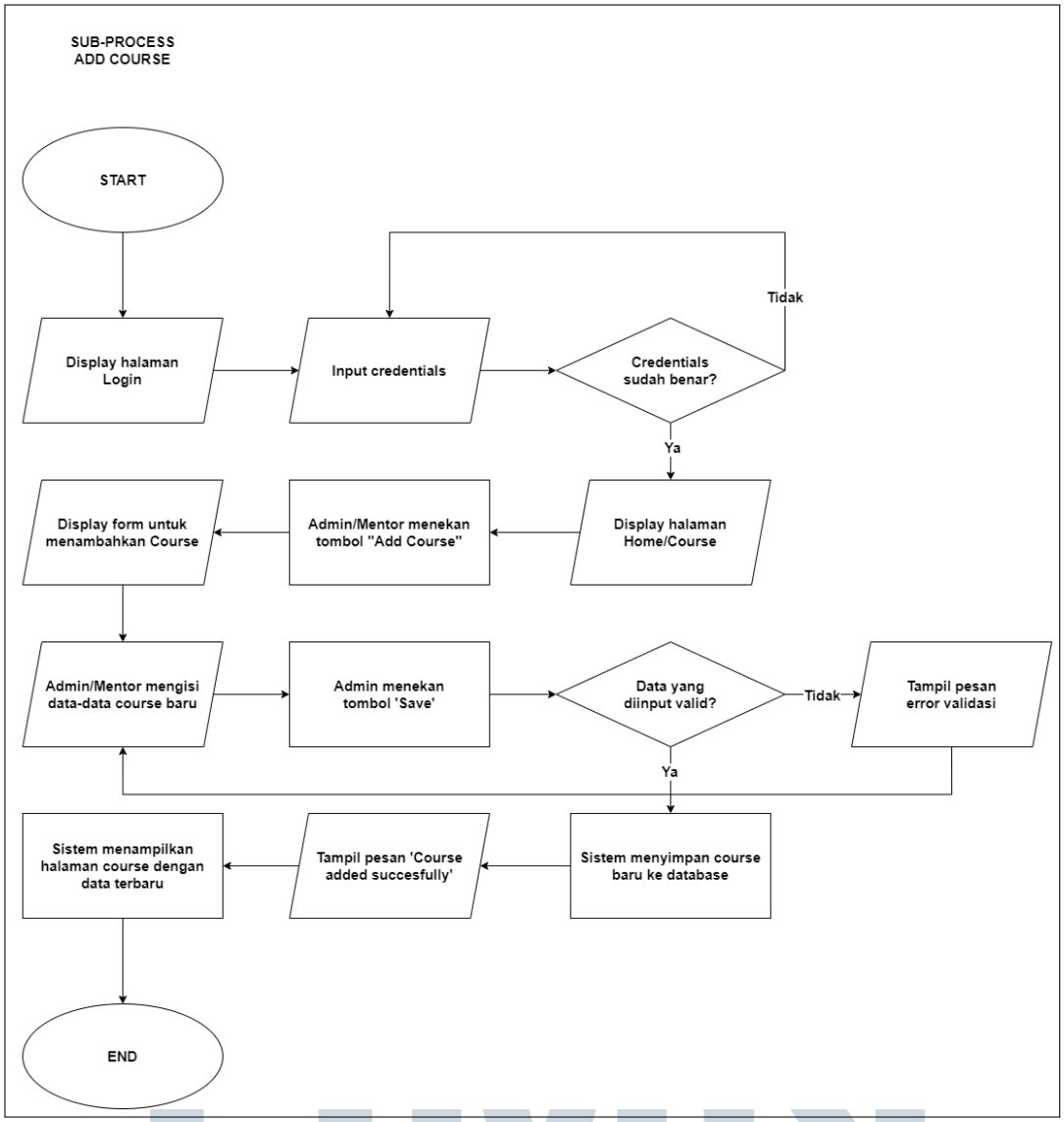
Aliran data dalam diagram ditunjukkan dengan panah yang menghubungkan entitas eksternal dengan proses internal dan *database*. Panah-panah tersebut menggambarkan arah perpindahan data serta jenis informasi yang dipertukarkan, sehingga memperlihatkan keterkaitan antar modul dalam sistem. Dengan struktur aliran data yang jelas, sistem *MID E-Learning* mampu menjaga konsistensi informasi, mengurangi redundansi data, serta mendukung proses pembelajaran dan pengelolaan sistem secara terintegrasi dan efisien.

B Flowchart Sistem

Flowchart sistem pada aplikasi *MID E-Learning* digunakan untuk menggambarkan alur proses utama yang dijalankan oleh setiap aktor dalam sistem, baik Admin maupun Student. Flowchart ini berfungsi untuk memvisualisasikan tahapan interaksi antara pengguna dan sistem secara rinci, mulai dari proses autentikasi, pengelolaan data, hingga penyimpanan informasi ke dalam *database*. Penjelasan setiap flowchart disajikan sebagai berikut.

1. **Admin – Add Course**

   Berdasarkan Gambar 3.5, *flowchart Add Course* menggambarkan alur proses penambahan mata pelajaran (*course*) yang dilakukan oleh *Admin* atau *Mentor* pada sistem *MID E-Learning*. Diagram alir ini bertujuan untuk menjelaskan secara detail interaksi antara pengguna dan sistem dalam mengelola data *course*, mulai dari proses autentikasi pengguna, pengisian formulir, validasi data, hingga penyimpanan data ke dalam *database*. Dengan adanya *flowchart* ini, proses penambahan *course* dapat dipahami secara terstruktur dan meminimalkan kesalahan dalam pengelolaan data.



Gambar 3.5. Flowchart alur sistem admin – *add course*

Berdasarkan *flowchart* tersebut, proses dimulai ketika *Admin* melakukan *login* dengan memasukkan kredensial yang valid berupa *username* dan *password*. Sistem akan melakukan proses autentikasi untuk memastikan bahwa pengguna memiliki hak akses yang sesuai. Apabila proses autentikasi gagal, sistem akan menolak akses dan mengarahkan pengguna kembali ke halaman *login*. Jika autentikasi berhasil, *Admin* akan diarahkan ke halaman utama atau halaman *Home/Course*.

Pada halaman *Home/Course*, *Admin* dapat melihat daftar mata pelajaran yang telah terdaftar dalam sistem. Untuk menambahkan mata

pelajaran baru, *Admin* menekan tombol *Add Course* yang tersedia pada halaman tersebut. Tindakan ini akan memicu sistem untuk menampilkan halaman formulir penambahan *course*. Formulir ini berfungsi sebagai media input data dan biasanya terdiri dari beberapa atribut utama, seperti nama mata pelajaran dan deskripsi mata pelajaran.

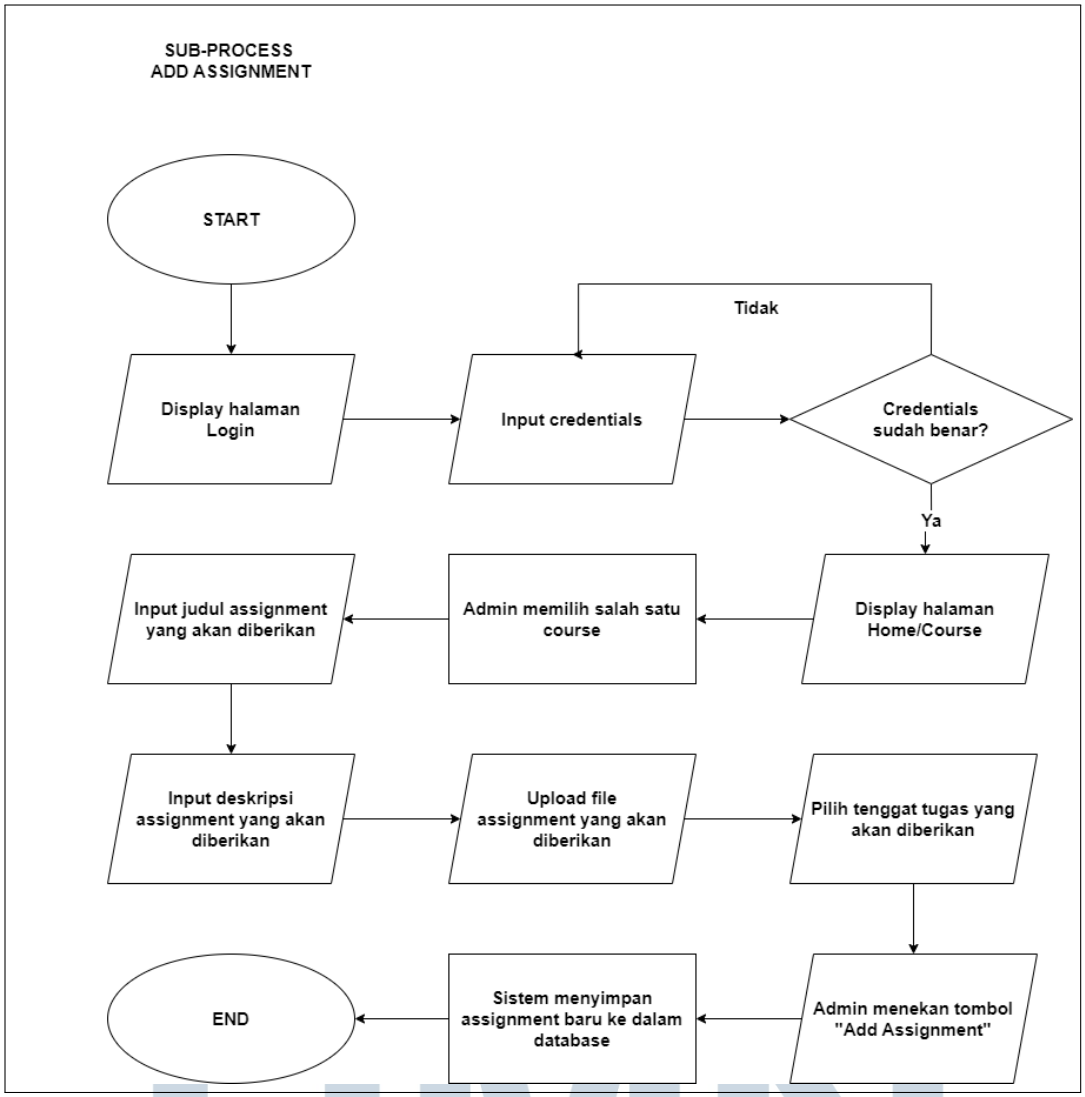
Selanjutnya, *Admin* mengisi seluruh data *course* yang diperlukan pada formulir tersebut. Setelah memastikan data telah diisi dengan benar, *Admin* menekan tombol *Save* untuk mengirimkan data ke sistem. Sistem kemudian melakukan proses validasi data, yang mencakup pemeriksaan kelengkapan data, format input, serta kemungkinan adanya data yang sama atau duplikasi *course* yang sudah terdaftar sebelumnya.

Apabila hasil validasi menunjukkan bahwa data yang dimasukkan tidak valid, sistem akan menampilkan pesan kesalahan (*error message*) sebagai bentuk umpan balik kepada *Admin*, sehingga pengguna dapat melakukan perbaikan data sebelum melanjutkan proses. Sebaliknya, apabila data dinyatakan valid, sistem akan melanjutkan proses dengan menyimpan data *course* baru ke dalam *database*.

Setelah proses penyimpanan data berhasil dilakukan, sistem akan menampilkan pesan keberhasilan (*success message*) kepada *Admin*. Selain itu, sistem secara otomatis akan memperbarui daftar *course* pada halaman *Home/Course*, sehingga mata pelajaran yang baru ditambahkan dapat langsung ditampilkan, diakses, dan dikelola lebih lanjut sesuai dengan kebutuhan pembelajaran dan pengelolaan sistem *MID E-Learning*.

2. **Admin – Add Assignment**

Seperti ditunjukkan pada Gambar 3.6, *flowchart Add Assignment* menggambarkan alur proses penambahan tugas (*assignment*) yang dilakukan oleh *Admin* pada suatu *course* tertentu di dalam sistem *MID E-Learning*. Diagram alir ini digunakan untuk memvisualisasikan tahapan interaksi antara pengguna dan sistem dalam mengelola data tugas, mulai dari proses autentikasi, pemilihan *course*, pengisian data tugas, hingga penyimpanan data ke dalam *database*. Dengan adanya *flowchart* ini, alur penambahan tugas dapat dipahami secara sistematis dan mendukung pengelolaan pembelajaran yang terstruktur.



Gambar 3.6. Flowchart alur sistem admin – *add assignment*

Berdasarkan *flowchart* tersebut, proses dimulai ketika *Admin* melakukan *login* ke dalam sistem dengan memasukkan kredensial yang valid. Sistem melakukan proses autentikasi untuk memastikan bahwa pengguna memiliki hak akses sebagai *Admin*. Jika autentikasi berhasil, *Admin* akan diarahkan ke halaman utama sistem.

Pada halaman utama, *Admin* memilih salah satu *course* yang telah terdaftar dan ingin diberikan tugas. Pemilihan *course* ini penting karena setiap *assignment* yang ditambahkan harus memiliki keterkaitan langsung dengan mata pelajaran tertentu agar dapat diakses oleh peserta yang terdaftar pada *course* tersebut. Setelah *course* dipilih, *Admin* mengakses fitur *Add*

Assignment untuk membuka formulir penambahan tugas.

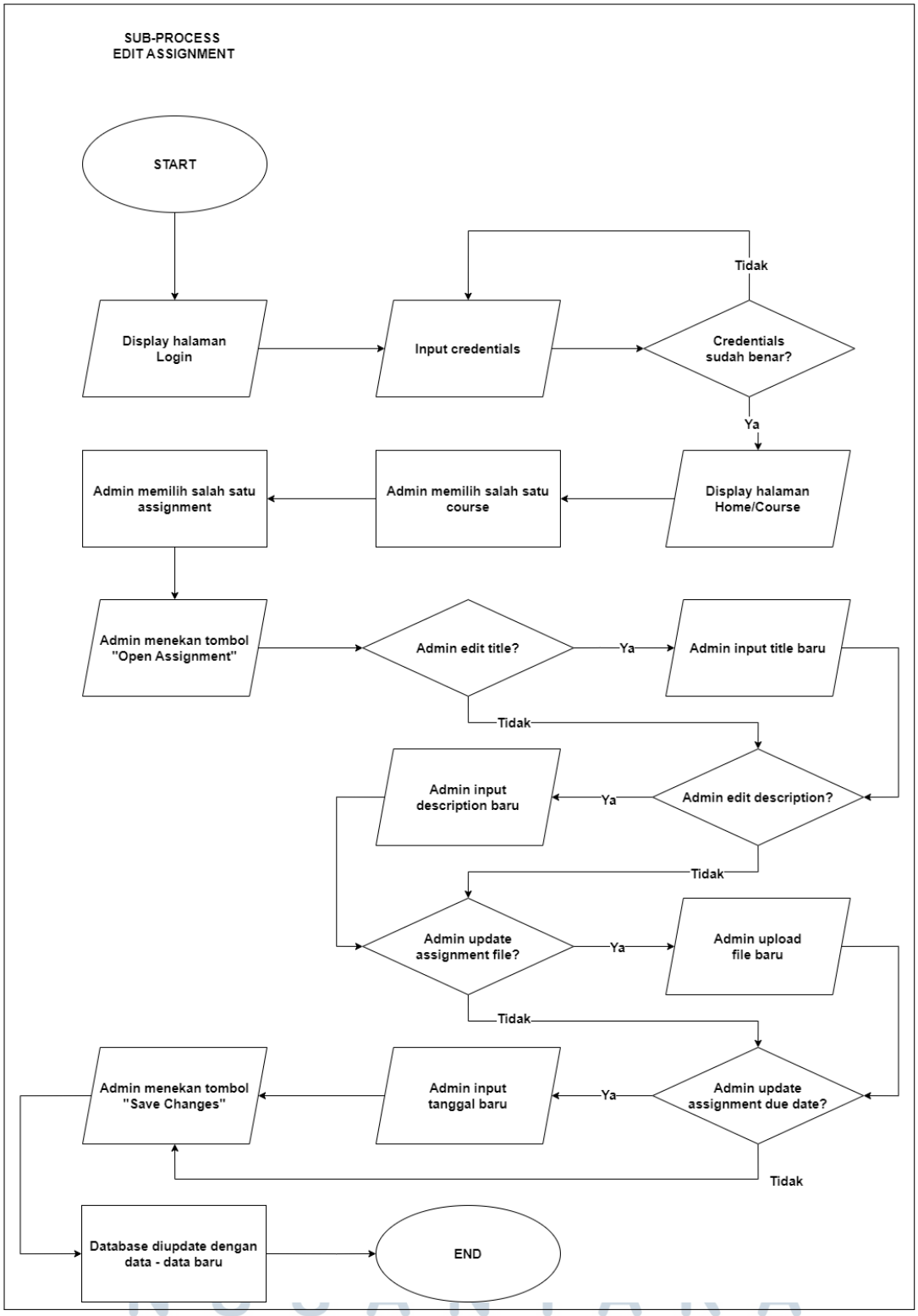
Selanjutnya, *Admin* mengisi data *assignment* pada formulir yang disediakan. Data yang diinput meliputi judul tugas, deskripsi tugas, file pendukung sebagai bahan atau instruksi tambahan, serta tenggat waktu (*deadline*) pengumpulan tugas. Informasi ini berfungsi sebagai pedoman bagi peserta dalam memahami dan menyelesaikan tugas yang diberikan.

Setelah seluruh data *assignment* diisi, *Admin* menekan tombol *Add Assignment* untuk mengirimkan data ke sistem. Sistem kemudian melakukan proses validasi untuk memastikan bahwa seluruh data telah diisi dengan benar dan sesuai dengan ketentuan yang berlaku, seperti kelengkapan data dan format input. Apabila validasi gagal, sistem akan menampilkan pesan kesalahan sebagai umpan balik kepada *Admin* agar dapat melakukan perbaikan data.

Apabila data dinyatakan valid, sistem akan menyimpan data *assignment* ke dalam *database* dan mengaitkannya dengan *course* yang telah dipilih sebelumnya. Setelah proses penyimpanan berhasil, sistem menampilkan pesan keberhasilan dan *assignment* yang baru ditambahkan akan langsung tersedia serta dapat diakses oleh peserta pada *course* tersebut. Dengan demikian, sistem *MID E-Learning* mampu mendukung proses distribusi tugas secara terorganisir, tepat sasaran, dan sesuai dengan alur pembelajaran yang telah dirancang.

3. **Admin – Edit Assignment**

Berdasarkan Gambar 3.7, *flowchart Edit Assignment* menggambarkan alur proses yang dijalankan oleh *Admin* dalam melakukan perubahan terhadap data tugas (*assignment*) yang telah dibuat sebelumnya pada sistem *MID E-Learning*. Diagram alir ini berfungsi untuk memvisualisasikan mekanisme pembaruan data tugas secara terkontrol, sehingga *Admin* dapat menyesuaikan informasi *assignment* tanpa perlu membuat tugas baru, sekaligus menjaga konsistensi data dalam sistem.



Gambar 3.7. Flowchart alur sistem admin – *edit assignment*

Berdasarkan *flowchart* tersebut, proses dimulai ketika *Admin*

melakukan *login* ke dalam sistem dengan memasukkan kredensial yang valid. Sistem melakukan proses autentikasi untuk memastikan bahwa pengguna memiliki hak akses sebagai *Admin*. Setelah berhasil masuk, *Admin* diarahkan ke halaman utama dan selanjutnya mengakses halaman detail *course* yang berisi daftar *assignment* yang telah dibuat.

Pada halaman detail *course*, *Admin* memilih salah satu *assignment* yang ingin diperbarui. Pemilihan ini dilakukan untuk memastikan bahwa perubahan hanya diterapkan pada tugas yang relevan. Setelah *assignment* dipilih, sistem menampilkan formulir *edit assignment* yang berisi data tugas sebelumnya, seperti judul, deskripsi, file pendukung, dan tenggat waktu (*deadline*).

Selanjutnya, *Admin* melakukan pengeditan pada data yang diperlukan, misalnya memperbarui deskripsi tugas, mengganti atau menambahkan file pendukung, maupun menyesuaikan tenggat waktu pengumpulan. Fitur ini memungkinkan *Admin* untuk menyesuaikan kebijakan akademik atau kebutuhan pembelajaran tanpa mengganggu struktur *course* yang telah berjalan.

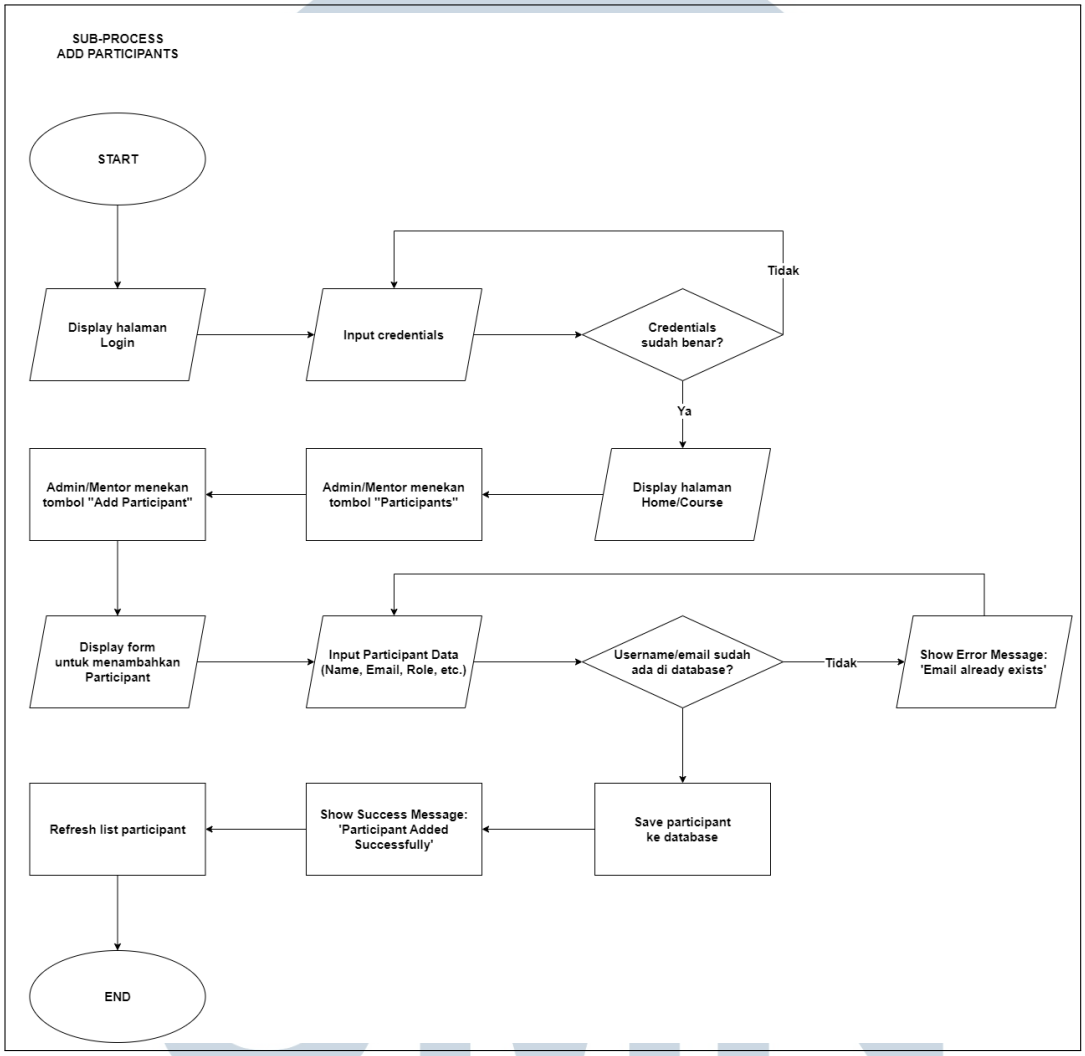
Setelah perubahan selesai dilakukan, *Admin* menekan tombol *Save* untuk mengirimkan data yang telah diperbarui ke sistem. Sistem kemudian melakukan proses validasi untuk memastikan bahwa data yang diinput telah sesuai dengan ketentuan yang berlaku dan tidak melanggar aturan sistem. Apabila validasi gagal, sistem akan menampilkan pesan kesalahan sebagai umpan balik kepada *Admin* agar perbaikan dapat dilakukan.

Apabila data dinyatakan valid, sistem akan menyimpan perubahan *assignment* ke dalam *database* dan memperbarui data tugas yang terkait dengan *course* tersebut. Setelah proses penyimpanan berhasil, sistem menampilkan pesan keberhasilan dan informasi *assignment* yang ditampilkan kepada pengguna merupakan versi terbaru. Dengan demikian, alur *edit assignment* ini mendukung pengelolaan tugas yang fleksibel, akurat, dan terintegrasi dalam sistem *MID E-Learning*.

4. **Admin – Add Participant**

Sebagaimana ditunjukkan pada Gambar 3.8, *flowchart Add Participant* menggambarkan alur proses yang dijalankan oleh *Admin* dalam menambahkan peserta baru ke dalam sistem *MID E-Learning*.

Diagram alir ini digunakan untuk memvisualisasikan tahapan input, validasi, dan penyimpanan data peserta, sehingga proses penambahan pengguna dapat dilakukan secara terkontrol dan sesuai dengan standar sistem.



Gambar 3.8. Flowchart alur sistem admin – *add participant*

Berdasarkan *flowchart* tersebut, proses dimulai ketika *Admin* melakukan *login* ke dalam sistem dengan memasukkan kredensial yang valid. Sistem terlebih dahulu melakukan proses autentikasi untuk memastikan bahwa pengguna memiliki hak akses sebagai *Admin*. Setelah berhasil masuk, *Admin* mengakses menu manajemen peserta yang berisi daftar pengguna yang telah terdaftar dalam sistem.

Pada menu tersebut, *Admin* memilih opsi *Add Participant* untuk membuka formulir penambahan peserta baru. Selanjutnya, *Admin* mengisi

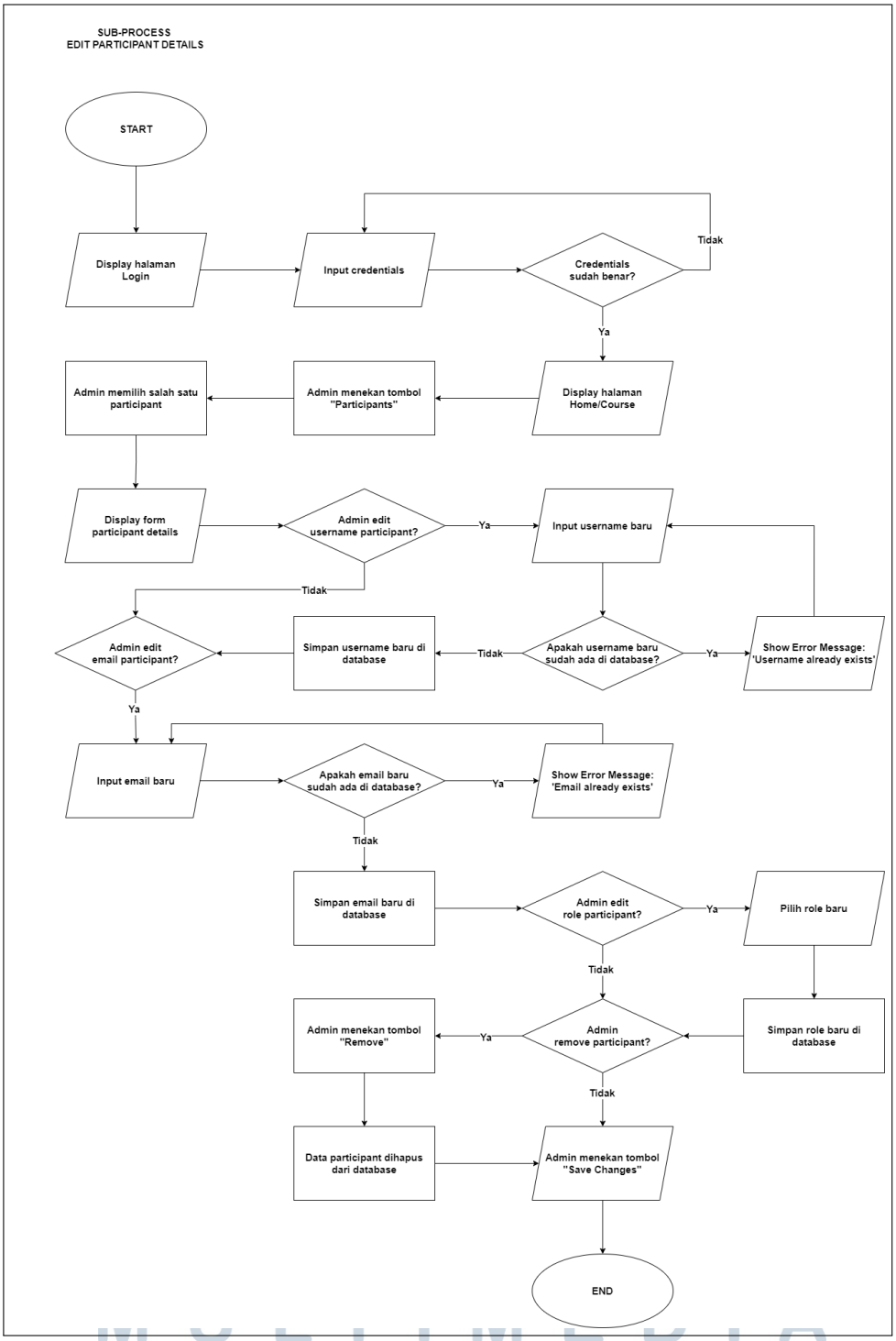
data peserta yang diperlukan, seperti identitas pengguna, informasi akun, serta penentuan peran atau hak akses yang akan diberikan. Pengisian data ini bertujuan agar peserta dapat terintegrasi dengan sistem sesuai dengan perannya dalam proses pembelajaran.

Setelah seluruh data diisi, *Admin* menekan tombol *Save* untuk mengirimkan data ke sistem. Sistem kemudian melakukan proses validasi terhadap data yang diinput, termasuk pemeriksaan kelengkapan dan kesesuaian format data. Apabila terdapat data yang tidak valid atau tidak sesuai dengan ketentuan sistem, sistem akan menampilkan pesan kesalahan sebagai umpan balik kepada *Admin* agar perbaikan dapat dilakukan.

Apabila data dinyatakan valid, sistem akan menyimpan data peserta baru ke dalam *database* dan menambahkan pengguna tersebut ke dalam daftar peserta sistem. Setelah proses penyimpanan berhasil, sistem menampilkan pesan keberhasilan, dan peserta yang telah ditambahkan dapat melakukan *login* serta mengakses platform *MID E-Learning* sesuai dengan hak akses yang telah ditetapkan. Dengan demikian, alur *add participant* ini mendukung pengelolaan pengguna yang terstruktur, aman, dan terintegrasi dalam sistem.

5. **Admin – Edit Participant Details**

Berdasarkan Gambar 3.9, *flowchart Edit Participant Details* menggambarkan alur proses yang dijalankan oleh *Admin* dalam melakukan pembaruan data peserta yang telah terdaftar pada sistem *MID E-Learning*. Diagram alir ini berfungsi untuk memvisualisasikan mekanisme perubahan data pengguna secara terkontrol, sehingga keakuratan informasi peserta tetap terjaga dan sesuai dengan kondisi terkini.



Gambar 3.9. Flowchart alur sistem admin – *edit participant details*

Berdasarkan *flowchart* tersebut, proses diawali ketika *Admin* melakukan *login* ke dalam sistem menggunakan kredensial yang valid. Sistem melakukan autentikasi untuk memastikan bahwa pengguna memiliki

hak akses sebagai *Admin*. Setelah berhasil masuk, *Admin* diarahkan ke halaman manajemen peserta yang menampilkan daftar seluruh peserta yang terdaftar dalam sistem.

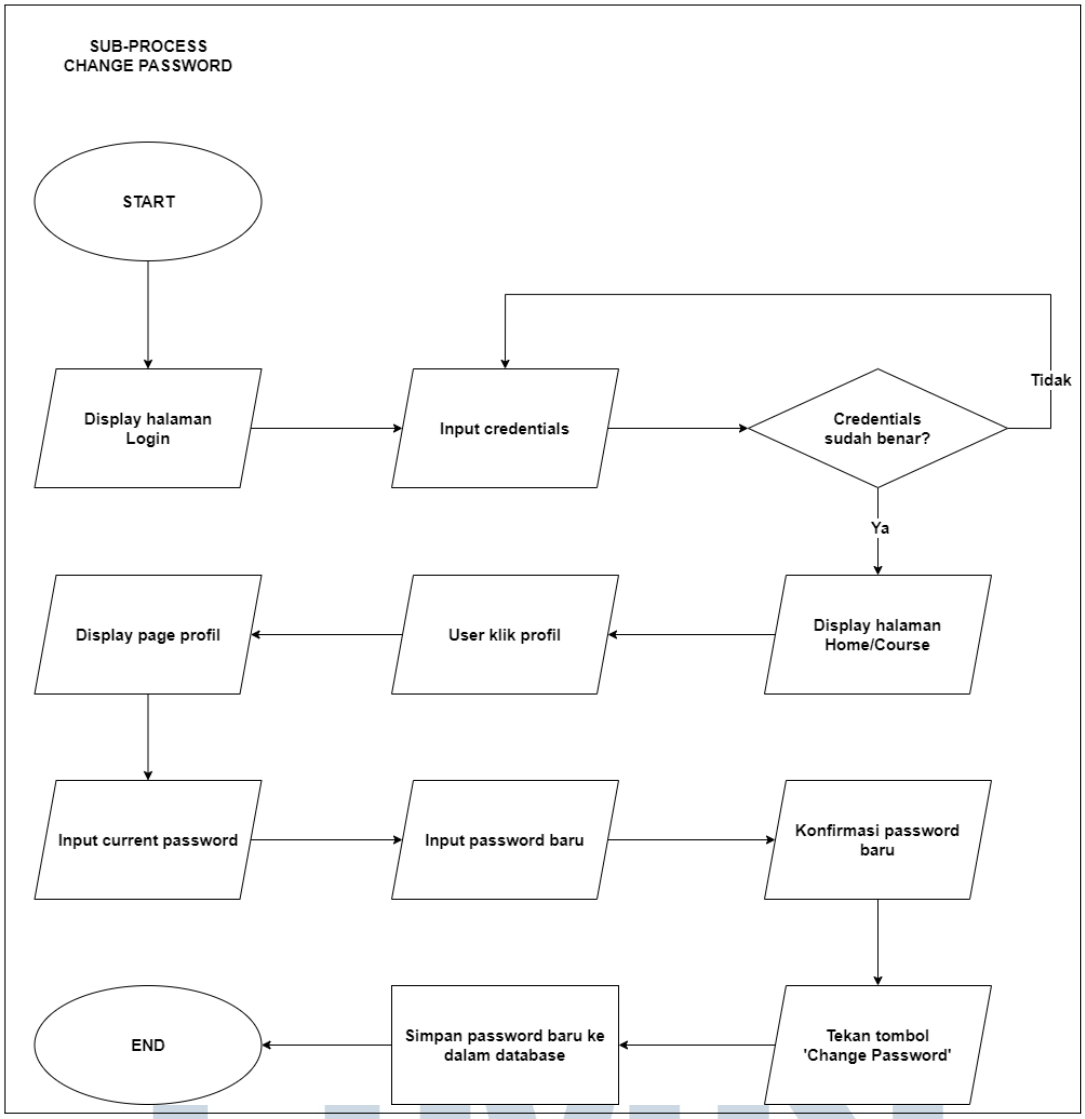
Selanjutnya, *Admin* memilih salah satu peserta yang informasinya akan diperbarui. Sistem kemudian menampilkan halaman detail peserta yang berisi data akun dan informasi pendukung lainnya. Pada tahap ini, *Admin* dapat melakukan pengeditan terhadap data yang diperlukan, seperti informasi identitas, data akun, atau pengaturan hak akses sesuai dengan kebijakan sistem.

Setelah perubahan data selesai dilakukan, *Admin* menekan tombol *Save* untuk mengonfirmasi pembaruan. Sistem kemudian menjalankan proses validasi terhadap data yang telah diubah guna memastikan kelengkapan dan kesesuaian format data. Apabila ditemukan data yang tidak valid, sistem akan menampilkan pesan kesalahan agar *Admin* dapat melakukan perbaikan.

Apabila seluruh data dinyatakan valid, sistem akan menyimpan informasi terbaru ke dalam *database* dan memperbarui data peserta pada sistem. Setelah proses penyimpanan berhasil, sistem menampilkan informasi peserta yang telah diperbarui pada halaman manajemen peserta. Dengan demikian, alur *edit participant details* ini mendukung pengelolaan data pengguna yang akurat, aman, dan terdokumentasi dengan baik dalam sistem *MID E-Learning*.

6. **Student – Change Password**

Berdasarkan Gambar 3.10, *flowchart Change Password* menggambarkan alur proses yang dilakukan oleh peserta (*student*) dalam mengubah kata sandi akun pada sistem *MID E-Learning*. Diagram alir ini dirancang untuk memastikan bahwa proses perubahan kata sandi berjalan secara terstruktur, aman, dan hanya dapat dilakukan oleh pemilik akun yang sah, sehingga keamanan data pengguna tetap terjaga.



Gambar 3.10. Flowchart alur sistem student – *change password*

Berdasarkan *flowchart* tersebut, proses diawali ketika peserta membuka halaman *login* dan memasukkan *credentials* berupa username dan kata sandi. Sistem kemudian melakukan proses autentikasi untuk memverifikasi kesesuaian data yang dimasukkan dengan data yang tersimpan di dalam *database*. Apabila *credentials* tidak valid, sistem akan menampilkan pesan kesalahan dan peserta diminta untuk mengulangi proses *login*. Sebaliknya, apabila *credentials* valid, peserta berhasil masuk ke dalam sistem dan diarahkan ke halaman *Home/Course*.

Setelah berhasil masuk, peserta mengakses menu profil dengan menekan ikon atau tombol profil yang tersedia pada antarmuka sistem. Sistem

kemudian menampilkan halaman profil pengguna yang berisi informasi akun serta opsi pengelolaan data pribadi. Pada tahap ini, peserta memilih fitur *change password* untuk memulai proses perubahan kata sandi.

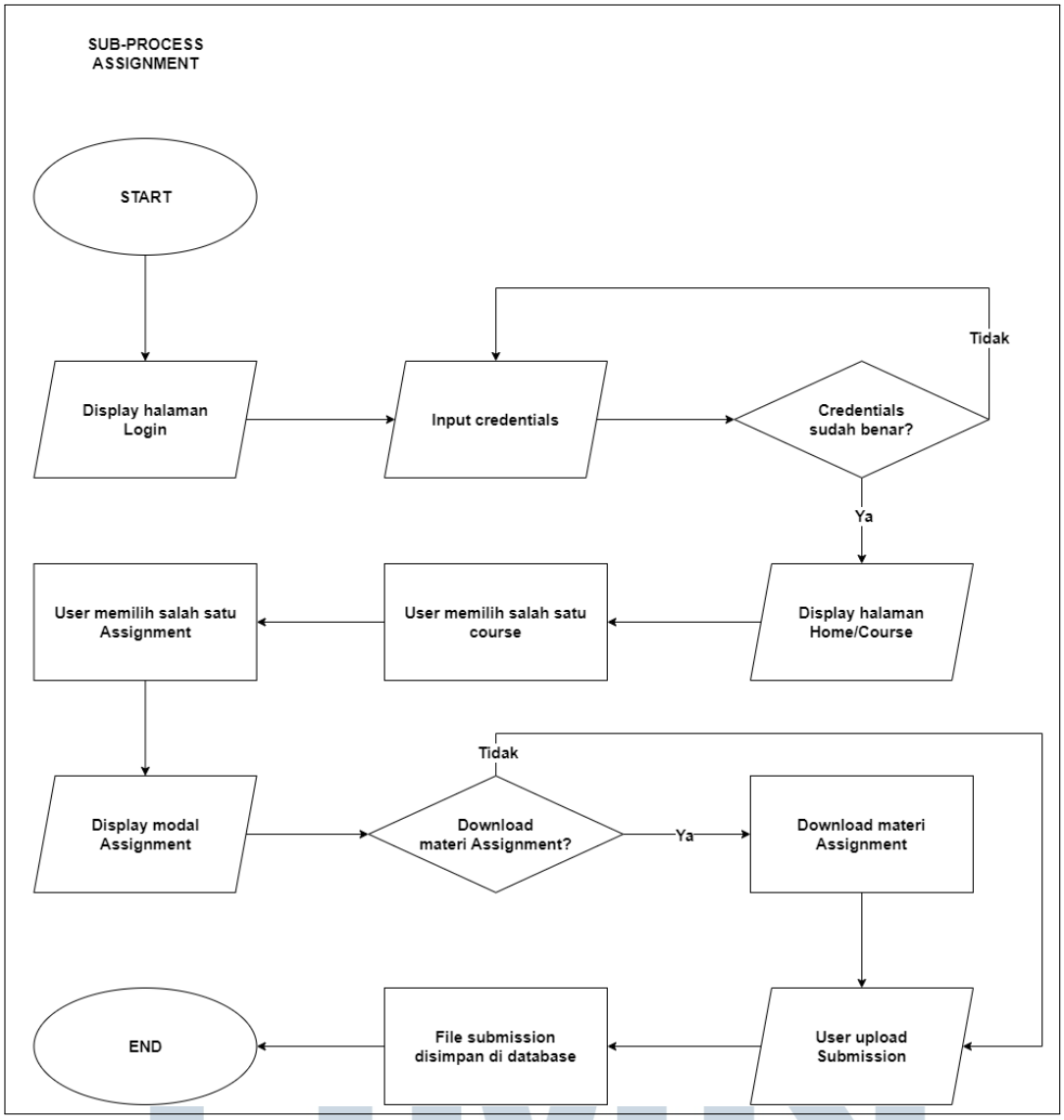
Selanjutnya, peserta diminta untuk memasukkan kata sandi lama (*current password*) sebagai langkah verifikasi keamanan. Tahapan ini bertujuan untuk memastikan bahwa perubahan kata sandi hanya dapat dilakukan oleh pengguna yang benar-benar mengetahui kata sandi sebelumnya. Setelah itu, peserta memasukkan kata sandi baru dan melakukan konfirmasi kata sandi baru guna memastikan kesesuaian dan menghindari kesalahan input.

Setelah seluruh data diisi dengan lengkap dan benar, peserta menekan tombol *Change Password* untuk mengirimkan permintaan perubahan kata sandi ke sistem. Sistem kemudian melakukan validasi terhadap kata sandi lama serta kesesuaian antara kata sandi baru dan konfirmasinya. Apabila validasi gagal, sistem akan menampilkan pesan kesalahan agar peserta dapat memperbaiki input yang diberikan.

Apabila seluruh proses validasi berhasil, sistem akan memperbarui kata sandi akun peserta dan menyimpannya ke dalam *database*. Dengan demikian, proses perubahan kata sandi dinyatakan selesai dan peserta dapat menggunakan kata sandi baru untuk proses *login* berikutnya. Alur ini memastikan bahwa mekanisme pengelolaan keamanan akun pada sistem *MID E-Learning* berjalan secara aman, terkontrol, dan sesuai dengan standar keamanan pengguna.

7. **Student – Assignment Upload**

Berdasarkan Gambar 3.11, *flowchart Assignment Upload* menggambarkan alur interaksi peserta dengan sistem *MID E-Learning* dalam proses pengumpulan *assignment*. Diagram alir ini dirancang untuk memastikan bahwa proses unggah tugas dilakukan secara terstruktur, aman, serta terdokumentasi dengan baik oleh sistem, sehingga setiap *submission* yang dikirimkan dapat dipertanggungjawabkan dan dikelola secara sistematis.



Gambar 3.11. Flowchart alur sistem student – *assignment upload*

Berdasarkan *flowchart* tersebut, proses diawali ketika peserta membuka halaman *login* dan memasukkan *credentials* yang dimiliki. Sistem kemudian melakukan proses autentikasi untuk memverifikasi kesesuaian data pengguna dengan data yang tersimpan di dalam *database*. Apabila *credentials* tidak valid, peserta diminta untuk mengulangi proses *login*. Sebaliknya, apabila autentikasi berhasil, peserta diarahkan ke halaman *Home/Course*.

Pada halaman *Home/Course*, peserta memilih salah satu *course* yang sedang diikuti. Sistem kemudian menampilkan daftar *assignment* yang

tersedia pada *course* tersebut. Peserta memilih *assignment* yang akan dikumpulkan, sehingga sistem menampilkan *modal assignment* yang berisi informasi tugas, deskripsi, tenggat waktu, serta opsi pendukung lainnya.

Jika peserta memerlukan referensi atau materi pendukung, peserta dapat memilih opsi untuk mengunduh materi *assignment*. Sistem akan menyediakan *file assignment* yang dapat diunduh dan dipelajari sebelum proses pengumpulan tugas dilakukan. Setelah memahami ketentuan tugas, peserta melanjutkan proses dengan mengunggah *file assignment* sebagai bentuk *submission*.

Selanjutnya, sistem menerima *file submission* yang diunggah dan melakukan proses penyimpanan ke dalam *database*. Penyimpanan ini bertujuan untuk memastikan bahwa data tugas tersimpan dengan aman serta dapat diakses oleh Mentor atau Admin untuk keperluan penilaian dan monitoring. Setelah proses penyimpanan berhasil, alur pengunggahan *assignment* dinyatakan selesai.

Dengan adanya alur *assignment upload* ini, sistem *MID E-Learning* mampu mendukung proses pengumpulan tugas secara tertib, terintegrasi, dan efisien, sekaligus meminimalkan risiko kehilangan data serta kesalahan dalam proses pengelolaan tugas peserta.

3.3.2 Desain UI

Proses perancangan antarmuka pengguna (*User Interface Design*) merupakan tahap awal yang sangat krusial dalam pengembangan sistem MID E-learning. Langkah ini saya lakukan sebelum memasuki proses pengkodean (*development*) dengan tujuan utama untuk memvisualisasikan struktur tampilan serta alur interaksi pengguna terhadap sistem secara menyeluruh. Dalam proyek ini, seluruh proses perancangan UI saya kerjakan secara mandiri, sekaligus dengan peran sebagai perancang (*UI Designer*) dan pengembang sistem (*Fullstack Developer*). Dengan mengambil peran ganda, saya memiliki kendali penuh dalam memastikan keselarasan antara desain visual, konsep sistem, dan implementasi teknis pada aplikasi yang dikembangkan.

Perancangan UI berfungsi sebagai panduan utama dalam membangun struktur tampilan, navigasi, serta elemen-elemen fungsional seperti menu pembelajaran, kartu informasi, grafik monitoring, tabel data, dan komponen

interaktif lainnya. Tujuan dari tahap ini adalah memastikan bahwa antarmuka sistem tidak hanya memenuhi kebutuhan fungsional, tetapi juga memberikan pengalaman pengguna yang intuitif, efisien, dan selaras dengan identitas visual PT. Madani Intelsysdata.

Dalam praktiknya, saya menggunakan aplikasi *Figma* sebagai alat bantu desain, yang mendukung proses pembuatan *wireframe*, *mockup*, serta prototipe interaktif. Meskipun saya bekerja secara mandiri tanpa bantuan tim, seluruh keputusan desain tetap diarahkan untuk memenuhi instruksi perusahaan, termasuk penggunaan warna identitas putih, biru, dan hijau, serta penerapan logo resmi perusahaan pada tampilan sistem. Dengan pendekatan ini, proses desain tidak hanya menghasilkan tampilan visual yang representatif, tetapi juga mempermudah transisi ke tahap pengembangan *front-end* dan *back-end*, karena seluruh keputusan desain telah disesuaikan sejak awal dengan kebutuhan teknis dan batasan sistem yang akan dibangun.

A Admin/Mentor

Antarmuka untuk Admin dirancang agar pengelola sistem dapat dengan mudah melakukan manajemen data, mengawasi aktivitas pengguna, serta mengatur konten pembelajaran yang tersedia. Desain dibuat fungsional, efisien, dan tetap konsisten dengan identitas visual perusahaan, sehingga administrator dapat menjalankan tugas pengelolaan secara terstruktur dan profesional tanpa kehilangan kenyamanan dalam penggunaan.

1. Login Page

Halaman *Login Page* untuk Admin, sebagaimana ditunjukkan pada Gambar 3.12, berfungsi sebagai gerbang utama bagi administrator sistem dalam mengakses fitur manajemen dan pengawasan pada platform pembelajaran PT. Madani Intelsysdata. Halaman ini dirancang untuk mendukung proses autentikasi yang aman dan efisien, sekaligus memberikan kesan profesional yang mencerminkan identitas sistem. Pada bagian atas halaman ditampilkan logo instansi beserta nama sistem sebagai identitas visual yang konsisten dan mudah dikenali oleh pengguna.

Gambar 3.12. Desain halaman login (Admin)

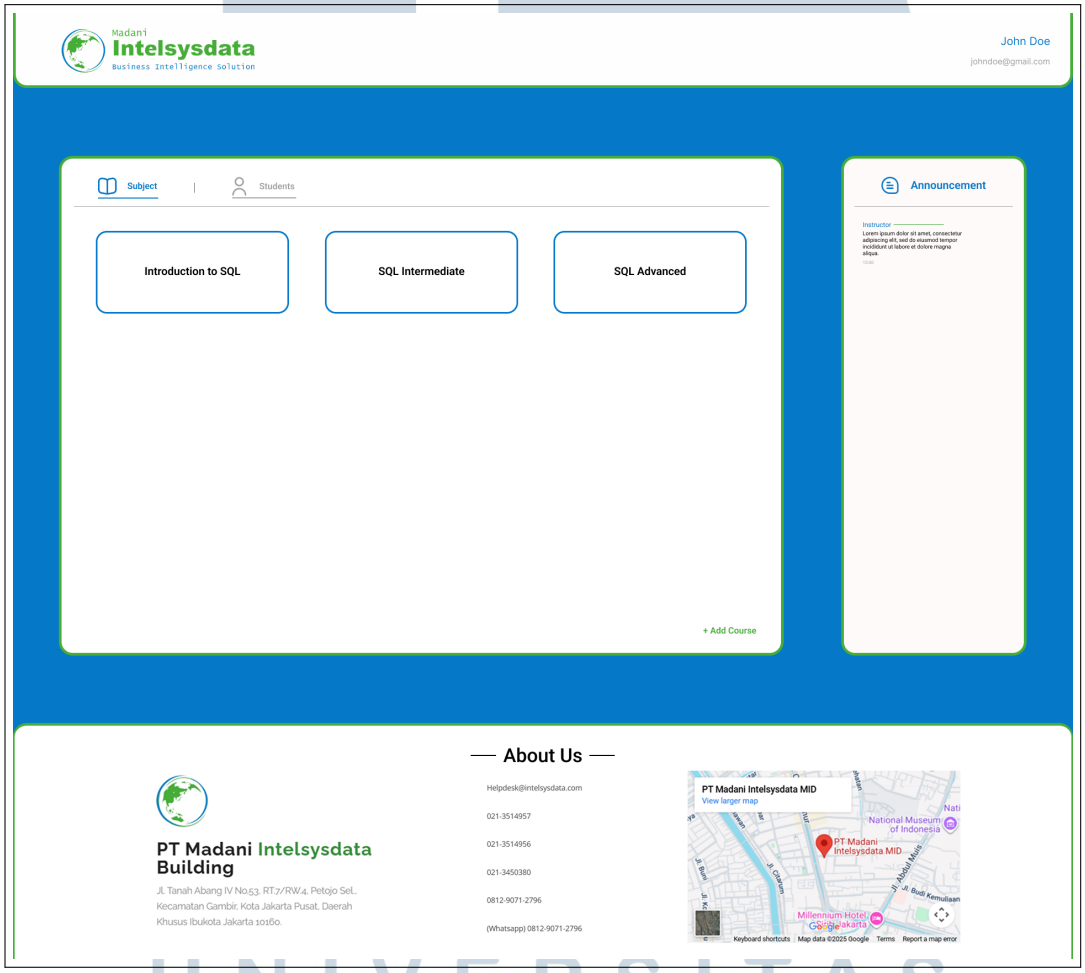
Pada bagian tengah halaman terdapat formulir *login* yang terdiri dari dua kolom input utama, yaitu *E-mail* dan *Password*. Kedua kolom ini digunakan sebagai data autentikasi untuk memverifikasi identitas Admin sebelum mengakses sistem. Selain itu, tersedia tombol *Log in* berwarna hijau yang berfungsi sebagai pemicu proses autentikasi. Tidak disediakan fitur registrasi pada halaman ini karena akun Admin dibuat dan dikelola secara terpusat oleh pihak pengelola sistem guna menjaga kontrol dan keamanan akses.

Seluruh elemen pada halaman *Login Page* Admin dirancang dengan mempertimbangkan aspek keamanan, kemudahan penggunaan, dan kejelasan antarmuka. Penempatan kolom input yang sederhana dan terfokus memungkinkan Admin melakukan proses *login* dengan cepat dan minim kesalahan. Penggunaan warna kontras pada tombol aksi utama membantu pengguna mengenali fungsi utama halaman secara intuitif.

Dari sisi visual, halaman ini dilengkapi dengan latar grafis bertema teknologi, seperti ilustrasi sirkuit digital dan ikon globe, yang merepresentasikan konektivitas dan sistem informasi modern. Struktur halaman *login* ini juga telah disesuaikan untuk mendukung mekanisme otorisasi berbasis peran (*role-based access control*). Dengan demikian, setelah proses *login* berhasil, Admin akan langsung diarahkan ke halaman *dashboard* sesuai dengan hak akses yang dimilikinya.

2. **Home / Subject List Admin**

Halaman *Home / Subject List* untuk Admin, sebagaimana ditunjukkan pada Gambar 3.13, merupakan tampilan utama yang muncul setelah administrator berhasil menyelesaikan proses *login*. Halaman ini berfungsi sebagai pusat kendali awal bagi Admin dalam mengelola daftar mata pelajaran (*subject*) atau *course* yang tersedia pada platform pembelajaran PT. Madani Intelsysdata.



Gambar 3.13. Desain halaman home / course list (Admin)

Pada bagian atas halaman ditampilkan *header* yang memuat logo instansi, nama sistem, serta informasi akun Admin yang sedang aktif. Di bagian utama halaman, daftar mata pelajaran ditampilkan dalam format *grid* yang tersusun rapi dan mudah dipindai.

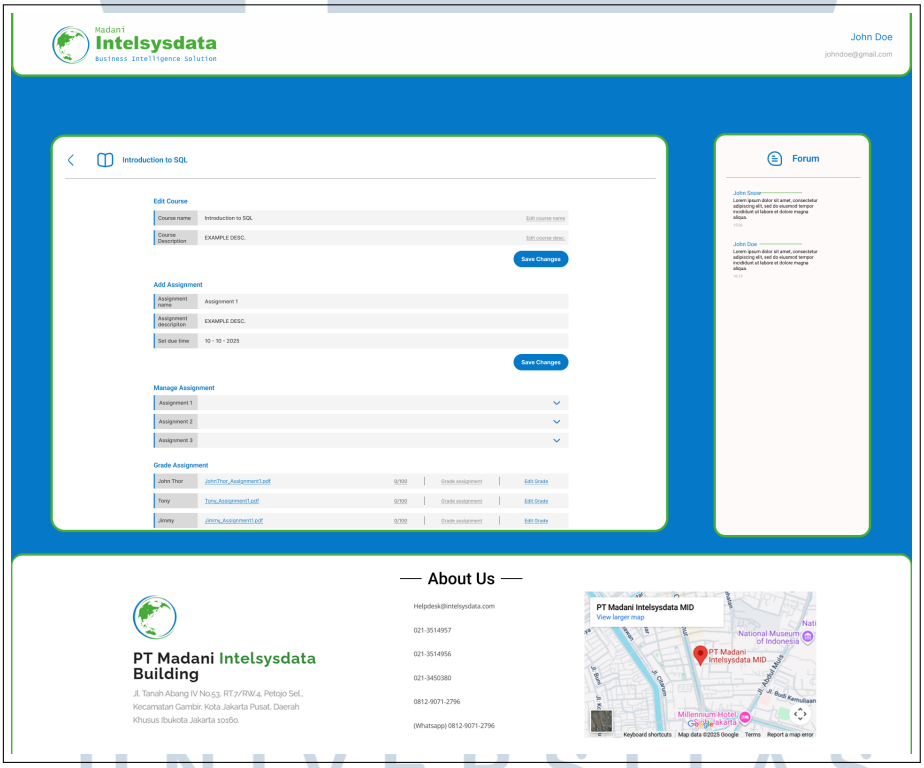
Pada bagian bawah daftar mata pelajaran, terdapat tombol + *Add Course* yang berfungsi sebagai akses utama untuk menambahkan mata

pelajaran baru ke dalam sistem. Tombol ini terintegrasi langsung dengan sistem *backend* sehingga proses penyimpanan data dapat dilakukan secara real-time.

Selain itu, halaman ini juga dilengkapi dengan panel pengumuman di sisi kanan serta informasi kontak dan lokasi institusi pada bagian bawah halaman.

3. **Course Details Admin**

Halaman *Course Details* untuk Admin, sebagaimana ditunjukkan pada Gambar 3.14, berfungsi sebagai pusat pengelolaan konten dan aktivitas pembelajaran untuk setiap mata pelajaran yang tersedia dalam sistem *MID E-Learning*.



Gambar 3.14. Desain halaman course details (Admin)

Pada bagian atas halaman ditampilkan judul mata pelajaran serta formulir pengeditan informasi *course*. Di bagian tengah halaman disediakan formulir penambahan tugas baru (*Add Assignment*) yang mencakup nama tugas, deskripsi, dan batas waktu pengumpulan.

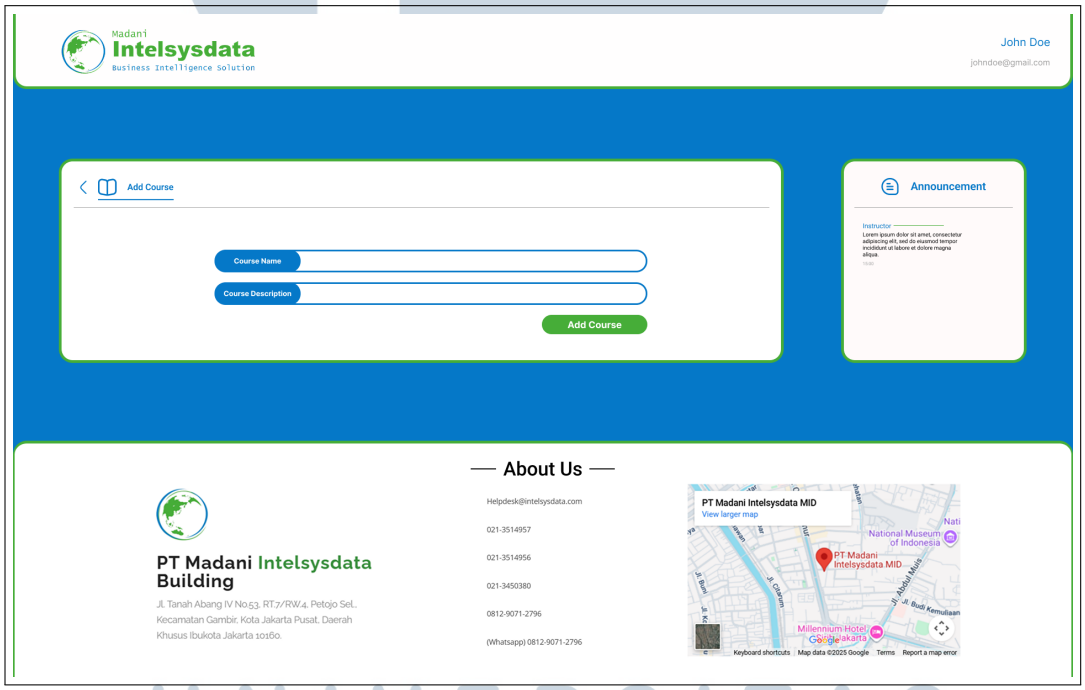
Bagian *Manage Assignment* dan *Grade Assignment* digunakan untuk mengelola tugas serta melakukan evaluasi pengumpulan tugas peserta. Di

sisi kanan halaman, tersedia panel forum diskusi sebagai sarana komunikasi antara Admin, mentor, dan peserta.

Secara keseluruhan, struktur halaman *Course Details* Admin dirancang untuk mendukung efisiensi kerja, kejelasan alur pengelolaan, serta integrasi penuh dengan sistem *backend*.

4. **Add Course Admin**

Halaman *Add Course* untuk Admin, sebagaimana ditunjukkan pada Gambar 3.15, berfungsi sebagai antarmuka input data yang digunakan untuk menambahkan mata pelajaran baru ke dalam sistem pembelajaran *MID E-Learning*. Halaman ini dirancang untuk mendukung proses perluasan konten pembelajaran secara terstruktur, sehingga Admin dapat mengelola daftar mata pelajaran sesuai dengan kebutuhan institusi.



Gambar 3.15. Desain halaman add course (Admin)

Pada bagian utama halaman, disediakan formulir penambahan mata pelajaran yang terdiri dari dua kolom input utama, yaitu *Course Name* dan *Course Description*. Kolom *Course Name* digunakan untuk memasukkan nama mata pelajaran, sedangkan kolom *Course Description* berfungsi untuk menjelaskan gambaran umum atau ruang lingkup materi yang akan dipelajari. Di bawah formulir tersebut, terdapat tombol aksi berwarna

hijau bertuliskan *Add Course* yang berfungsi untuk menyimpan data mata pelajaran yang telah diinput ke dalam sistem.

Struktur halaman ini difokuskan pada kemudahan penggunaan dan efisiensi proses input data. Tata letak formulir yang sederhana dan langsung membantu Admin dalam mengisi data dengan cepat serta mengurangi potensi kesalahan pengisian. Sebelum data dikirim ke sistem *backend*, sistem akan melakukan proses validasi untuk memastikan bahwa seluruh kolom yang wajib diisi telah terpenuhi dan sesuai dengan ketentuan yang berlaku.

Selain formulir penambahan mata pelajaran, halaman ini tetap mempertahankan elemen pendukung yang konsisten dengan halaman Admin lainnya, seperti panel pengumuman di sisi kanan sebagai sarana penyampaian informasi dan bagian footer yang memuat informasi kontak serta lokasi institusi. Konsistensi elemen ini bertujuan untuk menjaga keseragaman antarmuka dan meningkatkan pengalaman pengguna.

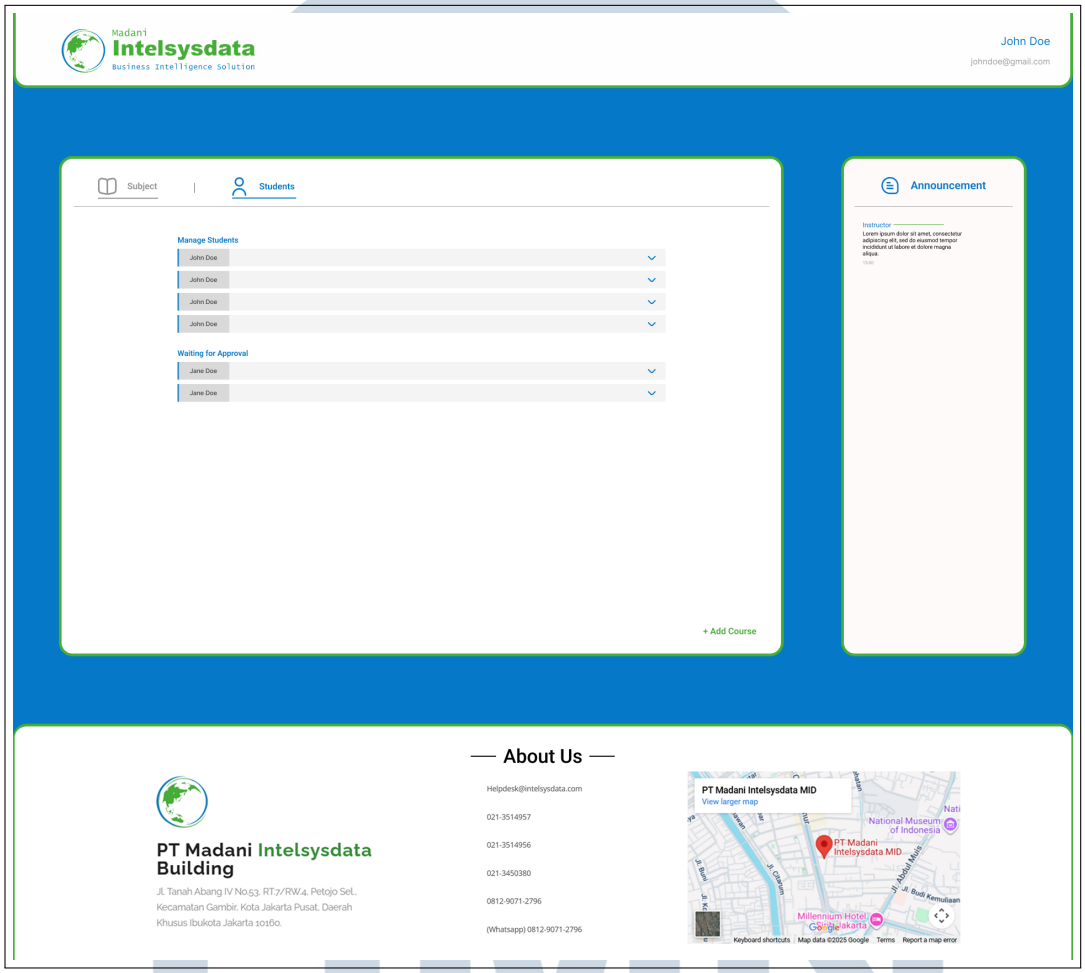
Secara keseluruhan, desain halaman *Add Course* Admin mendukung pengelolaan konten pembelajaran yang terintegrasi dan skalabel. Dengan antarmuka yang jelas serta proses input yang terkontrol, Admin dapat menambahkan mata pelajaran baru secara sistematis dan profesional, sehingga mendukung pengembangan platform pembelajaran *MID E-Learning* secara berkelanjutan.

5. **Student List Admin**

Halaman *Student List* untuk admin sebagaimana ditunjukkan pada Gambar 3.16 berfungsi sebagai modul utama dalam pengelolaan data peserta PKL yang terdaftar di dalam sistem e-learning. Halaman ini dirancang untuk memberikan kontrol penuh kepada admin dalam memantau status keikutsertaan peserta, mulai dari proses pendaftaran hingga peserta dinyatakan aktif secara resmi dalam sistem.

Tampilan halaman terbagi ke dalam dua bagian utama, yaitu *Manage Students* dan *Waiting for Approval*. Pada bagian *Manage Students*, ditampilkan daftar peserta yang telah aktif dan disetujui, lengkap dengan informasi dasar seperti nama peserta, identitas akun, serta status keanggotaan. Setiap entri peserta dilengkapi dengan menu aksi berbasis dropdown yang memungkinkan admin untuk melakukan berbagai tindakan, seperti melihat detail data peserta, melakukan pengeditan informasi akun,

maupun menghapus akun apabila sudah tidak diperlukan. Mekanisme ini mendukung pengelolaan data yang fleksibel dan terpusat tanpa harus berpindah ke halaman lain.



Gambar 3.16. Desain halaman student list (Admin)

Sementara itu, bagian *Waiting for Approval* menampilkan daftar peserta yang telah melakukan pendaftaran namun belum mendapatkan persetujuan dari admin. Pada bagian ini, admin dapat melakukan proses verifikasi secara manual dengan meninjau data peserta sebelum memberikan persetujuan atau penolakan. Proses ini penting untuk menjaga validitas dan keamanan sistem, serta memastikan bahwa hanya peserta yang memenuhi kriteria yang dapat mengakses fitur pembelajaran.

Struktur halaman disusun menggunakan tabel yang rapi dan responsif, dengan elemen interaktif seperti dropdown aksi dan indikator status yang memudahkan admin dalam memahami kondisi setiap peserta

secara cepat. Panel pengumuman di sisi kanan halaman tetap disediakan sebagai sarana penyampaian informasi umum, sedangkan informasi kontak pada bagian bawah halaman menjaga konsistensi tampilan antarmuka dengan halaman lainnya. Dengan desain dan alur kerja tersebut, halaman *Student List* mampu mendukung pengelolaan peserta PKL secara akurat, terstruktur, dan efisien sesuai kebutuhan operasional admin.

B User (Student)

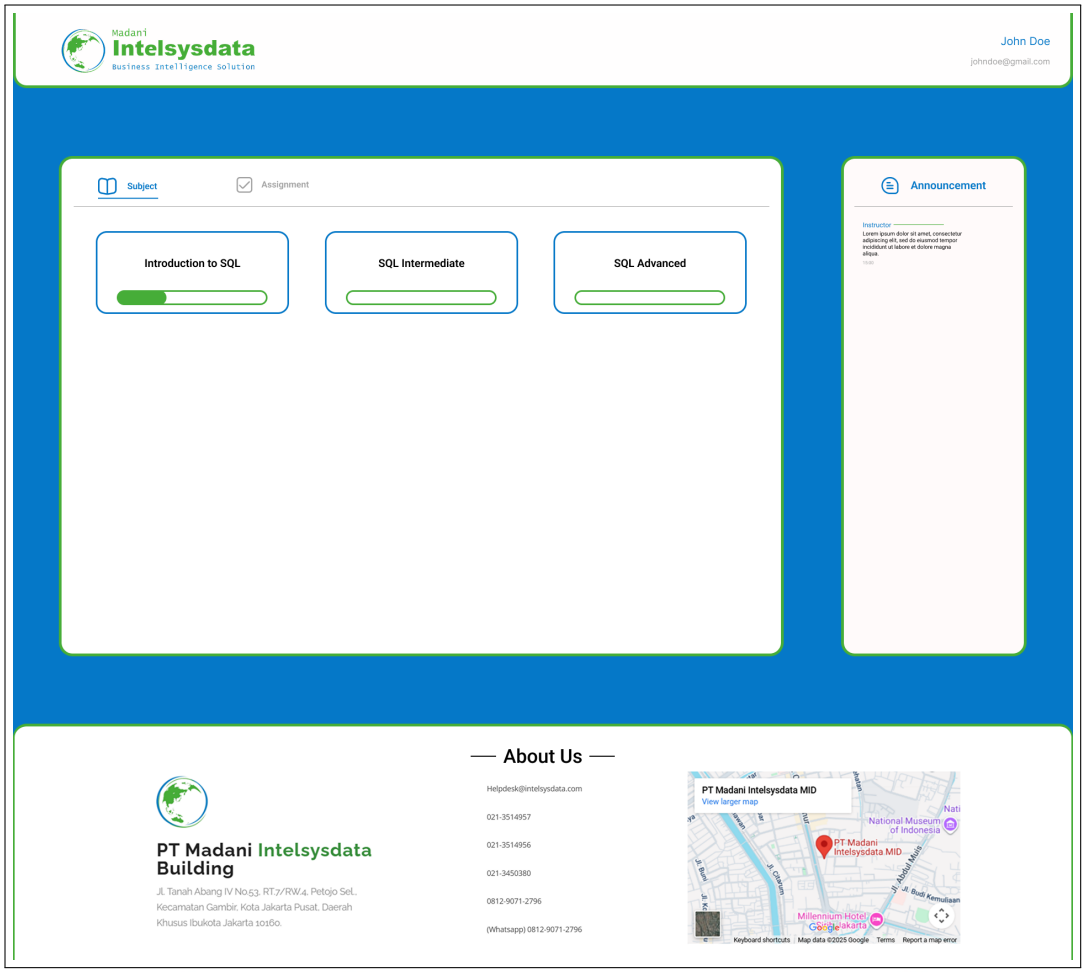
Antarmuka untuk User (Student) dirancang agar siswa PKL dapat dengan mudah mengakses materi pembelajaran, mengelola tugas, serta memantau progres belajar mereka. Desain dibuat sederhana, intuitif, dan konsisten dengan identitas visual perusahaan.

Halaman yang tersedia untuk User (Student) meliputi:

1. Home Page

Halaman *Home Page/Subject List* sebagaimana ditunjukkan pada Gambar 3.17 merupakan halaman utama yang pertama kali ditampilkan setelah pengguna berhasil melakukan proses login ke dalam sistem pembelajaran PT Madani Intelsysdata. Halaman ini berfungsi sebagai pusat navigasi awal bagi peserta untuk mengakses seluruh mata pelajaran yang telah terdaftar di dalam sistem. Selain itu, halaman ini juga memberikan gambaran umum mengenai aktivitas pembelajaran yang sedang berlangsung, sehingga peserta dapat langsung melanjutkan proses belajar sesuai dengan course yang diikuti.

Sebagai halaman utama, *Home Page* dirancang untuk menyajikan informasi inti secara ringkas dan terstruktur, tanpa membebani pengguna dengan detail teknis yang berlebihan. Melalui halaman ini, peserta dapat memahami posisi mereka dalam alur pembelajaran serta mengakses mata pelajaran secara cepat dan efisien.



Gambar 3.17. Desain halaman home (Student)

Pada bagian atas halaman, terdapat *header* yang secara konsisten menampilkan logo instansi, nama sistem, serta informasi akun pengguna yang sedang aktif. Header ini berfungsi sebagai identitas visual sistem sekaligus elemen navigasi dasar yang membantu pengguna tetap terorientasi selama menggunakan aplikasi. Desain header dibuat minimalis dan responsif agar dapat menyesuaikan tampilan pada berbagai ukuran layar dan perangkat.

Di bagian utama halaman, daftar mata pelajaran ditampilkan dalam bentuk tab dengan tab *Subject* sebagai tab aktif. Setiap mata pelajaran disajikan dalam bentuk kartu atau modul yang dilengkapi dengan indikator progres pembelajaran. Indikator ini merepresentasikan tingkat penyelesaian materi oleh peserta, sehingga pengguna dapat memantau perkembangan pembelajaran mereka secara mandiri. Penyusunan mata pelajaran dilakukan

secara terstruktur dan berurutan, mendukung alur pembelajaran yang sistematis dari materi dasar hingga materi lanjutan.

Di sisi kanan halaman, terdapat panel pengumuman yang berfungsi sebagai media komunikasi antara instruktur dan peserta. Panel ini menampilkan informasi penting seperti pengumuman akademik, jadwal kegiatan, serta pemberitahuan terkait tugas dan evaluasi pembelajaran. Panel pengumuman dirancang sebagai komponen dinamis yang dapat diperbarui secara berkala oleh instruktur, tanpa mengganggu fokus utama pengguna terhadap daftar mata pelajaran.

Seluruh elemen pada halaman *Home Page/Subject List* disusun dengan memperhatikan konsistensi visual, kejelasan hierarki informasi, serta kemudahan navigasi bagi pengguna. Tata letak halaman menggunakan struktur grid yang fleksibel dan responsif, sehingga mendukung kompatibilitas tampilan pada perangkat desktop maupun mobile. Pada bagian bawah halaman, ditampilkan informasi kontak lengkap institusi, meliputi alamat kantor, alamat email, nomor telepon, dan WhatsApp, serta peta lokasi interaktif yang membantu pengguna mengenali lokasi fisik perusahaan.

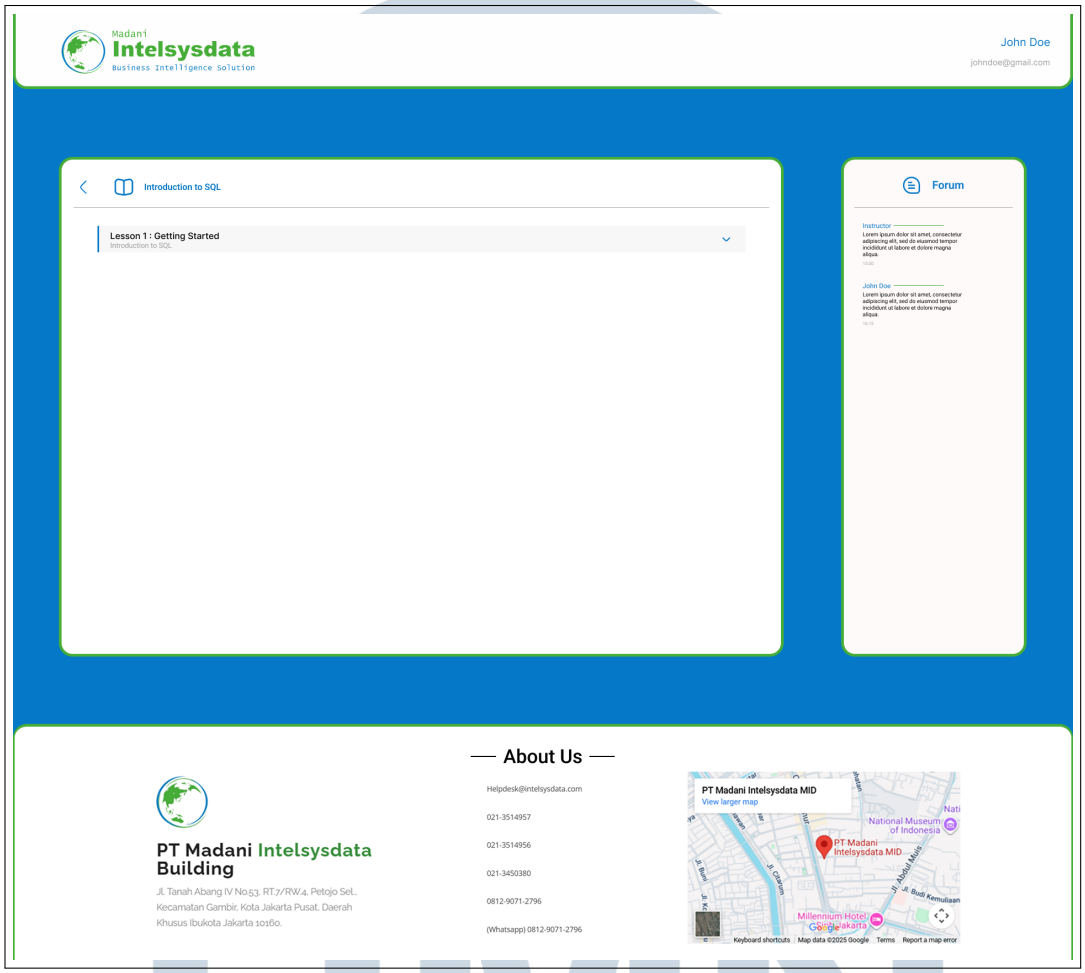
Pendekatan desain pada halaman ini tidak hanya berfokus pada aspek estetika, tetapi juga mempertimbangkan aspek fungsionalitas dan skalabilitas sistem. Struktur halaman memungkinkan penambahan mata pelajaran atau modul pembelajaran baru tanpa mengubah tata letak utama, sehingga sistem dapat berkembang secara berkelanjutan. Dengan demikian, halaman *Home Page* mampu memberikan pengalaman pengguna yang intuitif, informatif, dan profesional dalam mendukung proses pembelajaran daring secara efektif.

2. **Subjects Page**

Halaman *Subject Details* sebagaimana ditunjukkan pada Gambar 3.18 merupakan halaman lanjutan yang diakses oleh peserta setelah memilih salah satu mata pelajaran pada halaman *Home Page*. Halaman ini berfungsi sebagai pusat aktivitas pembelajaran untuk satu mata pelajaran tertentu, di mana peserta dapat melihat informasi materi serta mengelola tugas yang berkaitan langsung dengan mata pelajaran tersebut.

Secara fungsional, halaman ini dirancang untuk memfasilitasi interaksi peserta dengan konten pembelajaran pada level mata pelajaran,

sehingga seluruh informasi akademik yang relevan dapat diakses secara terfokus dalam satu tampilan.



Gambar 3.18. Desain halaman subjects (Student)

Pada bagian utama halaman, sistem menampilkan detail mata pelajaran yang sedang diikuti, meliputi judul mata pelajaran dan deskripsi singkat yang menjelaskan ruang lingkup materi pembelajaran. Informasi ini memberikan konteks awal kepada peserta sebelum melanjutkan ke aktivitas pembelajaran yang lebih spesifik.

Di bawah bagian deskripsi, ditampilkan daftar tugas (*assignment list*) yang terkait dengan mata pelajaran tersebut. Setiap tugas disajikan sebagai entri terpisah yang dapat dipilih oleh peserta untuk melihat detail lebih lanjut atau melakukan proses pengumpulan tugas. Penyajian daftar tugas ini membantu peserta dalam mengidentifikasi kewajiban akademik yang harus diselesaikan serta mengatur prioritas pengerjaan tugas secara

lebih terstruktur.

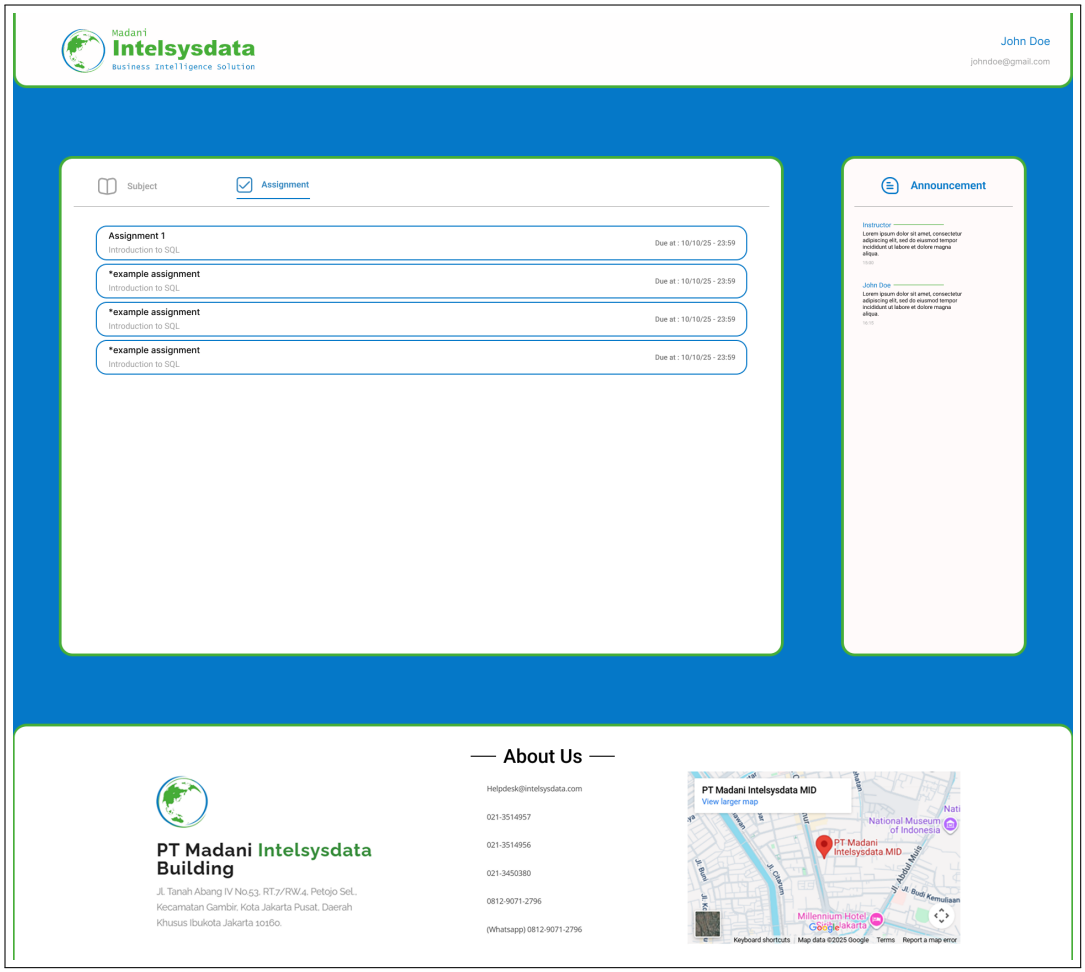
Selain itu, halaman ini juga menyediakan panel pengumuman yang berkaitan langsung dengan mata pelajaran yang sedang diakses. Panel tersebut digunakan untuk menyampaikan informasi tambahan dari instruktur, seperti arahan pengerjaan tugas, pembaruan materi, atau pengumuman akademik lain yang bersifat kontekstual terhadap mata pelajaran.

Dengan pemisahan konten yang jelas antara informasi mata pelajaran, daftar tugas, dan pengumuman, halaman *Subject Details* mampu mendukung proses pembelajaran yang terfokus dan terorganisir. Struktur halaman ini memungkinkan pengembangan lanjutan, seperti penambahan materi atau aktivitas pembelajaran baru, tanpa mengganggu alur penggunaan yang telah ada, sehingga sistem tetap fleksibel dan mudah dikembangkan sesuai kebutuhan pembelajaran daring.

3. **Assignment Page**

Halaman *Assignment List* sebagaimana ditunjukkan pada Gambar 3.19 merupakan halaman yang digunakan oleh peserta untuk mengelola aktivitas pengumpulan tugas selama mengikuti pembelajaran. Halaman ini berperan sebagai titik sentral yang menyajikan seluruh tugas dari berbagai mata pelajaran dalam satu tampilan terpadu, sehingga peserta dapat memahami beban akademik yang sedang dijalani secara menyeluruh.

Melalui halaman ini, peserta tidak hanya memperoleh informasi mengenai tugas yang harus diselesaikan, tetapi juga dapat mengatur prioritas pengerjaan berdasarkan tenggat waktu yang telah ditentukan. Dengan demikian, halaman *Assignment List* mendukung keteraturan proses belajar serta membantu peserta dalam mengelola waktu dan tanggung jawab akademik secara lebih efektif.



Gambar 3.19. Desain halaman assignment (Student)

Pada bagian utama halaman, daftar tugas ditampilkan pada tab *Assignment* yang berisi kumpulan tugas dari seluruh mata pelajaran yang diikuti peserta. Setiap tugas disajikan dengan informasi inti berupa judul tugas, ringkasan deskripsi, serta batas waktu pengumpulan. Penyajian ini memberikan konteks yang cukup bagi peserta untuk memahami tujuan tugas tanpa harus membuka detail lebih lanjut terlebih dahulu.

Pola tampilan daftar tugas dibuat seragam untuk setiap entri, sehingga peserta dapat dengan cepat mengenali informasi penting dan membandingkan tenggat waktu antar tugas. Pendekatan ini membantu peserta dalam menentukan urutan pengerjaan, khususnya ketika menghadapi beberapa tugas dalam periode waktu yang berdekatan.

Di sisi halaman, panel pengumuman berfungsi sebagai media pendukung yang menyampaikan informasi tambahan dari instruktur, seperti

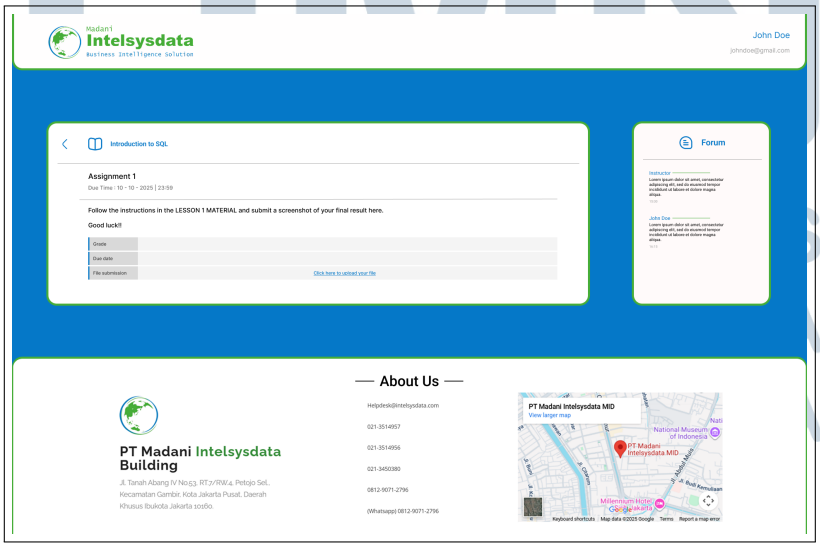
arahan pengerjaan tugas, pengingat tenggat waktu, atau penjelasan teknis tertentu. Keberadaan panel ini memperkaya konteks pembelajaran tanpa mengalihkan perhatian utama peserta dari daftar tugas yang tersedia.

Secara keseluruhan, halaman *Assignment List* dirancang untuk memberikan pengalaman pengelolaan tugas yang jelas, terarah, dan mudah dipahami. Struktur halaman yang terorganisir memungkinkan sistem menampilkan tugas secara konsisten sekaligus membuka peluang pengembangan fitur lanjutan di masa mendatang, tanpa mengubah alur penggunaan yang telah dipahami oleh peserta.

4. **Assignment Details Page**

Halaman *Assignment Details* sebagaimana ditunjukkan pada Gambar 3.20 merupakan halaman lanjutan yang digunakan peserta untuk mengakses informasi rinci dari satu tugas tertentu. Halaman ini dirancang sebagai titik fokus proses penyelesaian tugas, di mana peserta dapat memahami ketentuan tugas secara menyeluruh sebelum melakukan pengumpulan hasil pekerjaan.

Pada bagian konten utama, sistem menampilkan detail tugas secara terstruktur yang mencakup judul tugas, batas waktu pengumpulan, serta instruksi pengerjaan. Penyajian informasi ini bertujuan untuk memastikan peserta memperoleh pemahaman yang jelas mengenai ruang lingkup dan ketentuan tugas, sehingga risiko kesalahan dalam pengerjaan maupun pengumpulan dapat diminimalkan.



Gambar 3.20. Desain halaman assignment details (Student)

Fitur unggah berkas disediakan sebagai komponen utama pada halaman ini untuk mendukung proses pengumpulan tugas secara daring. Peserta dapat mengunggah file hasil pekerjaan langsung melalui sistem, kemudian menerima indikator status pengumpulan sebagai konfirmasi bahwa proses unggah telah berhasil dilakukan. Mekanisme ini membantu peserta dalam memantau kesiapan dan kelengkapan tugas sebelum melewati batas waktu yang ditentukan.

Secara keseluruhan, halaman *Assignment Details* dirancang untuk memfasilitasi proses penyelesaian dan pengumpulan tugas secara terarah dan terkontrol. Dengan pemusatan informasi dan fungsi utama dalam satu halaman, sistem mampu mendukung alur pembelajaran yang rapi, efisien, dan sesuai dengan kebutuhan akademik peserta.

3.3.3 Implementasi

Subbab ini menjelaskan proses penerapan sistem MID E-learning berdasarkan hasil perancangan yang telah dilakukan pada tahap sebelumnya. Pada tahap ini, seluruh kebutuhan fungsional sistem direalisasikan ke dalam bentuk antarmuka pengguna dan logika sistem yang terintegrasi dengan backend. Implementasi mencakup pengelolaan autentikasi, pengaturan hak akses pengguna, manajemen data pembelajaran, serta dukungan komunikasi antar pengguna. Seluruh fitur diimplementasikan dengan memperhatikan aspek keamanan, konsistensi data, dan kemudahan penggunaan, sehingga sistem dapat beroperasi secara efektif dan sesuai dengan tujuan pengembangan aplikasi pembelajaran daring.

A Admin/Mentor

Bagian ini membahas realisasi fungsional sistem MID E-learning dari sisi pengguna dengan hak akses administratif dan pembimbing. Pada tahap ini, sistem diimplementasikan untuk mendukung aktivitas pengelolaan pembelajaran secara menyeluruh, mulai dari proses autentikasi, manajemen mata pelajaran, pengelolaan peserta, hingga pemantauan progres dan evaluasi hasil belajar. Implementasi ini dirancang dengan menerapkan pengendalian akses berbasis peran (*role-based access control*) sehingga setiap fungsi yang tersedia hanya dapat diakses sesuai dengan kewenangan masing-masing peran, baik sebagai admin maupun mentor.

Dengan demikian, implementasi pada sisi *Admin/Mentor* memastikan bahwa proses administrasi, koordinasi, dan pengawasan pembelajaran dapat berjalan secara terstruktur, aman, dan terintegrasi dalam sistem MID E-learning.

1. **Login**

Halaman *Login* pada sistem MID E-learning berfungsi sebagai mekanisme autentikasi pengguna sebelum memperoleh akses ke dalam sistem, sebagaimana ditunjukkan pada Gambar 3.21. Pada tahap ini, pengguna diwajibkan memasukkan kredensial yang valid sebagai bentuk verifikasi identitas. Proses autentikasi ini dirancang untuk memastikan bahwa hanya pengguna yang telah terdaftar dan memiliki hak akses yang sah yang dapat menggunakan sistem sesuai dengan peran yang dimilikinya.

Gambar 3.21. Halaman Login sistem MID E-learning

Setelah pengguna mengirimkan data kredensial melalui halaman login, sistem akan melakukan proses validasi dengan mencocokkan informasi yang dimasukkan terhadap data pengguna yang tersimpan di dalam basis data. Proses autentikasi dan pengalihan halaman ini diimplementasikan melalui konfigurasi sistem yang ditunjukkan pada Kode 3.1 hingga Kode 3.6. Apabila proses autentikasi berhasil, sistem akan memberikan akses dan mengarahkan pengguna ke halaman utama sesuai dengan perannya, seperti *Admin*, *Mentor*, atau *Student*. Sebaliknya, apabila data yang dimasukkan tidak sesuai atau tidak valid, sistem akan menolak akses dan meminta pengguna untuk mengulangi proses login.

Implementasi autentikasi pada sistem ini dikelola menggunakan *Spring Security*, yang dikonfigurasikan melalui kelas *SecurityConfig*, sebagaimana ditunjukkan pada Kode 3.1. Pada konfigurasi ini, digunakan algoritma *BCrypt* sebagai *password encoder* untuk memastikan bahwa kata sandi pengguna disimpan dalam bentuk terenkripsi di dalam basis data, sehingga meningkatkan keamanan data pengguna.

```
@Bean
public BCryptPasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}

@Bean
public AuthenticationSuccessHandler successHandler() {
    return (request, response, authentication) -> {
        String redirectURL = "";
        if (authentication.getAuthorities().stream()
            .anyMatch(a -> a.getAuthority().equals("
    ROLE_ADMIN"))) {
            redirectURL = "/admin/dashboard";
        } else if (authentication.getAuthorities().stream()
            .anyMatch(a -> a.getAuthority().equals("
    ROLE_MENTOR"))) {
            redirectURL = "/mentor/dashboard";
        } else {
            redirectURL = "/user/dashboard";
        }
        response.sendRedirect(redirectURL);
    };
}
```

Kode 3.1: Konfigurasi password encoder dan success handler

AuthenticationSuccessHandler digunakan untuk mengatur proses pengalihan halaman setelah login berhasil, sebagaimana ditunjukkan pada Kode 3.1. Sistem akan membaca peran pengguna yang terautentikasi, kemudian mengarahkan pengguna ke halaman dashboard yang sesuai dengan perannya masing-masing. Mekanisme ini mendukung penerapan pengendalian akses berbasis peran (*role-based access control*).

Pengaturan alur autentikasi secara keseluruhan dilakukan melalui *SecurityFilterChain*, sebagaimana ditunjukkan pada Kode 3.2, yang mengatur halaman mana saja yang dapat diakses tanpa autentikasi serta

konfigurasi form login dan logout.

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http)
    throws Exception {
    http
      .authorizeHttpRequests(auth -> auth
          .requestMatchers("/", "/login", "/process-login",
                           "/logout", "/css/**", "/js/**", "
    /images/**")
          .permitAll()
          .anyRequest().authenticated()
      )
      .formLogin(login -> login
          .loginPage("/login")
          .loginProcessingUrl("/process-login")
          .successHandler(successHandler())
          .failureUrl("/login?error=true")
          .permitAll()
      )
      .logout(logout -> logout
          .logoutUrl("/logout")
          .logoutSuccessUrl("/login?logout=true")
      );
    return http.build();
}
```

Kode 3.2: Konfigurasi Security Filter Chain

Proses pengambilan data pengguna dan perannya dilakukan melalui implementasi *UserDetailsService*, sebagaimana ditunjukkan pada Kode 3.3. Pada tahap ini, sistem memuat data pengguna berdasarkan *username*, menetapkan peran dalam bentuk *authority*, serta memeriksa status persetujuan akun sebelum mengizinkan login.

```
@Override
public UserDetails loadUserByUsername(String username)
        throws UsernameNotFoundException {
    User user = userRepository.findByUsername(username)
        .orElseThrow(() -> new UsernameNotFoundException(
            "User tidak ditemukan: " + username));

    String role = user.getFormattedRole();

```

```
10      return org.springframework.security.core.userdetails.
    User.builder()
11          .username(user.getUsername())
12          .password(user.getPassword())
13          .authorities(List.of(new SimpleGrantedAuthority(role
    )))
14          .disabled(!Boolean.TRUE.equals(user.getApproved()))
15          .accountLocked(false)
16          .accountExpired(false)
17          .credentialsExpired(false)
18          .build();
19  }
```

Kode 3.3: Implementasi UserDetailsService

Status akun pengguna diperiksa melalui atribut *approved*, sebagaimana diimplementasikan pada Kode 3.3. Apabila akun belum disetujui oleh admin, maka sistem akan menonaktifkan akun tersebut dan menolak proses login. Hal ini memastikan bahwa hanya akun yang telah diverifikasi yang dapat mengakses sistem.

Selain proses login, sistem juga mendukung proses registrasi pengguna baru. Pada tahap registrasi, kata sandi pengguna akan langsung dienkripsi menggunakan *BCryptPasswordEncoder* sebelum disimpan ke dalam basis data, serta akun akan diset dalam kondisi belum disetujui, sebagaimana ditunjukkan pada Kode 3.4.

```
1  public void registerNewUser(UserRegistrationDto userDto) {
2      User user = new User();
3      user.setUsername(userDto.getUsername());
4      user.setEmail(userDto.getEmail());
5      user.setPassword(passwordEncoder.encode(userDto.
    getPassword()));
6      user.setRole("USER");
7      user.setApproved(false);
8      userRepository.save(user);
9  }
10
11 public void updateUserPassword(User user, String newPassword
    ) {
12     user.setPassword(passwordEncoder.encode(newPassword));
13     userRepository.save(user);
14 }
```

Kode 3.4: Proses registrasi dan pembaruan password

Antarmuka login diimplementasikan menggunakan halaman *Login.html* yang berisi formulir sederhana untuk memasukkan *username* dan *password*. Data dari formulir ini dikirim ke endpoint */process-login* untuk diproses oleh Spring Security, sebagaimana ditunjukkan pada Kode 3.5.

```
<form th:action="@{/process-login}" method="post">
  <input id="username" name="username" type="text" required
   />
  <input id="password" name="password" type="password"
   required />
  <button type="submit">Login</button>
</form>
```

Kode 3.5: Form login pada halaman Login

Apabila proses autentikasi gagal, sistem akan menampilkan pesan kesalahan yang disesuaikan dengan jenis kegagalan, seperti kesalahan kredensial atau akun yang belum disetujui oleh admin. Mekanisme ini diatur melalui *CustomAuthenticationFailureHandler*, sebagaimana ditunjukkan pada Kode 3.6.

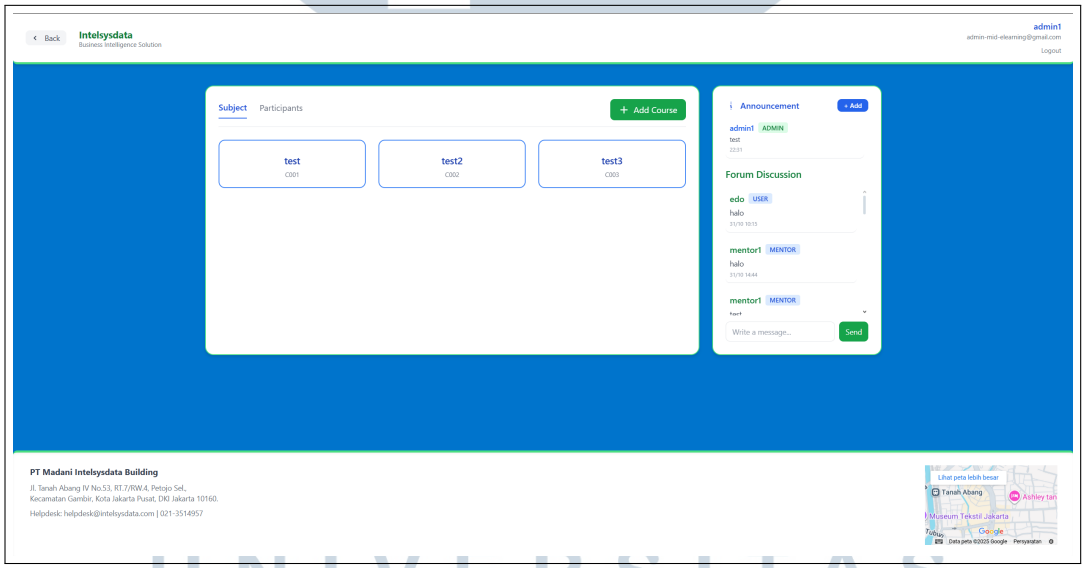
```
@Override
public void onAuthenticationFailure(HttpServletRequest
    request,
         HttpServletResponse response,
         AuthenticationException exception)
         throws IOException, ServletException {

     String errorMsg;
     if (exception instanceof DisabledException) {
         errorMsg = "Akun Anda belum disetujui oleh admin.";
     } else if (exception instanceof BadCredentialsException)
     {
         errorMsg = "Username atau password salah.";
     } else {
         errorMsg = "Gagal login. Silakan coba lagi.";
     }

     response.sendRedirect("/login?error=" +
         java.net.URLEncoder.encode(errorMsg, "UTF-8"));
}
```

Kode 3.6: Custom Authentication Failure Handler

Dengan kombinasi konfigurasi keamanan, pengelolaan peran pengguna, serta penanganan autentikasi yang terstruktur sebagaimana ditunjukkan pada Gambar 3.21 dan Kode 3.1 hingga Kode 3.6, halaman login berperan sebagai gerbang utama keamanan sistem MID E-learning. Implementasi ini memastikan integritas data, pengendalian akses yang ketat, serta pengalaman autentikasi yang aman dan terkelola secara profesional.

2. **Home / Subject List Admin**

Seperti yang bisa dilihat pada Gambar 3.22, halaman *Home / Subject List* pada sisi *Admin* berfungsi sebagai pusat pengelolaan mata pelajaran dalam sistem MID E-learning. Halaman ini menjadi titik awal bagi admin untuk mengakses seluruh mata pelajaran yang tersedia, serta melakukan pengelolaan awal terhadap struktur pembelajaran yang akan dijalankan di dalam sistem. Melalui halaman ini, admin dapat memilih mata pelajaran yang akan dikelola lebih lanjut dan memastikan ketersediaan konten pembelajaran secara terpusat.



Gambar 3.22. Halaman home / subject list (Admin)

Fungsionalitas pengumuman pada halaman ini diimplementasikan melalui mekanisme *backend* yang ditunjukkan pada Kode 3.7 hingga Kode 3.11. Pada Kode 3.7, *controller* menangani proses pengaksesan halaman detail mata pelajaran oleh admin sekaligus memuat data pendukung seperti tugas, pengumuman, dan diskusi forum. Selanjutnya, pada Kode 3.8, *controller* menangani proses penambahan pengumuman oleh admin dengan

melakukan validasi sederhana terhadap isi pesan sebelum disimpan ke dalam basis data melalui lapisan *service*. Proses penghapusan dan pengelolaan data pengumuman secara terstruktur diimplementasikan melalui *service layer* sebagaimana ditunjukkan pada Kode 3.9, sehingga konsistensi data tetap terjaga.

```
@GetMapping("/admin/course/{id}")
public String viewCourseDetails(@PathVariable("id") Long id,
    Model model) {
    Optional<Subject> subjectOpt = subjectService.
    getSubjectById(id);
    if (subjectOpt.isEmpty()) return "redirect:/admin/
    dashboard";
    Subject subject = subjectOpt.get();
    List<Assignment> assignments = assignmentService.
    getAssignmentsBySubjectId(id);
    model.addAttribute("subject", subject);
    model.addAttribute("assignments", assignments);
    model.addAttribute("announcements", announcementService.
    getAllAnnouncementsSorted());
    model.addAttribute("discussions", forumService.
    getAllDiscussions());
    return "admin/course-details";
}
```

Kode 3.7: Controller pengaksesan halaman course details oleh admin

Selain pengelolaan mata pelajaran, halaman ini juga terintegrasi dengan fitur pengumuman yang memungkinkan admin menyampaikan informasi penting secara langsung kepada pengguna. Proses penambahan pengumuman dilakukan melalui *controller* dengan validasi sederhana untuk memastikan bahwa konten tidak kosong sebelum disimpan ke dalam basis data, sebagaimana diimplementasikan pada Kode 3.8.

```
@PostMapping("/admin/announcement/add")
public String addAnnouncement(@RequestParam("content")
    String content, Principal principal) {
    if (content != null && !content.trim().isEmpty()) {
        Announcement a = new Announcement();
        a.setContent(content.trim());
        a.setUsername(principal.getName());
        a.setRole("ADMIN");
        announcementService.saveAnnouncement(a);
    }
```

```
    return "redirect:/admin/dashboard";
}
```

Kode 3.8: Controller penambahan pengumuman oleh admin

Data pengumuman yang tersimpan selanjutnya dikelola oleh *service layer*, yang bertugas mengambil seluruh pengumuman dan mengurutkannya berdasarkan waktu pembuatan terbaru, sebagaimana ditunjukkan pada Kode 3.9. Pendekatan ini memastikan informasi yang ditampilkan kepada pengguna selalu relevan dan mutakhir.

```
public List<Announcement> getAllAnnouncementsSorted() {
    return repo.findAllByOrderByCreatedAtDesc();
}
public void saveAnnouncement(Announcement a) { repo.save(a);
    }
public void deleteAnnouncement(Long id) { repo.deleteById(id
   ); }
```

Kode 3.9: Service pengelolaan data pengumuman

Penghapusan pengumuman dilakukan secara asinkron menggunakan *JavaScript* dan mekanisme *fetch API* yang dilengkapi dengan token CSRF, sebagaimana ditunjukkan pada Kode 3.10. Pendekatan ini memungkinkan admin menghapus pengumuman tanpa perlu melakukan pemuatan ulang halaman secara manual.

```
function deleteAnnouncement(id) {
  const csrfToken = document.querySelector('meta[name="_csrf
    "]').content;
  const csrfHeader = document.querySelector('meta[name="
    _csrf_header"]').content;
  fetch('/admin/announcement/' + id + '/delete', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json', [
    csrfHeader]: csrfToken }
  }).then(res => { if (res.ok) location.reload(); });
}
```

Kode 3.10: JavaScript penghapusan pengumuman

Selain pengumuman, halaman ini juga terhubung dengan fitur diskusi forum yang memungkinkan interaksi antar pengguna dalam sistem. Proses penyimpanan dan pengambilan pesan diskusi diimplementasikan melalui *service* dan *controller* sebagaimana ditunjukkan pada Kode 3.11,

sehingga pesan diskusi ditampilkan secara berurutan berdasarkan waktu pengiriman.

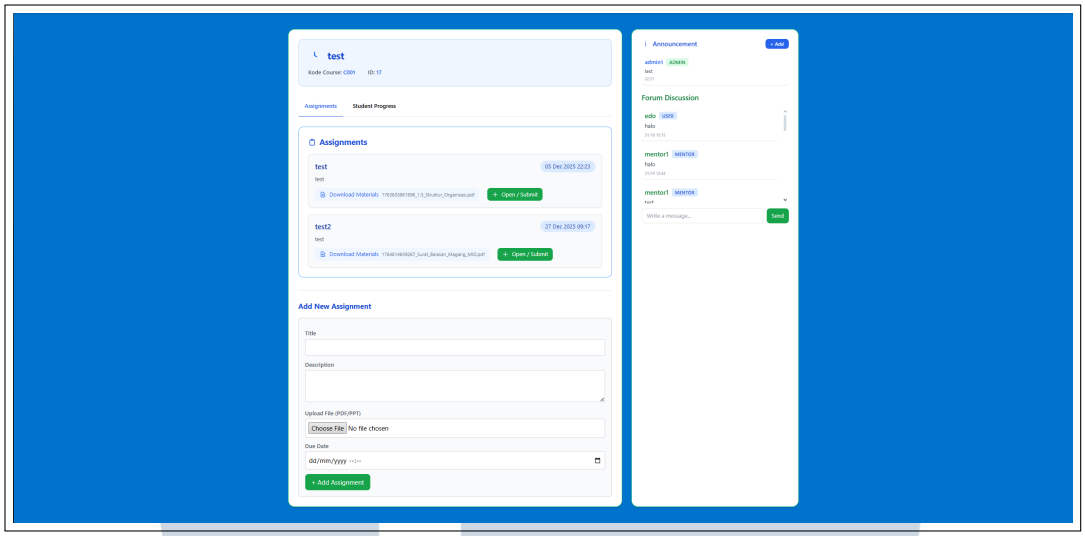
```
public List<Discussion> getAllDiscussions() {
    return discussionRepository.findAllByOrderByCreatedAtAsc
    ();
}
public void sendMessage(User sender, String message) {
    Discussion d = new Discussion();
    d.setSender(sender);
    d.setMessage(message);
    discussionRepository.save(d);
}
```

Kode 3.11: Service dan controller forum diskusi

Dengan integrasi antara halaman antarmuka, *controller*, *service*, dan basis data sebagaimana ditunjukkan pada Kode 3.7 hingga Kode 3.11, halaman *Home / Subject List Admin* berperan sebagai pusat kendali utama dalam pengelolaan pembelajaran. Seluruh aktivitas yang dilakukan admin pada halaman ini diproses secara konsisten di sisi backend dan disesuaikan dengan hak akses pengguna, sehingga mendukung penerapan pengelolaan pembelajaran berbasis peran (*role-based access control*) secara aman, terstruktur, dan berkelanjutan.

3. **Course Details Admin**

Halaman *Course Details* pada sisi *Admin* berfungsi sebagai pusat pengelolaan aktivitas pembelajaran untuk setiap mata pelajaran dalam sistem MID E-learning. Sebagaimana ditunjukkan pada Gambar 3.23, halaman ini menampilkan informasi detail mata pelajaran (*course*), daftar tugas (*assignment*), serta komponen pendukung pembelajaran seperti pengumuman dan diskusi forum yang terkait dalam satu tampilan terintegrasi.



Gambar 3.23. Halaman course details (Admin)

Secara teknis, halaman *Course Details Admin* diakses melalui endpoint backend yang dikelola oleh *controller*. Controller bertanggung jawab mengambil data mata pelajaran berdasarkan *ID*, memuat daftar tugas yang terasosiasi, serta mengirimkan seluruh data pendukung ke sisi tampilan (*view*). Implementasi proses tersebut ditunjukkan pada Kode 3.12.

```
@GetMapping("/{id}")
public String viewCourseDetails(@PathVariable("id") Long id,
    Model model) {
    Optional<Subject> subjectOpt = subjectService.
    getSubjectById(id);
    if (subjectOpt.isEmpty()) return "redirect:/admin/
    dashboard";

    Subject subject = subjectOpt.get();
    List<Assignment> assignments = assignmentService.
    getAssignmentsBySubjectId(id);

    model.addAttribute("subject", subject);
    model.addAttribute("assignments", assignments);
    model.addAttribute("user", userService.getCurrentUser())
    ;
    model.addAttribute("announcements", announcementService.
    getAllAnnouncementsSorted());
    model.addAttribute("discussions", forumService.
    getAllDiscussions());

```

```
15      return "admin/course-details";
16  }
```

Kode 3.12: Controller course details Admin

Controller tersebut memanfaatkan lapisan *service* untuk memisahkan logika bisnis dengan pengelolaan data. *SubjectService* digunakan untuk mengambil data mata pelajaran berdasarkan *ID*, sedangkan *AssignmentService* bertugas mengambil seluruh tugas yang terasosiasi dengan mata pelajaran tersebut. Implementasi metode pada lapisan service ditunjukkan pada Kode 3.13.

```
1  public List<Assignment> getAssignmentsBySubjectId(Long
       subjectId) {
2      return subjectRepository.findById(subjectId)
3              .map(assignmentRepository::findBySubject)
4              .orElse(List.of());
5  }
6
7  public Optional<Subject> getSubjectById(Long id) {
8      return subjectRepository.findById(id);
9  }
```

Kode 3.13: Service pengambilan subject dan assignment

Data yang telah diproses oleh controller selanjutnya ditampilkan pada sisi frontend menggunakan *template engine Thymeleaf*. Informasi mata pelajaran seperti nama, deskripsi, kode, dan ID ditampilkan pada bagian *course information*, sedangkan daftar tugas ditampilkan secara dinamis menggunakan struktur perulangan dengan validasi kondisi untuk menangani kondisi ketika belum terdapat tugas yang tersedia. Implementasi tampilan tersebut ditunjukkan pada Kode 3.14.

```
1  <div class="course-card">
2    <h1 th:text="${subject.name}">Course Name</h1>
3    <p th:text="${subject.description}">Course Description</p>
4    <p>Kode: <span th:text="${subject.code}">C001</span></p>
5    <p>ID: <span th:text="${subject.id}">1</span></p>
6  </div>
7
8  <div id="assignments-section">
9    <h2>Assignments</h2>
10
```

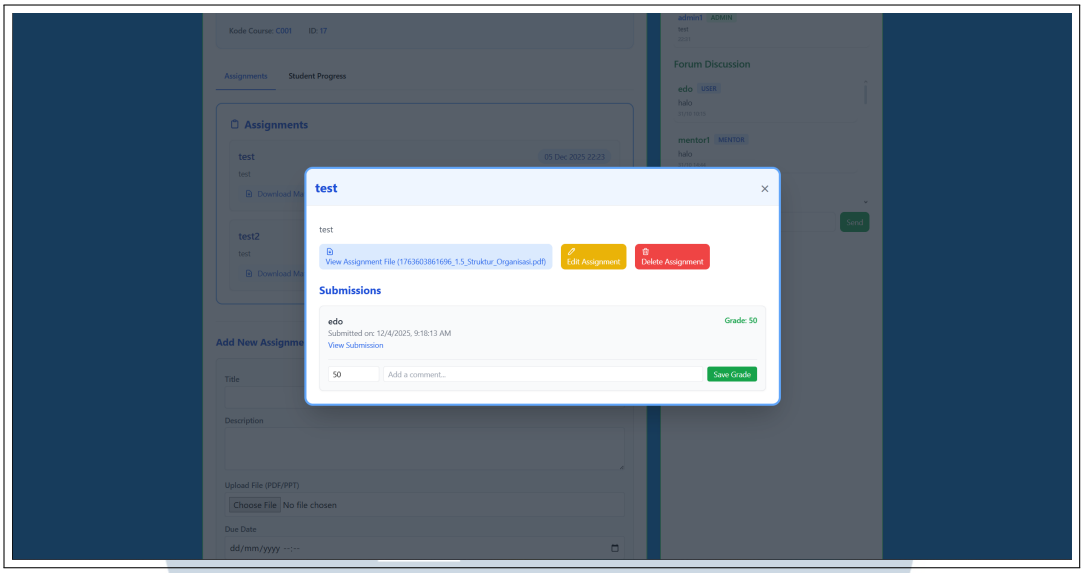
```
<div th:if="${assignments != null and !assignments.isEmpty()}">
  <div th:each="assignment : ${assignments}">
    <h3 th:text="${assignment.title}">Assignment Title</h3>
    <span th:text="${#temporals.format(assignment.dueDate, 'dd MMM yyyy HH:mm')}"></span>
    <p th:text="${assignment.description}"></p>

    <a th:if="${assignment.filePath != null}"
       th:href="@{/${assignment.filePath}}" download>
      Download <span th:text="${assignment.fileName}"></span>
    </a>
  </div>
</div>

<div th:if="${assignments == null or assignments.isEmpty()}">
  <p>No Assignments Yet</p>
</div>
</div>
```

Kode 3.14: Template course details dan assignment list

Melalui integrasi antara controller, service, dan template sebagaimana ditunjukkan pada Kode 3.12 hingga Kode 3.14, halaman *Course Details Admin* mampu menyajikan data mata pelajaran dan tugas secara terstruktur, konsisten, dan sinkron dengan basis data. Selain mendukung pengelolaan tugas, halaman ini juga berperan sebagai pusat koordinasi pembelajaran melalui integrasi pengumuman dan forum diskusi, sehingga admin memiliki kendali penuh terhadap jalannya proses pembelajaran dalam satu mata pelajaran secara terpusat dan berkelanjutan.

4. **Assignment Details Admin**

Halaman *Assignment Details* pada sisi *Admin* berfungsi sebagai sarana pengelolaan detail tugas dan proses penilaian dalam sistem MID E-learning. Sebagaimana ditunjukkan pada Gambar 3.24, halaman ini memungkinkan admin untuk melihat informasi lengkap suatu tugas, mengelola data tugas (edit dan hapus), serta melakukan penilaian terhadap hasil pengumpulan tugas peserta secara terstruktur dan terintegrasi dengan sistem backend.



Gambar 3.24. Halaman assignment details (Admin)

Secara arsitektural, halaman ini memanfaatkan pendekatan *hybrid rendering*, di mana data dasar halaman dimuat melalui *server-side rendering*, sedangkan detail tugas dan daftar pengumpulan (*submissions*) diambil secara dinamis menggunakan mekanisme *AJAX*. Untuk keperluan tersebut, sistem menyediakan endpoint khusus yang mengembalikan data *assignment* dalam format JSON, termasuk relasi data pengumpulan tugas peserta, sebagaimana ditunjukkan pada Kode 3.15.

```
@GetMapping("/course/{subjectId}/assignments/{assignmentId}")
@ResponseBody
@Transactional(readOnly = true)
public ResponseEntity<Assignment> getAssignmentDetailsJson(
        @PathVariable Long subjectId,
        @PathVariable Long assignmentId) {

    Optional<Assignment> assignmentOpt = assignmentService.getAssignmentById(assignmentId);
    if (assignmentOpt.isEmpty()) {
        return ResponseEntity.notFound().build();
    }

    Assignment assignment = assignmentOpt.get();
    if (!assignment.getSubject().getId().equals(subjectId)) {
        return ResponseEntity.notFound().build();
```

```
    }

    assignment.getSubmissions().size(); // force load jika LAZY
    return ResponseEntity.ok(assignment);
}
```

Kode 3.15: Controller pengambilan assignment sebagai JSON

Selain menampilkan detail tugas, halaman ini juga mendukung proses pengelolaan tugas melalui fitur edit dan hapus. Admin dapat membuka halaman formulir edit tugas untuk memperbarui informasi tugas, tenggat waktu (*due date*), maupun berkas pendukung. Proses pembaruan dan penghapusan data tersebut ditangani oleh controller dengan memanfaatkan lapisan service, sebagaimana ditunjukkan pada Kode 3.16.

```
@GetMapping("/assignment/{id}/edit")
public String showEditAssignmentForm(@PathVariable("id") Long assignmentId, Model model) {
    Optional<Assignment> assignmentOpt = assignmentService.getAssignmentById(assignmentId);
    if (assignmentOpt.isEmpty()) {
        return "redirect:/admin/dashboard";
    }
    model.addAttribute("assignment", assignmentOpt.get());
    return "admin/edit-assignment";
}

@PostMapping("/assignment/{id}/edit")
public String editAssignment(@PathVariable("id") Long assignmentId,
        @ModelAttribute("assignment") Assignment assignment,
        @RequestParam(value = "file", required = false) MultipartFile file,
        @RequestParam("dueDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME) LocalDateTime dueDate) {

    assignment.setDueDate(dueDate);
    assignmentService.updateAssignment(assignmentId, assignment, file);
    Long subjectId = assignmentService.getAssignmentSubjectId(assignmentId);
    return "redirect:/admin/course/" + subjectId;
}
```

```
22
23 @PostMapping("/assignment/{id}/delete")
24 public String deleteAssignment(@PathVariable("id") Long
       assignmentId) {
25     Long subjectId = assignmentService.
       getAssignmentSubjectId(assignmentId);
26     assignmentService.deleteAssignment(assignmentId);
27     return "redirect:/admin/course/" + subjectId;
28 }
```

Kode 3.16: Controller edit dan hapus assignment

Proses penilaian tugas dilakukan secara asinkron menggunakan *AJAX*. Admin dapat memasukkan nilai dan komentar untuk setiap pengumpulan tugas tanpa perlu melakukan pemuatan ulang halaman. Endpoint penilaian tersebut ditunjukkan pada Kode 3.17.

```
1  @PostMapping("/submission/{id}/grade")
2  @ResponseBody
3  public ResponseEntity<Map<String, Object>> gradeSubmission(
4          @PathVariable("id") Long submissionId,
5          @RequestParam("grade") String grade,
6          @RequestParam("comment") String comment) {
7
8      Map<String, Object> response = new HashMap<>();
9      submissionService.gradeSubmission(submissionId, grade,
       comment);
10     response.put("success", true);
11     return ResponseEntity.ok(response);
12 }
```

Kode 3.17: Controller penilaian submission

Logika bisnis terkait pengelolaan tugas dan pengumpulan tugas diimplementasikan pada lapisan service. Lapisan ini bertanggung jawab atas validasi data, penyimpanan nilai, serta penghapusan data apabila diperlukan, sebagaimana ditunjukkan pada Kode 3.18.

```
1 public Optional<Assignment> getAssignmentById(Long id) {
2     return assignmentRepository.findById(id);
3 }
4
5 public void gradeSubmission(Long submissionId, String grade,
      String comment) {
```

```
    Submission submission = submissionRepository.findById(
    submissionId)
        .orElseThrow(() -> new RuntimeException("Submission
    not found"));
    submission.setGrade(Double.parseDouble(grade));
    submission.setComment(comment);
    submissionRepository.save(submission);
}

public void deleteSubmissionById(Long submissionId) {
    Submission submission = submissionRepository.findById(
    submissionId)
        .orElseThrow(() -> new RuntimeException("Submission
    not found"));
    submissionRepository.delete(submission);
}
```

Kode 3.18: Service assignment dan submission

Pada sisi *client-side*, JavaScript digunakan untuk mengambil detail tugas, menampilkan data ke dalam modal, merender daftar pengumpulan tugas, serta mengirimkan nilai dan komentar melalui *AJAX*. Implementasi mekanisme tersebut ditunjukkan pada Kode 3.19.

```
async function fetchAssignmentDetails(id) {
  const response = await fetch(`/admin/course/${subjectId}/
    assignments/${id}`);
  currentAssignment = await response.json();
  populateViewModal(currentAssignment);
}

document.querySelectorAll('.grade-form').forEach(form => {
  form.addEventListener('submit', async (event) => {
    event.preventDefault();
    const submissionId = form.dataset.submissionId;
    const grade = form.querySelector('[name="grade"]').value
    ;
    const comment = form.querySelector('[name="comment"]').
    value;

    await fetch(`/admin/submission/${submissionId}/grade`, {
      method: 'POST',
      headers: { 'Content-Type': 'application/x-www-form-
    urlencoded' },
```

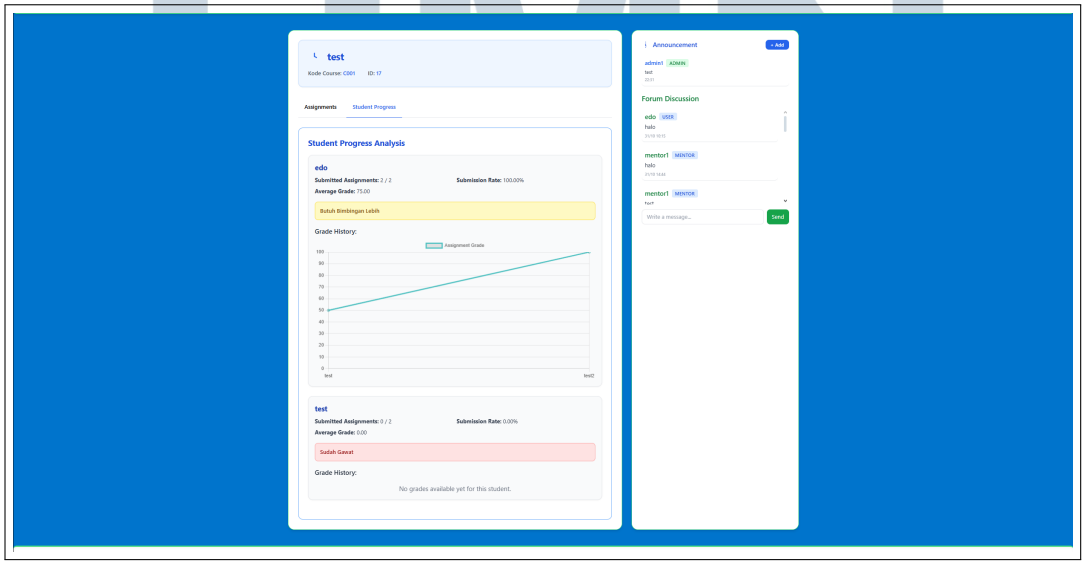
```
      body: new URLSearchParams({ grade, comment })
    });

    fetchAssignmentDetails(currentAssignment.id);
  });
});
```

Kode 3.19: AJAX fetch assignment dan proses penilaian

Dengan integrasi antara controller, service, dan JavaScript sebagaimana ditunjukkan pada Kode 3.15 hingga Kode 3.19, halaman *Assignment Details Admin* mampu mendukung proses evaluasi pembelajaran secara menyeluruh. Admin dapat mengelola tugas, menilai hasil pengumpulan, serta memastikan setiap perubahan data tersimpan dan tersinkronisasi secara real-time, sehingga mendukung pengelolaan pembelajaran yang efi

5. **Student Progress Admin**

Halaman *Student Progress* pada sisi *Admin* berfungsi sebagai pusat pemantauan performa peserta dalam suatu mata pelajaran atau kelas. Sebagaimana ditunjukkan pada Gambar 3.25, halaman ini memungkinkan admin untuk meninjau tingkat partisipasi peserta, progres pengumpulan tugas, serta kualitas hasil belajar secara individual berdasarkan data yang dihitung dan disajikan oleh sistem secara terstruktur.



Gambar 3.25. Halaman student progress (Admin)

Secara konseptual, data progres peserta dihitung pada lapisan *Service* dengan mengombinasikan informasi mata pelajaran, daftar tugas, data pengumpulan tugas (*submissions*), serta nilai yang telah diberikan. Perhitungan ini menghasilkan indikator utama berupa jumlah tugas yang dikumpulkan, persentase pengumpulan, rata-rata nilai, status performa, serta riwayat nilai untuk keperluan visualisasi grafik, sebagaimana ditunjukkan pada Kode 3.20.

```
@Transactional(readOnly = true)
public List<StudentProgressDTO> getStudentProgressForCourse(Long courseId) {
    Optional<Subject> subjectOptional = subjectRepository.findById(courseId);
    if (subjectOptional.isEmpty()) return Collections.emptyList();

    List<User> students = userRepository.findByRole("USER");
    List<Assignment> courseAssignments = assignmentRepository.findBySubjectId(courseId);
    List<StudentProgressDTO> studentProgressList = new ArrayList<>();

    for (User student : students) {
        int submittedAssignments = 0;
        int totalAssignments = courseAssignments.size();
        double totalGrades = 0;
        int gradedAssignmentsCount = 0;
        List<GradeChartDataDTO> gradeHistory = new ArrayList<>();

        for (Assignment assignment : courseAssignments) {
            Optional<Submission> submissionOptional =
                submissionRepository.findByAssignmentIdAndUserId(
                    assignment.getId(), student.getId());

            if (submissionOptional.isPresent()) {
                submittedAssignments++;
                Submission submission = submissionOptional.get();
                if (submission.getGrade() != null) {
                    totalGrades += submission.getGrade();
                    gradedAssignmentsCount++;
```

```
                    gradeHistory.add(
                        new GradeChartDataDTO(
                            assignment.getTitle(),
submission.getGrade()));
                }
            }
        }

        double submissionPercentage =
            (totalAssignments > 0)
            ? ((double) submittedAssignments /
totalAssignments) * 100
            : 100;

        double averageGrade =
            (gradedAssignmentsCount > 0)
            ? totalGrades / gradedAssignmentsCount
            : 0;

        boolean hasSubmissionProblem = submissionPercentage
< 50;
        boolean hasGradeProblem = averageGrade < 75;

        String progressStatus;
        String statusColor;
        if (averageGrade > 75) {
            progressStatus = "Bagus";
            statusColor = "green";
        } else if (averageGrade >= 65) {
            progressStatus = "Butuh Bimbingan Lebih";
            statusColor = "yellow";
        } else {
            progressStatus = "Sudah Gawat";
            statusColor = "red";
        }

        studentProgressList.add(new StudentProgressDTO(
            student.getId(),
            student.getUsername(),
            submittedAssignments,
            totalAssignments,
            submissionPercentage,
            averageGrade,
```

```
            hasSubmissionProblem,
            hasGradeProblem,
            gradeHistory,
            progressStatus,
            statusColor
        ));
    }

    return studentProgressList;
}
```

Kode 3.20: Service perhitungan progres peserta

Data progres yang telah dihitung kemudian diekspos melalui endpoint REST API khusus agar dapat digunakan secara dinamis oleh sisi frontend. Endpoint ini diamankan menggunakan mekanisme otorisasi berbasis peran (*role-based access control*), sebagaimana ditunjukkan pada Kode 3.21.

```
@RestController
@RequestMapping("/admin/api")
@PreAuthorize("hasRole('ADMIN')")
public class AdminStudentProgressApiController {

    private final StudentProgressService
    studentProgressService;

    @GetMapping("/courses/{courseId}/student-progress")
    public List<StudentProgressDTO> getStudentProgress(
            @PathVariable Long courseId) {
        return studentProgressService
                .getStudentProgressForCourse(courseId);
    }
}
```

Kode 3.21: API controller student progress

Selain melalui API, halaman *Student Progress* juga dapat dirender secara *server-side* menggunakan Controller utama untuk menampilkan tabel ringkasan progres peserta. Proses ini ditunjukkan pada Kode 3.22.

```
@GetMapping("/course/{courseId}/progress")
public String showStudentProgress(
        @PathVariable("courseId") Long courseId,
        Model model) {
```

```
5
6       Subject subject = subjectService.getSubjectById(courseId
        )
7           .orElseThrow(() ->
8               new RuntimeException("Course not found"));
9
10      List<StudentProgressDTO> studentProgress =
11          studentProgressService.getStudentProgressForCourse(
        courseId);
12
13      model.addAttribute("subject", subject);
14      model.addAttribute("studentProgress", studentProgress);
15      return "admin/student-progress";
16  }
```

Kode 3.22: Controller render halaman student progress

Pada sisi tampilan, data progres peserta ditampilkan dalam bentuk tabel yang memuat informasi persentase pengumpulan tugas, rata-rata nilai, serta indikator permasalahan performa. Struktur tampilan tersebut ditunjukkan pada Kode 3.23.

```
1  <table class="table">
2    <thead>
3      <tr>
4        <th>Student</th>
5        <th>Submission \%</th>
6        <th>Average Grade</th>
7        <th>Problems</th>
8        <th>Grade History</th>
9      </tr>
10   </thead>
11   <tbody>
12     <tr th:each="progress : ${studentProgress}">
13       <td th:text="${progress.username}"></td>
14       <td th:text="${progress.submissionPercentage} + '%'"><
       /td>
15       <td th:text="${progress.averageGrade}"></td>
16       <td>
17         <span th:if="${progress.hasSubmissionProblem}">
18           Low Submission
19         </span>
20         <span th:if="${progress.hasGradeProblem}">
21           Low Grade
```

```
        </span>
      </td>
      <td>
        <canvas th:id="'chart-' + ${progress.userId}"
                width="400" height="100"></canvas>
      </td>
    </tr>
  </tbody>
</table>
```

Kode 3.23: Template student progress

Untuk visualisasi grafik nilai, data progres peserta dapat diambil menggunakan *AJAX* dan dirender secara dinamis pada sisi klien. Implementasi mekanisme tersebut ditunjukkan pada Kode 3.24.

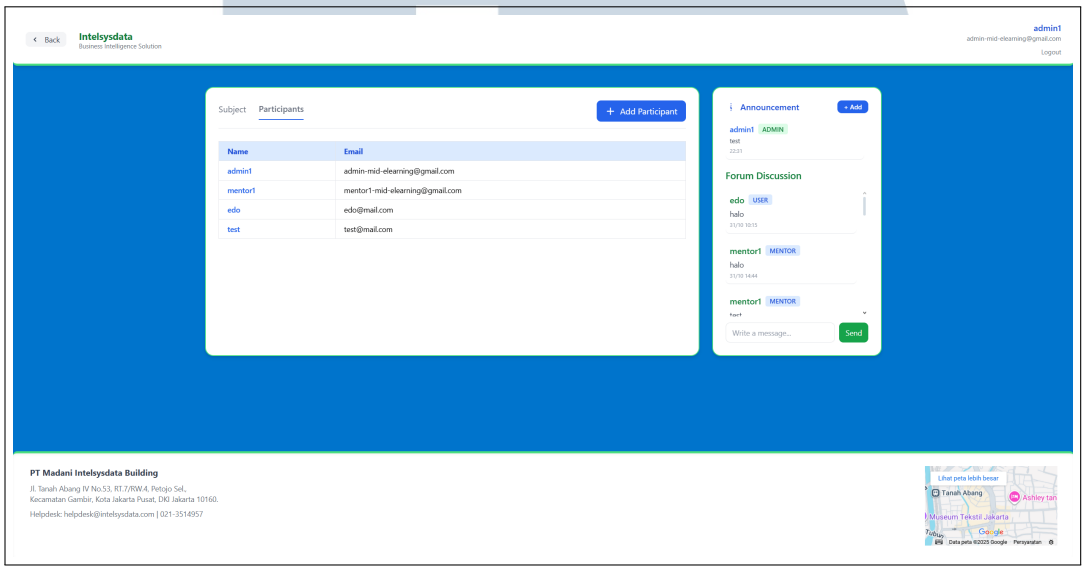
```
async function loadStudentProgressData(subjectId) {
  const res = await fetch(
    `/admin/api/courses/${subjectId}/student-progress`);
  const data = await res.json();
  renderStudentProgress(data);
}

function renderStudentProgress(data) {
  data.forEach(student => {
    const labels =
      student.gradeHistory.map(g => g.assignmentTitle);
    const values =
      student.gradeHistory.map(g => g.grade);

    // inisialisasi Chart.js
  });
}
```

Kode 3.24: AJAX load student progress dan render grafik

Dengan dukungan Gambar 3.25 serta integrasi kode pada Kode 3.20 hingga Kode 3.24, halaman *Student Progress Admin* memungkinkan admin melakukan pemantauan performa peserta secara komprehensif berbasis data. Informasi kuantitatif, indikator status, dan visualisasi grafik membantu admin dalam melakukan evaluasi serta intervensi pembelajaran secara tepat sasaran dan berkelanjutan.

6. **Participants List Admin**

Halaman *Participants List* pada sisi *Admin* berfungsi sebagai pusat pengelolaan data peserta dalam sistem MID E-learning. Sebagaimana ditunjukkan pada Gambar 3.26, halaman ini memungkinkan admin untuk melihat seluruh daftar pengguna yang terdaftar di dalam sistem, termasuk peserta (*user*), mentor, maupun admin, serta melakukan penambahan peserta baru secara langsung melalui antarmuka yang disediakan.



Gambar 3.26. Halaman participants list (Admin)

Secara teknis, halaman *Participants List* dirender melalui *Controller* pada sisi backend yang bertugas mengambil data seluruh pengguna dari sistem. Proses pengambilan dan pengiriman data peserta ke sisi tampilan ditangani oleh controller sebagaimana ditunjukkan pada Kode 3.25.

```
@GetMapping("/participants")
public String showParticipants(Model model) {
    model.addAttribute("students", userService.getAllUsers()
    );
    return "admin/participants";
}
```

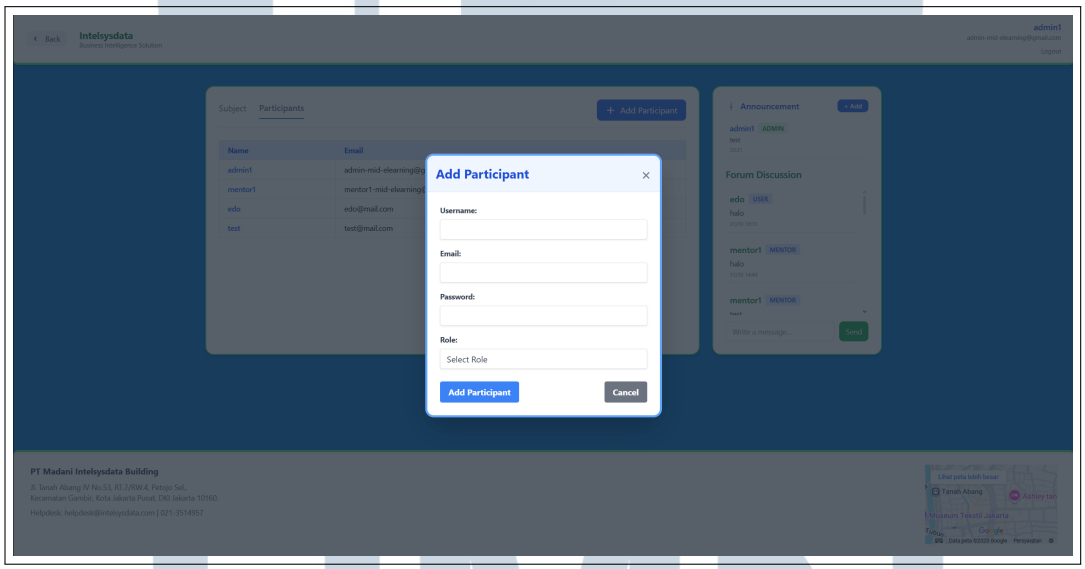
Kode 3.25: Controller participants list admin

Controller tersebut memanfaatkan *UserService* untuk mengambil data pengguna secara terpusat, sehingga proses pengelolaan data peserta tetap mengikuti prinsip pemisahan tanggung jawab antara lapisan kontrol dan logika bisnis. Seluruh data pengguna yang diambil kemudian

ditampilkan pada halaman *participants* dalam bentuk tabel, sesuai dengan struktur yang ditampilkan pada Gambar 3.26.

7. **Add Participant Admin**

Halaman *Add Participant* pada sisi *Admin* berfungsi sebagai antarmuka input untuk menambahkan peserta baru ke dalam sistem MID E-learning. Sebagaimana ditunjukkan pada Gambar 3.27, melalui halaman ini admin dapat membuat akun pengguna secara langsung dengan menentukan identitas dasar serta peran (*role*) peserta yang akan menentukan hak akses dalam sistem.



Gambar 3.27. Form add participant (Admin)

Secara konseptual, proses penambahan peserta diawali dengan pengisian formulir yang divalidasi menggunakan *Data Transfer Object* (DTO). Struktur DTO yang digunakan untuk memastikan validitas data masukan ditunjukkan pada Kode 3.26.

```
public class ParticipantRegistrationDto {

    @NotBlank(message = "Username tidak boleh kosong")
    @Size(min = 3, max = 50, message = "Username harus
   antara 3-50 karakter")
    private String username;

    @NotBlank(message = "Email tidak boleh kosong")
    @Email(message = "Format email tidak valid")
```

```
    private String email;

    @NotBlank(message = "Password tidak boleh kosong")
    @Size(min = 6, message = "Password minimal 6 karakter")
    private String password;

    private String role;
}
```

Kode 3.26: DTO registrasi peserta

Halaman *Add Participant* ditampilkan melalui metode *GET* pada Controller, sedangkan proses penyimpanan data peserta dilakukan melalui metode *POST*. Implementasi Controller yang menangani kedua proses tersebut ditunjukkan pada Kode 3.27.

```
@GetMapping("/participant/add")
public String showAddParticipantPage(Model model) {
    model.addAttribute("participant", new
    ParticipantRegistrationDto());
    return "admin/add-participant";
}

@PostMapping("/participant/add")
@ResponseBody
public ResponseEntity<Map<String, Object>> addParticipant(
        @Valid @ModelAttribute("participant")
        ParticipantRegistrationDto participantDto,
        BindingResult result) {

    if (result.hasErrors()) {
        String errorMessage = result.getFieldErrors().stream
    ()
                .map(e -> e.getField() + ": " + e.
    getDefaultMessage())
                .findFirst()
                .orElse("Invalid data provided.");
        return ResponseEntity.badRequest()
                .body(Map.of("status", "error", "message",
    errorMessage));
    }

    if (userRepository.findByUsername(
            participantDto.getUsername()).isPresent()) {
```

```
        return ResponseEntity.badRequest()
                .body(Map.of("status", "error",
                             "message", "Username already
    exists!"));
    }

    if (userRepository.findByEmail(
            participantDto.getEmail()).isPresent()) {
        return ResponseEntity.badRequest()
                .body(Map.of("status", "error",
                             "message", "Email already
    exists!"));
    }

    userService.registerParticipant(participantDto);
    return ResponseEntity.ok(
        Map.of("status", "success",
               "message", "Participant added successfully!")
    );
}
```

Kode 3.27: Controller add participant admin

Untuk menjaga konsistensi data dan keamanan sistem, proses registrasi peserta diproses pada lapisan *Service*. Implementasi logika bisnis untuk enkripsi kata sandi dan penetapan peran pengguna ditunjukkan pada Kode 3.28.

```
public void registerParticipant(
        ParticipantRegistrationDto participantDto) {

    User user = new User();
    user.setUsername(participantDto.getUsername());
    user.setEmail(participantDto.getEmail());
    user.setPassword(
        passwordEncoder.encode(
            participantDto.getPassword()));
    user.setRole(
        participantDto.getRole() != null
        ? participantDto.getRole()
        : "USER");

    userRepository.save(user);
```

```
}
```

Kode 3.28: Service registrasi peserta

Sebelum proses penyimpanan dilakukan, sistem memanfaatkan metode pada lapisan *Repository* untuk memastikan bahwa data *username* dan *email* belum terdaftar sebelumnya. Mekanisme validasi ini ditunjukkan pada Kode 3.29.

```
Optional<User> findByUsername(String username);
Optional<User> findByEmail(String email);
```

Kode 3.29: Repository validasi pengguna

Pada sisi tampilan, formulir *Add Participant* disajikan dalam bentuk *modal* yang dilengkapi dengan mekanisme pengiriman data menggunakan *AJAX*. Implementasi formulir dan skrip pengiriman data tersebut ditunjukkan pada Kode 3.30.

```
<form id="add-participant-form"
      th:action="@{/admin/participant/add}"
      method="post">
  <input type="hidden"
         th:name="${_csrf.parameterName}"
         th:value="${_csrf.token}" />

  <input type="text" name="username" required />
  <input type="email" name="email" required />
  <input type="password" name="password" required />

  <select name="role" required>
    <option value="">Select Role</option>
    <option value="STUDENT">Student</option>
    <option value="MENTOR">Mentor</option>
  </select>

  <button type="submit">Add Participant</button>
</form>

<script>
document.addEventListener('DOMContentLoaded', () => {
  const form = document.getElementById('add-participant-form
    ');
  form.addEventListener('submit', e => {
    e.preventDefault();
```

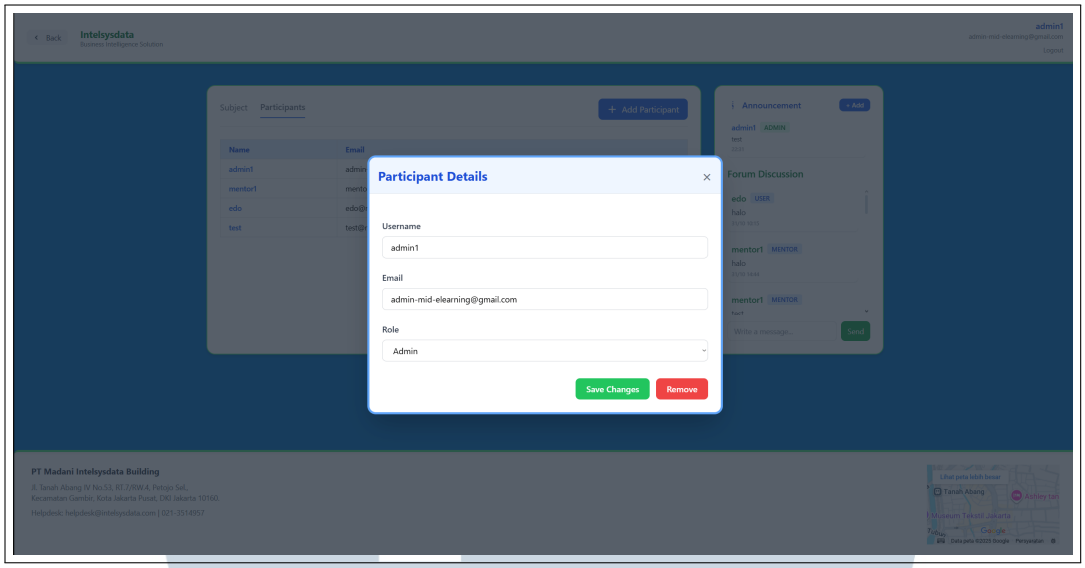
```
    const fd = new FormData(form);
    fetch(form.action, { method: 'POST', body: fd })
      .then(r => r.json())
      .then(data => {
        if (data.status === 'success') {
          Swal.fire('Success', data.message, 'success')
              .then(() => location.reload());
        } else {
          Swal.fire('Error', data.message, 'error');
        }
      });
  });
});
</script>
```

Kode 3.30: Form add participant dan AJAX

Dengan demikian, halaman *Add Participant Admin* sebagaimana ditunjukkan pada Gambar 3.27, didukung oleh rangkaian proses backend dan frontend yang direpresentasikan pada Kode 3.26 hingga Kode 3.30, sehingga proses penambahan akun pengguna dapat dilakukan secara aman, terkontrol, dan terintegrasi.

8. **Participant Details Admin**

Halaman *Participant Details* pada sisi *Admin* berfungsi sebagai antarmuka untuk menampilkan serta mengelola detail informasi akun peserta yang telah terdaftar dalam sistem MID E-learning. Melalui halaman ini, admin dapat melakukan peninjauan data identitas peserta serta memperbarui peran (*role*) akun guna menyesuaikan hak akses pengguna dalam sistem. Tampilan halaman *Participant Details Admin* dapat dilihat pada Gambar 3.28.



Gambar 3.28. Halaman participant details (Admin)

Secara konseptual, halaman ini bekerja dengan mekanisme pengambilan data peserta secara dinamis berdasarkan *ID* yang dipilih oleh admin. Data peserta diambil melalui *REST API* pada sisi server dan ditampilkan dalam sebuah jendela *modal*, sehingga admin dapat melakukan pengelolaan data tanpa perlu berpindah halaman.

Pengambilan data detail peserta dilakukan melalui *API Controller* yang hanya dapat diakses oleh pengguna dengan peran *ADMIN*, sebagaimana ditunjukkan pada Kode 3.31. API ini bertugas mengembalikan data peserta berdasarkan *ID* yang diterima dari sisi klien.

```
@RestController
@RequestMapping("/api/admin/participants")
@PreAuthorize("hasRole('ADMIN')")
public class AdminParticipantApiController {

    private final UserService userService;

    public AdminParticipantApiController(UserService
    userService) {
        this.userService = userService;
    }

    @GetMapping("/{id}")
    public ResponseEntity<User> getParticipant(@PathVariable
     Long id) {
```

```
        return userService.getUserById(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }
}
```

Kode 3.31: API pengambilan data participant

Pada sisi klien, proses pengambilan data dilakukan menggunakan JavaScript melalui fungsi *fetchParticipantDetails*, seperti ditunjukkan pada Kode 3.32. Fungsi ini melakukan permintaan data ke API, kemudian mengisikan data *username*, *email*, dan *role* ke dalam field formulir pada jendela *modal*. Selain itu, atribut *action* pada formulir juga disesuaikan secara dinamis berdasarkan *ID* peserta yang dipilih.

```
function fetchParticipantDetails(id) {
  fetch(`/api/admin/participants/${id}`)
    .then(response => {
      if (!response.ok)
        throw new Error('Network response was not ok');
      return response.json();
    })
    .then(participant => {
      document.getElementById('modal-username').value =
          participant.username || '';
      document.getElementById('modal-email').value =
          participant.email || '';
      document.getElementById('modal-role').value =
          participant.role || 'USER';

      const editForm =
          document.getElementById('editParticipantForm');
      if (editForm)
        editForm.action =
          `/admin/participant/${participant.id}/update`;

      const modal =
          document.getElementById('participant-details-modal
    ');
      if (modal) modal.classList.remove('hidden');
    })
    .catch(() => {
      alert('Gagal memuat detail peserta.');
    });
```

```
}
```

Kode 3.32: AJAX pengambilan dan pengisian data participant

Jendela *modal* yang ditampilkan berisi formulir *Edit Participant*, sebagaimana ditunjukkan pada Kode 3.33, yang memungkinkan admin melakukan perubahan data peserta. Formulir ini mencakup field *Username*, *Email*, serta *Role*, dan telah dilengkapi dengan *CSRF token* untuk menjaga keamanan proses pengiriman data.

```
<form id="editParticipantForm" action="#" method="post">
  <input type="hidden"
         th:name="${_csrf.parameterName}"
         th:value="${_csrf.token}" />

  <div>
    <label>Username</label>
    <input type="text"
           id="modal-username"
           name="username" />
  </div>

  <div>
    <label>Email</label>
    <input type="email"
           id="modal-email"
           name="email" />
  </div>

  <div>
    <label>Role</label>
    <select id="modal-role" name="role">
      <option value="USER">User</option>
      <option value="MENTOR">Mentor</option>
      <option value="ADMIN">Admin</option>
    </select>
  </div>

  <button type="submit">Save Changes</button>
</form>
```

Kode 3.33: Form edit participant

Proses penyimpanan perubahan data peserta dilakukan melalui metode *POST* pada *AdminDashboardController*, seperti ditunjukkan pada

Kode 3.34. Controller ini bertugas memperbarui data pengguna di dalam basis data serta memastikan perubahan peran tersimpan secara konsisten.

```
@PostMapping("/participant/{id}/update")
public String updateParticipant(
        @PathVariable Long id,
        @ModelAttribute("participant") User updatedUser,
        @RequestParam("role") String role) {

    Optional<User> existingOpt =
        userService.getUserById(id);

    if (existingOpt.isPresent()) {
        User existing = existingOpt.get();
        existing.setUsername(updatedUser.getUsername());
        existing.setEmail(updatedUser.getEmail());
        existing.setRole(role);
        userService.saveUser(existing);
    }
    return "redirect:/admin/dashboard";
}
```

Kode 3.34: Controller update participant

Dengan adanya halaman *Participant Details Admin* ini, admin dapat mengelola akun peserta secara fleksibel dan terkontrol. Integrasi antara API, AJAX, dan controller server-side memastikan bahwa perubahan data dilakukan secara aman, konsisten, dan efisien, sehingga struktur peran serta keamanan akses dalam sistem MID E-learning dapat terjaga dengan baik. Gambar 3.28 dan Kode 3.31 sampai Kode 3.34 merepresentasikan keseluruhan alur pengelolaan data peserta pada halaman ini.

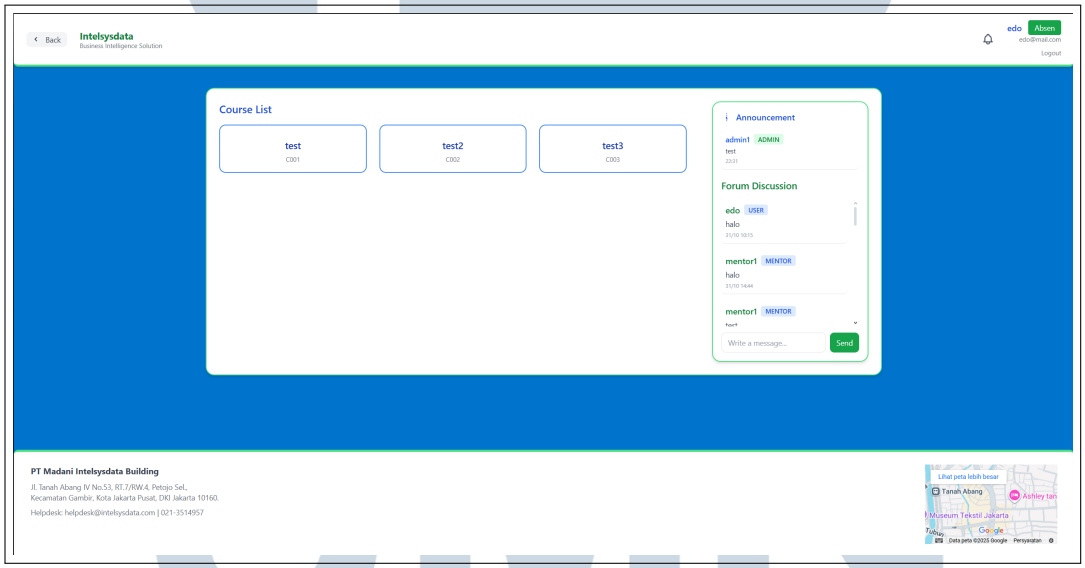
B Student

Bagian ini membahas realisasi fungsional sistem MID E-learning dari sisi pengguna dengan hak akses administratif dan pembimbing. Pada tahap ini, sistem diimplementasikan untuk mendukung aktivitas pengelolaan pembelajaran secara menyeluruh, mulai dari proses autentikasi, manajemen mata pelajaran, pengelolaan peserta, hingga pemantauan progres dan evaluasi hasil belajar. Implementasi ini dirancang dengan menerapkan pengendalian akses berbasis peran (*role-based access control*) sehingga setiap fungsi yang tersedia hanya dapat diakses sesuai

dengan kewenangan masing-masing peran, baik sebagai admin maupun mentor. Dengan demikian, implementasi pada sisi *Admin/Mentor* memastikan bahwa proses administrasi, koordinasi, dan pengawasan pembelajaran dapat berjalan secara terstruktur, aman, dan terintegrasi dalam sistem MID E-learning.

1. **Home / Dashboard Student**

   Halaman *Home / Dashboard Student* merupakan halaman utama yang ditampilkan setelah pengguna dengan peran *student* berhasil masuk ke dalam sistem MID E-learning. Halaman ini berfungsi sebagai pusat akses pembelajaran bagi student dengan menampilkan daftar mata pelajaran yang tersedia, serta menyediakan panel komunikasi berupa pengumuman dan forum diskusi. Tampilan halaman *Home / Dashboard Student* ditunjukkan pada Gambar 3.29.



Gambar 3.29. Home page student

Pada bagian utama halaman, sistem menampilkan daftar mata pelajaran dalam bentuk kartu. Setiap kartu merepresentasikan satu mata pelajaran yang telah dibuat oleh admin atau mentor, serta menampilkan informasi berupa nama dan kode mata pelajaran. Student dapat mengakses halaman detail mata pelajaran dengan menekan nama mata pelajaran yang tersedia pada kartu tersebut.

Pada sisi kanan halaman, ditampilkan panel komunikasi yang terdiri dari pengumuman dan forum diskusi. Pengumuman disajikan secara kronologis berdasarkan waktu terbaru, sedangkan forum diskusi

memungkinkan terjadinya interaksi antara student, mentor, dan admin melalui pesan teks.

Ketika halaman dashboard student diakses, sistem memanfaatkan informasi autentikasi untuk memperoleh data pengguna yang sedang login. Data tersebut kemudian digunakan untuk memuat daftar mata pelajaran, forum diskusi, serta pengumuman sebelum seluruh data dikirimkan ke sisi tampilan, sebagaimana ditunjukkan pada Kode 3.35.

```
@GetMapping("/user/dashboard")
public String userDashboard(Authentication authentication,
    Model model) {
     String username = authentication.getName();
     User user = userService.getUserByUsername(username)
             .orElseThrow(() -> new RuntimeException("User
    not found: " + username));

     model.addAttribute("user", user);
     model.addAttribute("subjects", subjectService.
    getAllSubjects());
     model.addAttribute("discussions", forumService.
    getAllDiscussions());
     model.addAttribute("announcements", announcementService.
    getAllAnnouncementsSorted());

     return "user/dashboard";
}
```

Kode 3.35: Student dashboard controller

Berdasarkan implementasi controller tersebut, sistem terlebih dahulu memperoleh identitas pengguna melalui sesi autentikasi. Selanjutnya, data mata pelajaran dimuat melalui lapisan service, sedangkan data pengumuman dan forum diskusi diambil sebagai informasi pendukung untuk ditampilkan pada dashboard student.

Data mata pelajaran yang ditampilkan pada halaman ini diperoleh melalui metode service yang bertugas mengambil seluruh data mata pelajaran dari basis data dengan mekanisme transaksi baca saja, sehingga tidak memengaruhi integritas data yang tersimpan, sebagaimana ditunjukkan pada Kode 3.36.

```
@Transactional(readOnly = true)
public List<Subject> getAllSubjects() {
```

```
    return subjectRepository.findAll();
}
```

Kode 3.36: Get all subjects service

Pada sisi tampilan, halaman dashboard student menggunakan template berbasis Thymeleaf untuk menampilkan daftar mata pelajaran secara dinamis. Proses iterasi dilakukan terhadap data mata pelajaran yang dikirimkan oleh controller, sehingga setiap mata pelajaran dapat ditampilkan dalam bentuk kartu dengan animasi bertahap, seperti ditunjukkan pada Kode 3.37.

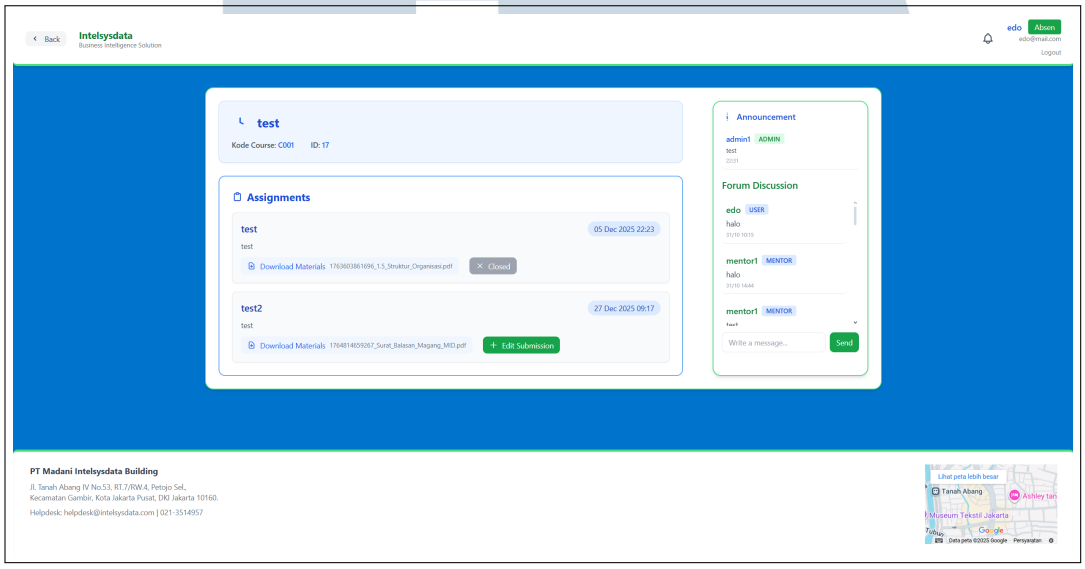
```
<h2 class="text-xl font-semibold mb-4 text-blue-700">Course List</h2>
<div class="grid md:grid-cols-3 sm:grid-cols-2 grid-cols-1 gap-6">
  <div th:each="s, stat : ${subjects}"
       th:style="'animation-delay: ' + ${stat.index * 0.1} + 's'"
       class="course-card animate-slide-down border-2 border-blue-500 rounded-xl text-center p-4 relative hover:bg-blue-50">
    <a th:href="@{'/user/course-details/' + ${s.id}}"
       class="block text-blue-800 font-semibold text-lg hover:underline mt-2">
      <span th:text="${s.name}">Course Name</span>
    </a>
    <p class="text-xs text-gray-500 mt-1" th:text="${s.code}">C001</p>
  </div>
</div>
```

Kode 3.37: Course list on student dashboard

Dengan adanya halaman *Home / Dashboard Student* ini, student memperoleh akses terpusat terhadap mata pelajaran, pengumuman, dan forum diskusi. Integrasi antara controller, service, dan template tampilan memastikan informasi pembelajaran dapat diakses secara terstruktur dan efisien, sehingga mendukung kelancaran proses pembelajaran daring dalam sistem MID E-learning. Gambar 3.29 serta Kode 3.35–3.37 merepresentasikan keseluruhan alur dan implementasi halaman dashboard student.

2. **Course Details Student**

Halaman *Course Details* pada sisi *Student* berfungsi sebagai antarmuka utama bagi peserta untuk melihat detail suatu mata pelajaran yang diikuti beserta daftar tugas yang tersedia. Halaman ini memungkinkan peserta mengakses informasi umum mata pelajaran serta memantau status pengumpulan tugas secara terstruktur dan berbasis waktu, sebagaimana ditunjukkan pada Gambar 3.30.



Gambar 3.30. Halaman course details student

Secara konseptual, halaman ini memanfaatkan pemrosesan data pada sisi server untuk membangun daftar tugas yang telah dikombinasikan dengan data pengumpulan milik peserta yang sedang login. Pendekatan ini memungkinkan sistem menampilkan kondisi setiap tugas secara kontekstual, termasuk status pengumpulan, tenggat waktu, serta keberadaan pengumpulan sebelumnya.

Pengolahan data utama dilakukan pada *controller StudentDashboardController* melalui endpoint *GET* dengan parameter *ID* mata pelajaran. Pada tahap awal, sistem mengambil data akun peserta berdasarkan sesi autentikasi aktif, kemudian memuat data mata pelajaran sesuai dengan *ID* yang dipilih. Proses ini ditunjukkan pada Kode 3.38.

```
@GetMapping("/user/course-details/{id}")
public String courseDetails(@PathVariable("id") Long id,
        Authentication authentication, Model model) {

```

```
    String username = authentication.getName();
    User user = userService.getUserByUsername(username)
        .orElseThrow(() -> new RuntimeException("User not
    found"));

    model.addAttribute("user", user);

    Subject subject = subjectService.getSubjectById(id)
        .orElseThrow(() -> new RuntimeException("Course not
    found"));

    model.addAttribute("subject", subject);

    List<Assignment> assignments =
        assignmentService.getAssignmentsBySubject(subject);

    List<AssignmentWithSubmissionDTO>
    assignmentsWithSubmission =
        assignments.stream()
            .map(assignment -> {
                Optional<Submission> userSubmission =
                    submissionService
                    .getSubmissionByAssignmentAndStudent(
                        assignment, user);
                return new AssignmentWithSubmissionDTO(
                    assignment, userSubmission);
            })
            .collect(Collectors.toList());

    model.addAttribute("assignmentsWithSubmission",
        assignmentsWithSubmission);

    return "user/course-details";
}
```

Kode 3.38: Controller course details student

Pengambilan daftar tugas dilakukan melalui *AssignmentService* dengan memanfaatkan relasi antara entitas tugas dan mata pelajaran, sehingga hanya tugas yang relevan dengan mata pelajaran tersebut yang ditampilkan kepada peserta, sebagaimana ditunjukkan pada Kode 3.39.

```
public List<Assignment> getAssignmentsBySubject(Subject
    subject) {
```

```
    return assignmentRepository.findBySubject(subject);
}
```

Kode 3.39: Service pengambilan assignment berdasarkan subject

Untuk setiap tugas, sistem mengambil data pengumpulan terakhir milik peserta melalui *SubmissionService*. Data pengumpulan diurutkan berdasarkan waktu unggah untuk memastikan bahwa status yang digunakan merupakan data terbaru, sebagaimana ditunjukkan pada Kode 3.40.

```
public Optional<Submission>
    getSubmissionByAssignmentAndStudent(
        Assignment assignment, User user) {

    List<Submission> submissions =
        submissionRepository
        .findByAssignmentAndUser(assignment, user);

    if (submissions.isEmpty())
        return Optional.empty();

    submissions.sort((s1, s2) ->
        s2.getUploadTime().compareTo(s1.getUploadTime()));

    return Optional.of(submissions.get(0));
}
```

Kode 3.40: Service pengambilan submission terakhir student

Hasil penggabungan data tugas dan data pengumpulan direpresentasikan dalam sebuah *Data Transfer Object* yang menyimpan objek tugas beserta status pengumpulan secara opsional. Struktur ini mempermudah pemrosesan logika tampilan pada sisi antarmuka pengguna, sebagaimana ditunjukkan pada Kode 3.41.

```
public class AssignmentWithSubmissionDTO {
    private Assignment assignment;
    private Optional<Submission> submission;

    public AssignmentWithSubmissionDTO(
            Assignment assignment,
            Optional<Submission> submission) {
        this.assignment = assignment;
        this.submission = submission;
    }
```

```
}
```

Kode 3.41: DTO assignment with submission

Pada sisi tampilan, *template* melakukan iterasi terhadap daftar *assignmentsWithSubmission*. Setiap kartu tugas menampilkan judul, deskripsi, tenggat waktu, serta tombol aksi yang disesuaikan dengan kondisi tenggat dan status pengumpulan. Sistem secara otomatis menonaktifkan proses pengumpulan apabila tenggat waktu telah terlewati, sebagaimana ditunjukkan pada Kode 3.42.

```
<div th:if="${assignmentsWithSubmission != null
    and !assignmentsWithSubmission.isEmpty()}">

  <div th:each="assignmentWithSubmission :
      ${assignmentsWithSubmission}"
       th:with="assignment=
         ${assignmentWithSubmission.assignment},
         submission=
         ${assignmentWithSubmission.submission}">

    <h3 th:text="${assignment.title}">
        Assignment Title</h3>

    <span th:text="${#temporals.format(
        assignment.dueDate,
        'dd MMM yyyy HH:mm')}">
        Deadline</span>

    <p th:text="${assignment.description}">
        Assignment description</p>

    <a th:if="${assignment.filePath != null}"
       th:href="@{'/' + ${assignment.filePath}}"
       download>
       Download materials
    </a>

    <div th:unless="${#temporals.createNow()
        .isAfter(assignment.dueDate)}">
      <button class="open-submission-modal">
        <span th:text="${submission.isPresent()
          ? 'Edit submission'
          : 'Submit your work'}">
```

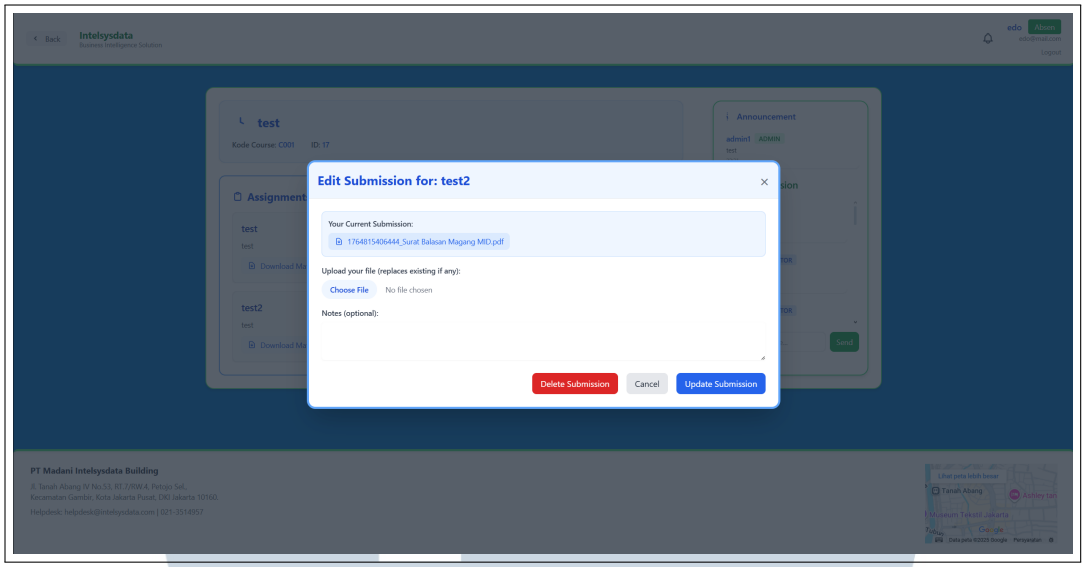
```
        </span>
      </button>
    </div>

    <div th:if="${#temporals.createNow()
        .isAfter(assignment.dueDate)}">
      <button disabled>Closed</button>
    </div>

  </div>
</div>
```

Kode 3.42: Template daftar assignment student

Dengan adanya halaman *Course Details Student* ini, peserta dapat memantau dan mengelola aktivitas tugas secara terstruktur. Integrasi antara controller, service, DTO, dan template tampilan memastikan konsistensi data, validasi tenggat waktu, serta kesiapan sistem untuk mendukung proses pengumpulan tugas secara berkelanjutan. Gambar 3.30 serta Kode 3.38 Kode 3.42 merepresentasikan keseluruhan alur dan implementasi halaman course details student.

3. **Submission Student**

Halaman *Submission* pada sisi *Student* direpresentasikan dalam bentuk jendela *modal* yang digunakan untuk melakukan pengumpulan, pembaruan, maupun penghapusan berkas tugas. Modal ini muncul ketika peserta menekan tombol pengumpulan pada salah satu kartu tugas di halaman *Course Details* dan memungkinkan seluruh proses dilakukan tanpa perpindahan halaman, sebagaimana ditunjukkan pada Gambar 3.31.



Gambar 3.31. Modal submission student

Secara konseptual, mekanisme pengumpulan tugas dirancang untuk menyesuaikan kondisi data yang dimiliki peserta. Sistem membedakan antara proses pengumpulan baru dan proses pembaruan berdasarkan keberadaan data pengumpulan sebelumnya, sehingga alur interaksi tetap konsisten dan intuitif.

Struktur utama modal terdiri dari formulir dengan metode *POST* dan tipe *multipart*. Formulir ini dilengkapi dengan *CSRF token* serta *hidden input* untuk menyimpan *ID assignment* yang sedang diproses. Apabila peserta telah memiliki pengumpulan sebelumnya, informasi berkas lama akan ditampilkan sebagai referensi. Implementasi struktur modal ini ditunjukkan pada Kode 3.43.

```
<div id="submission-modal"
     class="fixed inset-0 hidden">

  <form id="submission-form"
        method="post"
        enctype="multipart/form-data">

    <input type="hidden"
           th:name="${_csrf.parameterName}"
           th:value="${_csrf.token}" />

    <input type="hidden"
           id="submission-assignment-id"
```

```
            name="assignmentId" />

    <div id="existing-submission-info"
         class="hidden">
      <p>Your current submission:</p>
      <a id="existing-submission-file-link"
         href="#">
        <span id="existing-submission-file-name"></span>
      </a>
    </div>

    <label>Upload your file</label>
    <input type="file"
           name="file"
           id="file-upload" />

    <label>Notes</label>
    <textarea name="note"
              rows="3"></textarea>

    <button type="submit">
        Submit
    </button>
  </form>
</div>
```

Kode 3.43: Modal submission student

Pengendalian logika modal dilakukan pada sisi klien menggunakan *JavaScript*. Ketika tombol pengumpulan ditekan, sistem membaca atribut *data-** pada kartu tugas untuk menentukan apakah peserta telah memiliki pengumpulan sebelumnya. Informasi ini digunakan untuk mengatur judul modal, menampilkan atau menyembunyikan informasi berkas lama, serta menentukan status wajib unggah berkas, sebagaimana ditunjukkan pada Kode 3.44.

```
openBtns.forEach(btn => btn.addEventListener('click',
    function() {
  const assignmentId = this.dataset.assignmentId;
  const title = this.closest('.assignment-card')
                    .querySelector('h3').textContent;
  const subId = this.dataset.submissionId;
  const subPath = this.dataset.submissionFilePath;
  const subName = this.dataset.submissionFileName;
```

```
8
9   submissionAssignmentIdInput.value = assignmentId;
10  submissionForm.action =
11    `/user/assignment/${assignmentId}/submit`;
12
13  if (subId && subPath && subName) {
14    document.getElementById('submission-modal-title')
15      .textContent = 'Edit submission for: ' + title;
16    existingInfo.classList.remove('hidden');
17    existingLink.href = '/' + subPath;
18    existingName.textContent = subName;
19    fileInput.removeAttribute('required');
20    deleteBtn.classList.remove('hidden');
21    deleteBtn.dataset.submissionId = subId;
22  } else {
23    document.getElementById('submission-modal-title')
24      .textContent = 'Submit for: ' + title;
25    existingInfo.classList.add('hidden');
26    fileInput.setAttribute('required', 'required');
27    deleteBtn.classList.add('hidden');
28  }
29
30  submissionModal.classList.remove('hidden');
31 }));
```

Kode 3.44: JavaScript membuka dan mengisi modal submission

Selain proses pembukaan modal, sistem juga menyediakan fungsi untuk menutup dan mereset kondisi formulir. Proses ini memastikan tidak ada data yang tertinggal ketika peserta berpindah dari satu tugas ke tugas lainnya, sebagaimana ditunjukkan pada Kode 3.45.

```
1 function closeSubmissionModal() {
2   submissionModal.classList.add('hidden');
3   submissionForm.reset();
4   existingInfo.classList.add('hidden');
5   fileInput.setAttribute('required', 'required');
6   deleteBtn.classList.add('hidden');
7 }
```

Kode 3.45: JavaScript menutup dan mereset modal

Apabila peserta memilih untuk menghapus pengumpulan yang telah ada, proses tersebut dilakukan melalui permintaan *DELETE* secara asinkron ke sisi server. Permintaan ini dilengkapi dengan *CSRF header* untuk

menjaga keamanan operasi penghapusan data, sebagaimana ditunjukkan pada Kode 3.46.

```
deleteBtn.addEventListener('click', function() {
  const submissionId = this.dataset.submissionId;
  const csrfToken =
    document.querySelector('meta[name="_csrf"]').content;
  const csrfHeader =
    document.querySelector('meta[name="_csrf_header"]').
   content;

  if (submissionId && confirm('Delete submission?')) {
    fetch(`/user/submission/${submissionId}`, {
      method: 'DELETE',
      headers: {
        [csrfHeader]: csrfToken
      }
    });
  }
});
```

Kode 3.46: AJAX delete submission student

Pada sisi server, pengolahan pengumpulan tugas ditangani oleh *SubmissionController*. Endpoint *POST* digunakan untuk menangani proses pembuatan maupun pembaruan data pengumpulan, sedangkan endpoint *DELETE* digunakan untuk menghapus data pengumpulan berdasarkan *ID* yang diterima, sebagaimana ditunjukkan pada Kode 3.47.

```
@PostMapping("/assignment/{id}/submit")
public String submitAssignment(
        @PathVariable("id") Long assignmentId,
        @RequestParam("file") MultipartFile file,
        @RequestParam(value = "note", required = false)
    String note,
        Authentication authentication) {
    // create or update submission
}

@DeleteMapping("/submission/{id}")
public ResponseEntity<?> deleteSubmission(
        @PathVariable("id") Long submissionId,
        Authentication authentication) {
    // delete submission
```

```
15 }
```

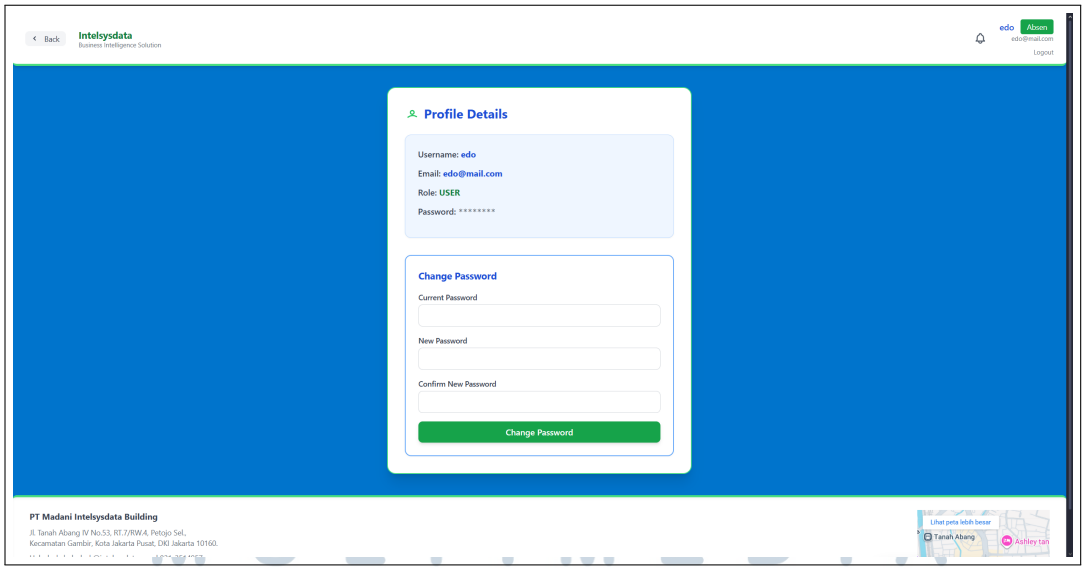
Kode 3.47: Controller submission student

Dengan adanya mekanisme *Submission Student* ini, peserta dapat melakukan pengelolaan pengumpulan tugas secara fleksibel dan terkontrol. Integrasi antara modal antarmuka, logika *JavaScript*, serta controller pada sisi server memastikan validasi tenggat waktu, keamanan permintaan, dan konsistensi data tetap terjaga dalam sistem MID E-learning. Gambar 3.31 serta Kode 3.43 sampai Kode 3.47 merepresentasikan keseluruhan alur dan implementasi fitur pengumpulan tugas pada sisi student.

4. **Change Password Student**

Halaman *Change Password* pada sisi *Student* berfungsi sebagai antarmuka bagi pengguna untuk memperbarui kata sandi akun secara mandiri. Fitur ini disediakan sebagai bagian dari upaya menjaga keamanan akun, sehingga pengguna dapat mengganti kata sandi secara berkala tanpa perlu melibatkan pihak administrator.

Tampilan halaman *Change Password Student* ditunjukkan pada Gambar 3.32.



Gambar 3.32. Halaman change password student

Secara konseptual, proses perubahan kata sandi dilakukan melalui mekanisme validasi berlapis pada sisi server. Sistem memastikan bahwa permintaan berasal dari pengguna yang sedang terautentikasi, kata sandi

lama yang dimasukkan sesuai dengan data yang tersimpan, serta kata sandi baru memenuhi kriteria keamanan yang telah ditetapkan.

Formulir perubahan kata sandi dikirim menggunakan metode *POST* dan disediakan pada halaman profil pengguna. Formulir ini terdiri dari tiga *field* utama, yaitu kata sandi saat ini, kata sandi baru, dan konfirmasi kata sandi baru. Seluruh *field* bersifat wajib untuk mencegah terjadinya pembaruan data yang tidak lengkap. Implementasi formulir perubahan kata sandi ditunjukkan pada Kode 3.48.

```
<form th:action="@{/user/profile/change-password}"
      method="post">

  <label>Current password</label>
  <input type="password"
         name="currentPassword"
         required />

  <label>New password</label>
  <input type="password"
         name="newPassword"
         required />

  <label>Confirm new password</label>
  <input type="password"
         name="confirmPassword"
         required />

  <button type="submit">
      Change password
  </button>
</form>

<div th:if="${error}"
     th:text="${error}"></div>
<div th:if="${success}"
     th:text="${success}"></div>
```

Kode 3.48: Form change password student

Pengolahan permintaan perubahan kata sandi ditangani oleh *UserProfileController*. Pada tahap awal, sistem mengambil data pengguna berdasarkan sesi autentikasi yang aktif, kemudian memverifikasi kecocokan kata sandi lama menggunakan mekanisme pencocokan terenkripsi. Validasi

berikutnya memastikan bahwa kata sandi baru dan konfirmasi memiliki nilai yang sama serta memenuhi batas minimum panjang karakter yang ditentukan oleh sistem. Implementasi logika tersebut ditunjukkan pada Kode 3.49.

```
@PostMapping("/user/profile/change-password")
public String changePassword(
        Authentication authentication,
        @RequestParam String currentPassword,
        @RequestParam String newPassword,
        @RequestParam String confirmPassword,
        Model model) {

    String username = authentication.getName();
    User user =
        userService.getUserByUsername(username)
        .orElse(null);

    model.addAttribute("user", user);

    if (user == null) {
        model.addAttribute("error",
            "User not found.");
        return "user/profile-details";
    }

    if (!passwordEncoder.matches(
            currentPassword,
            user.getPassword())) {
        model.addAttribute("error",
            "Current password is incorrect.");
        return "user/profile-details";
    }

    if (!newPassword.equals(confirmPassword)) {
        model.addAttribute("error",
            "New password and confirmation do not match.");
        return "user/profile-details";
    }

    if (newPassword.length() < 6) {
        model.addAttribute("error",
            "New password must be at least 6 characters.");
```

```
39          return "user/profile-details";
40      }
41
42      userService.updateUserPassword(user, newPassword);
43      model.addAttribute("success",
44          "Password changed successfully.");
45
46      return "user/profile-details";
47  }
```

Kode 3.49: Controller change password student

Proses pembaruan kata sandi pada *service layer* dilakukan dengan menerapkan enkripsi sebelum data disimpan kembali ke dalam basis data. Pendekatan ini memastikan bahwa kata sandi tidak pernah disimpan dalam bentuk teks asli dan tetap memenuhi standar keamanan sistem. Implementasi proses pembaruan kata sandi ditunjukkan pada Kode 3.50.

```
1  public void updateUserPassword(
2          User user,
3          String newPassword) {
4
5      user.setPassword(
6          passwordEncoder.encode(newPassword));
7      userRepository.save(user);
8  }
```

Kode 3.50: Service update password user

Dengan adanya fitur *Change Password Student*, sistem memberikan kontrol langsung kepada pengguna dalam menjaga keamanan akun. Validasi berlapis, penerapan enkripsi, serta integrasi dengan mekanisme autentikasi memastikan proses perubahan kata sandi berjalan aman, konsisten, dan sesuai dengan praktik keamanan yang baik dalam sistem MID E-learning.

3.4 Kendala dan Solusi yang Ditemukan

Dalam proses pelaksanaan magang dan pengembangan sistem *MID E-Learning*, terdapat sejumlah kendala yang muncul. Kendala ini tidak hanya bersifat teknis, tetapi juga berkaitan dengan aspek manajerial dan komunikasi. Meskipun jumlah kendala yang dihadapi relatif terbatas, dampaknya cukup signifikan terhadap jalannya proyek. Berikut uraian kendala utama beserta solusi yang diterapkan:

Kendala

1. Kurangnya bimbingan dan komunikasi Karena bekerja secara mandiri, arahan dari supervisor maupun rekan kerja sangat terbatas. Hal ini menimbulkan kesulitan dalam memastikan bahwa rancangan sistem benar-benar sesuai dengan standar perusahaan dan kebutuhan pengguna akhir. Minimnya diskusi juga membuat proses validasi ide menjadi lebih lambat, sehingga setiap keputusan desain harus diambil dengan penuh pertimbangan dan risiko. Kondisi ini menuntut penulis untuk lebih proaktif dalam mencari referensi eksternal dan melakukan analisis mandiri agar sistem tetap relevan dan fungsional.

2. Manajemen waktu dan beban kerja Dengan tanggung jawab penuh dari tahap perancangan, implementasi, hingga dokumentasi, pengaturan waktu menjadi tantangan besar. Setiap modul yang dirancang memiliki kompleksitas tersendiri, sehingga apabila tidak dikelola dengan baik dapat menumpuk di akhir periode magang. Selain itu, adanya tuntutan kualitas yang tinggi membuat proses pengerjaan tidak bisa dilakukan secara terburu-buru. Beban kerja yang besar ini menuntut adanya disiplin dalam menyusun prioritas, membagi tugas ke dalam bagian-bagian kecil, serta menjaga konsistensi dalam setiap tahapan pengembangan.

Solusi

1. Inisiatif komunikasi mandiri Untuk mengatasi keterbatasan bimbingan, dilakukan upaya proaktif berupa penyusunan laporan mingguan yang terstruktur, dokumentasi teknis yang jelas, serta permintaan umpan balik secara berkala kepada supervisor. Dengan cara ini, meskipun komunikasi tidak intensif, tetap ada jalur konfirmasi yang membantu memastikan sistem yang dibangun sejalan dengan ekspektasi perusahaan. Selain itu, penulis juga memanfaatkan forum pengembang, artikel teknis, dan dokumentasi resmi framework sebagai sumber referensi tambahan untuk memperkuat keputusan desain.

2. Penerapan manajemen waktu yang disiplin Solusi utama untuk mengatasi beban kerja adalah dengan membuat jadwal kerja harian dan mingguan yang realistis. Setiap modul dibagi ke dalam *milestone* kecil agar progres

lebih mudah dipantau. Penggunaan *version control system* seperti Git juga membantu dalam menjaga keteraturan pekerjaan, sehingga setiap perubahan dapat terdokumentasi dengan baik. Dengan strategi ini, pekerjaan dapat diselesaikan secara bertahap tanpa menimbulkan penumpukan di akhir, sekaligus menjaga kualitas hasil yang diharapkan.