**BAB I**

**PENDAHULUAN**

**1.1 Latar Belakang**

Tujuan dari suatu pembelajaran adalah agar peserta didik dapat memahami materi yang diajarkan oleh pengajar melalui metode yang berbeda-beda, namun tujuan dari metode-metode tersebut adalah pemahaman yang baik dari peserta didiknya dan mereka aktif serta menerapkan segala sesuatu yang telah dipelajarinya berdasarkan pengetahuan maupun pemahaman yang dimilikinya. Banyak cara yang dilakukan oleh pengajar seperti membuat *quiz*, menambah waktu belajar, memperbanyak latihan soal, sampai memberi tugas yang banyak. Hal tersebut diharapkan membuat peserta didik tetap belajar di lingkungan sekolah atau di luar sekolah dan mempersiapkan untuk menghadapi soal-soal yang diberikan pengajar sehingga mendapatkan hasil yang baik, serta  kemampuan peserta didik pun dapat diketahui dari hal tersebut.

Pemahaman peserta didik yang baik itu sendiri berasal dari cara belajar dan lingkungan yang sesuai dengan peserta didik. Dengan ini pengajar perlu mengikuti cara belajar peserta didik sehingga mereka dapat memahami materi dengan mengetahui tingkat kemampuan dan cara belajar serta  bagaimana untuk meningkatkannya. Masalah yang dihadapi pengajar sendiri jika pembelajaran dilakukan di kelas memiliki peserta didik yang banyak, pengajar sulit untuk mengetahui dan menyesuaikan cara belajar masing-masing dari peserta didik. Dari cara belajar dan lingkungannya peserta didik juga akan beragam dan pemahaman menjadi tidak seimbang antar peserta didik.

1

Salah satu contoh masalah berikut terdapat di salah satu Universitas di Bandung yaitu Universitas Nasional Pasim yang beralamat di Jalan Dakota No. 8A. Dimana terdapat suatu jurusan yaitu Manajemen Informatika dimana terdapat dua jenis mahasiswa, yaitu mahasiswa beasiswa PUB (Pemberdayaan Umat Berkelanjutan) dan mahasiswa Reguler. Mahasiswa beasiswa PUB sendiri adalah mahasiswa yang mengikuti beberapa tahap penyeleksian dalam penerimaan mahasiswanya. Terdapat pelatihan pemograman yang dapat membantu mahasiswa PUB dalam pembelajaran pemograman di perkuliahan. Sedangkan untuk mahasiswa Reguler hanya mengikuti pembelajaran di kuliah dan belajar mandiri, namun terkadang tertinggal dalam memahami suatu materi dikarenakan mahasiswa kurang berlatih dan  tidak aktif untuk mencari tahu tentang materi tersebut.

Padahal pembelajaran sangat bergantung dari peserta didik karena setiap manusia pasti memiliki cara sendiri untuk memahami suatu pelajaran. Namun dapat diketahui juga, jika lingkungan seseorang dapat mempengaruhi dalam belajar. Seperti mahasiswa PUB melakukan pelatihan pemrograman, sedangkan Mahasiswa Reguler tidak mendapatkan pelatihan seperti PUB. Pengajar juga menganalisis selama beberapa tahun hasil dari pembelajaran mahasiswa jurusan Manajemen Informatika yang  memperlihatkan data bahwa mahasiswa beasiswa PUB lebih baik dibandingkan mahasiswa Reguler.

Menurut data di Jurusan Manajemen Informatika, beberapa mahasiswa dapat meningkatkan kemampuannya. Istilah yang kami berikan bagi mahasiswa yang mengalami peningkatan kemampuan secara signifikan adalah *Good Learner*. Misalkan ada mahasiswa yang awalnya dia selalu mendapatkan nilai yang rendah dan kemudian dia mengalami peningkatan kemampuan sehingga mendapatkan nilai yang tinggi. Bagi mahasiswa yang belum mengalami peningkatan atau mahasiswa yang nilainya masih di bawah akan diarahkan untuk mengikuti jalur belajar *Good Learner*.

Pengajar juga berharap dapat meningkatkan kemampuan mahasiswanya agar memiliki kemampuan yang seimbang dan dapat mengikuti cara belajar *Good Learner*  saat mengerjakan soal atau latihan dari dosen sehingga dosen dapat melihat peningkatan dari mahasiswa tersebut dan juga akan mudah untuk memberi bimbingan bagi mahasiswa yang sulit memahami materi dari pengajar.

Untuk membantu pengajar maka dibutuhkan sistem yang akan digunakan dalam kegiatan mengajar sehingga dapat lebih mudah untuk menilai perkembangan mahasiswa dalam belajar. Mengikuti kemajuan teknologi saat ini, dikenal istilah *E-Learning*atau Pembelajaran Elektronik yang memiliki definisi sebagai sebuah bentuk teknologi informasi yang diterapkan di bidang pendidikan berupa *website* yang dapat diakses di mana saja. Pemanfaatan *e-learning* dapat memotivasi peningkatan kualitas pembelajaran dan materi ajar, kualitas aktivitas dan kemandirian mahasiswa, serta komunikasi antara dosen dengan mahasiswa maupun antar mahasiswa. Dan hal yang dibutuhkan adalah aplikasi yang akan kami bangun dapat menemukan cara belajar *Good Learner* sehingga mahasiswa yang bukan *Good Learner* dapat ditunjukan untuk mengikuti jalur belajar *Good Learner*.

*E-Learning* ini akan menyediakan beberapa fitur dan latihan soal yang disesuaikan dengan kemampuan mahasiswa sehingga mahasiswa dapat menjawab soal tersebut, yang artinya sistem memiliki sifat adapatif. Sistem adaptif ini dapat menyesuaikan diri dengan proses adaptasi berdasarkan berbagai tujuan dan preferensi pengguna dan untuk memudahkan pengajar mengetahui hasil belajar mahasiswa pada *E-Learning*, maka sistem ini juga akan menghasilkan ciri - ciri dari  mahasiswa dengan beberapa parameter tertentu, dapat berupa seberapa rajin mahasiswa belajar dengan menggunakan aplikasi yang penulis bangun dan berapa besar nilai *quiz* yang didapatkan. Sistem tersebut dinamakan sistem yang terpersonalisasi, dikarenakan menggambarkan kemampuan dari masing-masing personal.

Untuk membantu hal tersebut, dibutuhkan metode belajar atau pengujian kemampuan mahasiswa dalam memahami materi. Maka penulis akan mengajukan judul  “*GOOD LEARNER-BASED ADAPTIVE AND PERSONALIZED E-LEARNING*”. Diharapkan dosen mengetahui dan memahami cara belajar peserta didiknya serta meningkatan pemahaman mahasiswa dalam suatu pembahasan sehingga mahasiswa dapat mengerti materi dan dosen dapat mencapai tujuan pengajarannya pada setiap mata kuliah.

**1.2 Identifikasi Masalah**

Berdasarkan latar belakang yang telah penulis kemukakan, maka penulis mengidentifikasi beberapa masalah sebagai berikut :

1. Bagaimana membangun sistem yang dapat mengetahui kemampuan mahasiswa di Jurusan Manajemen Informatika, Universitas Nasional Pasim, secara adaptif dan terpersonalisasi.
2. Bagaimana menerapkan aplikasi *E-learning* adatif dan terpersonalisasi di

Universitas Nasional Pasim sehingga mampu mendukung kegiatan belajar mengajar.

1. Bagaimana meningkatkan kemampuan dengan mengikuti cara belajar *Good Learner.*

**1.3** **Maksud dan Tujuan Penelitian**

**1.3.1 Maksud Penelitian**

Membangun Aplikasi *E-Learning* yang dapat membantu Dosen untuk mengetahui cara belajar *good learner* melalui perekaman belajar mahasiswa dalam aplikasi *E-Learning* yang adaptif dan terpersonalisasi dengan memberi *quiz* - *quiz* yang sesuai diajarkan dosen sehingga cara belajar *good learner* tersebut dapat diterapkan pada mahasiswa Manajemen Informatika yang lain.

* + 1. **Tujuan Penelitian**

Berdasarkan Rumusan Masalah yang penulis kemukakan, Penulis menentukan beberapa tujuan dari pembuatan Tugas Akhir ini yaitu :

1. Membangun sistem yang dapat mengetahui kemampuan mahasiswa di jurusan Manajemen Informatika,Universitas Nasional Pasim, secara

adaptif dan terpersonalisasi.

1. Menerapkan aplikasi *E-learning* adatif dan terpersonalisasi di Universitas Nasional Pasim sehingga mampu mendukung kegiatan belajar mengajar.
2. Meningkatkan kemampuan mahasiswa Reguler atau mahasiswa beasiswa PUB  dengan mengikuti cara belajar *Good Learner.*

**1.4 Batasan Masalah**

Adapun batasan masalah dalam pembangunan aplikasi ini adalah :

1. Data yang digunakan diambil dari Universitas Nasional Pasim.
2. Mata kuliah yang digunakan merupakan mata kuliah dasar pemrograman yaitu Logika dan Algoritma yang menggunakan bahasa pemograman Python.
3. Aplikasi memberikan informasi (hasil) pencapaian tingkat kemampuan setiap mahasiswa kepada dosen dan tidak memberikan penanganan dari hasil tersebut.
4. Menerapkan Teori Pembelajaran Modeling dalam menghasilkan generasi *Good Learner*. Maksud dari Teori Pembelajaran Modeling adalah Teori yang diterapkan dalam sebuah Pembelajaran terdapat model yang bisa ditiru untuk menjadi lebih baik dalam suatu bidang tertentu.

**1.5 Kegunaan Penelitian**

**1. Bagi Penulis**

Penelitian ini akan menambah wawasan penulis dalam mengetahui cara belajar *Good Learner* dan sistem pembelajaran elektronik yang adatif dan terpersonalisasi serta meningkatkan kemampuan penulis dalam bidang pemrograman.

**2. Bagi Dosen Logika dan Algoritma Universitas Nasional Pasim**

Penelitian ini sangat diharapkan dapat membantu dosen Logika dan Algoritma di Universitas Nasional Pasim menjalankan tugasnya sebagai pengajar, memudahkan dalam menilai kemampuan mahasiswa, mengarahkan, dan mengetahui cara belajar *Good Learner*

**3. Bagi Mahasiswa Jurusan Manajemen Informatika**

Dapat meningkatkan cara belajar dan meningkatkan kemampuan seperti *Good Learner* serta memudahkan mahasiswa untuk belajar materi yang diajarkan.

**1.6 Metodologi Penelitian**

* + 1. **Metode Pengumpulan Data**

Dalam menyusun Tugas Akhir ini, penulis melakukan penerapan metode penelitian  dalam memperoleh data-data yang dibutuhkan sehingga penyusunan Tugas Akhir dapat diselesaikan dengan baik. Adapun metode penelitian yang penulis lakukan adalah:

1. **Studi Lapangan**

Merupakan metode yang dilakukan oleh penulis secara langsung pada Universitas Nasional Pasim sebagai objek penelitian khususnya data mahasiswa jurusan Manajemen Informatika, data nilai matakuliah Logika dan Algoritma.

**2.  Studi Pustaka**

Merupakan metode yang penulis lakukan untuk mendapatkan data-data teoritis baik yang bersumber dari buku maupun artikel yang berhubungan dengan masalah yang penulis hadapi.

**3.   Wawancara**

Pengumpulan data dengan wawancara ini digunakan untuk meyakinkan bahwa data yang diperoleh benar-benar akurat dan pada kesempatan ini penulis mewawancarai dosen mata kuliah Logika dan Algoritma di Universitas Nasional Pasim untuk mendapatkan data.

**4.    Studi Literatur**

Merupakan metode yang dilakukan oleh penulis dengan mengunjungi dan mempelajari *website* atau situs - situs yang berhubungan dengan penelitian ini

**1.6.2 Model Proses Pengembangan Sistem**

**A.   *Prototype***

Penulis memilih untuk menggunakan model proses *Prototype*. Model proses ini dipilih dengan harapan mendapatkan umpan balik yang lebih awal dari pemakai. Dengan metode *Prototype* ini pengembang dan pemakai dapat saling berinteraksi selama proses pembuatan sistem. Berikut adalah penjelasan dari tahap-tahap pemodelan proses *Prototype*.

1. Mengidentifikasi kebutuhan pemakai

Pengembang mewawancarai pemakai untuk mendapatkan gagasan dari apa yang diinginkan pemakai terhadap sistem.

1. Membangun *Prototype*

**Pengembang membangun *prototyping* dengan membuat perancangan sementara yang berfokus pada penyajian kepada pemakai (misalnya dengan membuat format *input* dan *output*).**

1. Evaluasi *Prototype*

Evaluasi dilakukan oleh pemakai dengan menilai *prototype* yang telah dibuat oleh pengembang apakah sudah sesuai dengan keinginan pemakai. Jika sudah sesuai maka langkah selanjutnya dilakukan tetapi jika tidak sesuai maka *prototype* akan diperbaiki kembali dengan mengulang langkah sebelumnya.

1. Pengkodean

**Dalam tahap ini *Prototype* yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.**

1. Menguji Sistem

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, maka harus di tes dahulu sebelum digunakan. Pengujian ini dilakukan dengan *Black Box*.

1. Evaluasi Sistem

Pada tahap ini, pemakai mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan. Jika sudah sesuai maka langkah selanjutnya dilakukan, akan tetapi jika tidak maka akan diperbaiki kembali dengan mengulang langkah Pengkodean dan Menguji Sistem.

1. Menggunakan Sistem

Perangkat lunak yang telah diuji dan diterima pemakai siap untuk digunakan.

**Gambar 1.1** Model *Prototype*

**1.6.3 Metode Pendekatan Pembangunan Sistem**

Metode pendekatan pengembangan sistem yang digunakan dalam Tugas Akhir  ini   adalah  dengan melakukan pendekatan *Object Oriented*.  Adapun   cara   menggambarkan   hasil  analisis dengan pendekatan *Object Oriented* ini menggunakan UML (*Unified Modeling Language*) yang digambarkan dalam bentuk :

* *Use Case Diagram*
* *Activity Diagram*
* *Sequence Diagram*
* *Class Diagram*

**1.7 Lokasi dan Waktu Penelitian**

Lokasi penelitian dilakukan di Universitas Nasional Pasim, Jl. Dakota No. 8A, Cicendo, Sukaraja, Bandung. Adapun rincian kegiatan penyusunan Tugas Akhir yang direncanakan penulis adalah sebagai berikut :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kegiatan** | **Bulan** | | | | | | | | |
| **Okt** | **Nov** | **Des** | **Jan** | **Feb** | **Mar** | **Apr** | **Mei** | **Jun** |
| 1. | Identifikasi Kebutuhan |  |  |  |  |  |  |  |  |  |
| 2. | Membangun *Prototype* |  |  |  |  |  |  |  |  |  |
| 3. | Evaluasi *Prototype* |  |  |  |  |  |  |  |  |  |
| 4. | Pengkodean |  |  |  |  |  |  |  |  |  |
| 5. | Menguji Aplikasi |  |  |  |  |  |  |  |  |  |
| 6. | Evaluasi Aplikasi |  |  |  |  |  |  |  |  |  |
| 7. | Penggunaan Aplikasi |  |  |  |  |  |  |  |  |  |

**Tabel 1.1** Rincian Kegiatan Penyusunan Tugas Akhir

**BAB II**

**LANDASAN TEORI**

**2.1** **Pengertian *Good Learner***

Seseorang dikatakan *Good Learner* apabila ia dapat meningkatkan kemampuannya secara signifikan. Perubahan ini biasanya dikarenakan adanya keinginan dari orang tersebut dan juga usaha yang dilakukan untuk perubahan yang lebih baik. *Good* dalam Bahasa inggris sendiri berarti bagus atau baik sedangkan *Learner* adalah berasal dari kata *Learn* mempelajari jika diberi imbuhan -*er* dibelakang kata kerja maka akan menjadi kata benda atau pelaku yaitu Pelajar. Jika dihubungkan maka akan diartikan Pelajar yang baik. Kami menyimpulkan jika yang disebut *Good Learner* ialah dapat mengikuti Pembelajaran dan meningkatkannya.

**2.2** Pengertian *E-Learning*

Jaya Kumar C. Koran mendefinisikan E-learningsebagai pembelajaran yang menggunakan rangkaian elektronik (LAN, WAN, atau *internet*) untuk menyampaikan isi pembelajaran, interaksi, atau bimbingan. Sedangkan Dong mendefinisikan *E-*learning sebagai kegiatan belajar asynchronousmelalui perangkat elektronik komputer yang memperoleh bahan belajar yang sesuai dengan kebutuhannya.

Onno W. Purbo menjelaskan bahwa istilah “e” atau singkatan dari elektronik dalam e-learningdigunakan sebagai istilah untuk segala teknologi yang digunakan untuk mendukung usaha-usaha pengajaran lewat teknologi elektronik internet. Internet, Intranet, satelit, *tape* audio/video, TV interaktif dan CD-ROM adalah sebagian dari media elektronik yang digunakan pengajaran boleh disampaikan secara synchronously (pada waktu yang sama) ataupun asynchronously (pada waktu yang berbeda). Materi pengajaran dan pembelajaran yang disampaikan melalui media ini mempunyai teks, grafik, animasi, simulasi, audio dan video. Perbedaan pembelajaran tradisional dengan e-learning yaitu kelas tradisional, dosen/guru dianggap sebagai orang yang serba tahu dan ditugaskan untuk menyalurkan ilmu pengetahuan kepada pelajarnya. Sedangkan di dalam pembelajaran e-learningfokus utamanya adalah mahasiswa/siswa. Mahasiswa mandiri pada waktu tertentu dan bertanggung-jawab untuk pembelajarannya. Hasbullah menjelaskan filosofis e-learning sebagai berikut :

12

1. E-learning merupakan penyampaian informasi, komunikasi, pendidikan,

pelatihan secara on-line.

1. E-learning menyediakan seperangkat alat yang dapat memperkaya nilai belajar secara konvensional (model belajar konvensional, kajian terhadap buku teks, CD-ROM, dan pelatihan berbasis komputer) sehingga dapat menjawab tantangan perkembangan globalisasi.
2. E-learning tidak berarti menggantikan model belajar konvensional didalam kelas, tetapi memperkuat model belajar tersebut melalui pengayaan konten dan pengembangan teknologi Pendidikan.
3. Kapasitas mahasiswa amat bervariasi tergantung pada bentuk isi dan cara  
    penyampaiannya. Makin baik keselarasan antar konten dan alat penyampai dengan gaya belajar, maka akan lebih baik kapasitas siswa yang pada gilirannya akan memberi hasil yang lebih baik.

Menurut Onno W. Purbo mensyaratkan tiga hal yang wajib dipenuhi dalam merancang e-learning, yaitu : sederhana, personal, dan cepat. Sistem yang sederhana akan memudahkan peserta didik dalam memanfaatkan teknologi dan menu yang ada, dengan kemudahan pada panel yang disediakan, akan mengurangi pengenalan sistem e-learning itu sendiri, sehingga waktu belajar peserta dapat diefisienkan untuk proses belajar itu sendiri dan bukan pada belajar menggunakan sistem e-learning-nya. Syarat personal berarti pengajar dapat berinteraksi dengan baik seperti layaknya seorang guru yang berkomunikasi dengan murid di depan kelas. Dengan pendekatan dan interaksi yang lebih personal, peserta didik diperhatikan kemajuannya, serta dibantu segala persoalan yang dihadapinya.

**2.2.1** **Komponen *E-learning***

Secara garis besar, ada 3 komponen utama yang menyusun *e-learning*:

1. *E-learning System*

Sistem perangkat lunak yang memvirtualisasi proses belajar mengajar konvensional. Bagaimana manajemen kelas, pembuatan materi atau konten, forum diskusi, sistem penilaian (rapor), sistem ujian *online* dan segala fitur yang berhubungan dengan manajemen proses belajar mengajar. Sistem perangkat lunak tersebut sering disebut dengan *Learning Management System* (LMS).

2. *E-learning Content* (Isi)

Konten dan bahan ajar yang ada pada *e-learning system* (*learning management system*). Konten dan bahan ajar ini bisa dalam bentuk *Multimedia-based Content* (konten berbentuk multimedia interaktif) atau *Text-based Content* (konten berbentuk teks seperti pada buku pelajaran biasa).

3.  *E-learning Infrastructure* (Peralatan)

Infrastruktur *e-learning* dapat berupa *personal computer* (PC), jaringan komputer dan perlengkapan multimedia.

**2.2.2** **Kategori *E-Learning***

Ditinjau dari segi interaksi antara sistem dengan manusia maka ada tiga kategori dasar dari *e-learning*, yaitu:

1. *Synchronous Learning*

Pada  pembelajaran  *synchronous*  kondisinya  mirip dengan pembelajaran konvensional hanya saja pada *e-learning* hal ini tidak ditandai dengan kehadiran secara fisik. Pada bentuk *synchronous* ini pendidik (instruktur), peserta didik dan rekan-rekannya  melakukan  pertemuan secara  *online* di  internet. Melakukan proses belajar mengajar seolah sedang berada pada ruang fisik yang sama.

1. *Self-directed Learning*

Pada kategori  ini peserta didik melakukan pembelajaran  secara mandiri. Dengan mengakses berbagai  referensi  dan  bahan  belajar  yang  disediakan. Tidak  ada instruktur ataupun waktu khusus untuk berdiskusi dengan sesama peserta didik. Masing-masing  peserta  didik melakukan proses belajar sesuai dengan kebutuhanya.

1. *Asynchronous (Collaborative) Learning*

Kategori  ini  mengkombinasikan  karakteristik  dari kedua  kategori  sebelumnya. Peserta didik belajar secara mandiri namun tetap berkomunikasi dengan peserta didik  lainnya maupun dengan pendidik walaupun  tidak harus di waktu khusus. Penggunaan *email*,instant  *message*  (*Yahoo!  Messenger,  Gtalk*)  ataupun  *board* pada forum dapat digunakan sebagai media komunikasi dan interaksi baik dengan pendidik maupun sesama peserta didik.

**2.3** **Pengembangan *Adaptive* *E-learning***

Pengembangan *adaptive* *e-learning* berbeda dalam beberapa cara dari pengembangan perangkat lunak umum. Perbedaan ini terutama terkait dengan fasilitas navigasi, peran pengguna, adaptasi dinamis isi, navigasi dan presentasi. Pengembangan sistem *adaptive* *e-learning*  ini berawal dari adanya kebutuhan proses pembelajaran berbasis *web* terhadap sistem adaptif. Dengan menggunakan pendekatan sistem adaptif, diharapkan proses pembelajaran melalui *web* akan menjadi lebih optimal. Hal ini, karena sistem adaptif dapat menyesuaikan diri dengan proses adaptasi berdasarkan berbagai tujuan dan preferensi pengguna, sebagaimana dijelaskan oleh Fröschl berikut ini:

*“An adaptive system adapts itself or another system to various circumstances. The process of adaptation is based on user’s goals and preferences”* yang artinya Sebuah sistem adaptif akan mengadaptasi dengan dirinya sendiri atau sistem yang lain dari berbagai lingkup. Proses adaptasi didasari oleh tujuan dan pilihan pengguna.

Dengan demikian *e-learning* dapat difasilitasi untuk menyampaikan materi pembelajaran sesuai dengan kebutuhan dan preferensi pengguna.  Sistem yang demikian bisa dikatakan sistem  *e-learning* adaptif. Sistem *e-learning* disebut bersifat adaptif apabila sistem mampu menyesuaikan secara otomatis kepada pengguna berdasarkan asumsi tentang pengguna. Disamping itu, *e-learning* yang adaptif, seharusnya mampu memberi kesempatan kepada pengguna untuk mengubah perilaku sistem sesuai dengan keinginan pengguna itu sendiri. Sistem *e-learning* adaptif perlu mengakomodasi kondisi atau karakteristik pengguna dan menyimpan semua informasi ini dalam model pengguna dan selanjutnya sistem akan memanfaatkan informasi ini sebagai dasar untuk menyampaikan materi pembelajaran.

Menurut Brusilovsky, teknik melakukan adaptasi secara garis besar dapat dikategorikan menjadi dua, yaitu *Adaptive presentation* (adaptasi isi) dan *Adaptive* *navigation support* (adaptasi link). *Adaptive presentation* adalah teknik yang digunakan untuk mengadaptasikan isi halaman *web* berdasarkan model pengguna. Teknik ini meliputi, presentasi teks adaptif dan presentasi multimedia adaptif. Sedangkan *Adaptive navigation support* adalahteknik yang digunakan untuk memodifikasi *link* yang dapat diakses pengguna pada waktu tertentu.

Selanjutnya tingkat adaptasi dalam sistem *e-learning* adaptif dapat dibedakan tergantung pada siapa yang mengambil inisiatif, sistem atau siswa. Pendekatan ini dibedakan menjadi dua bentuk dalam adaptasi yaitu, *Adaptivity* dan *Adaptability*. Menurut Papanikolaou bahwa *Adaptivity* mengacu pada sistem *e-learning* adaptif yang menyesuaikan presentasi menggunakan data tentang siswa dengan cara sistem yang dikontrol. Sedangkan *Adaptability* adalah suatu sistem *e-learning* adaptif yang mendukung *modifiability* pengguna akhir dengan menyediakan kontrol siswa atas fungsi tertentu. Dalam merancang sistem *e-learning* adaptif, penting untuk menerapkan kedua tingkat adaptasi tersebut

**2.4**  **Personalisasi *E-Learning***

Personalisasi *e-learning* adalah sebuah strategi dimana sistem *e-learning* akan mampu melayani pengguna sesuai kebutuhannya. Dengan kata lain situs *e-learning* tersebut akan tampil dan seolah-olah tahu keinginan, kebutuhan dan apa-apa yang ingin diperoleh setiap pengguna. Walaupun jumlah pengguna sistem tersebut banyak maka sistem *e-learning* dengan personalisasi tetap akan tampil dinamis sesuai kebutuhan dan karakter tiap pengguna. Sistem tersebut akan bekerja sebagaimana sistem *e-learning* pada umumnya. Sistem akan mengamati perilaku dari pengguna dan dicatat dalam *database*. Pengamatan dan pencatatan sistem *e-learning* meliputi topik materi belajar, diktat, buku dan bahan kuliah digital apa saja yang sering diakses oleh seorang pengguna. Diharapkan cara ini akan meningkatkan kedekatan sistem terhadap pengguna, memberikan kesan sistem tersebut ramah terhadap pengguna dan mengerti kebutuhan dan keinginan pengguna.

* 1. **Java**

Java adalah sebuah teknologi yang diperkenalkan oleh Sun Microsystems pada pertengahan tahun 1990. Menurut definisi Sun, Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Java sebagai sebuah teknologi sebanding hanya sebuah bahasa pemrograman, karena Java lebih lengkap dibanding sebuah bahasa pemrograman konvensional. Teknologi Java memiliki empat komponen penting, yaitu:

1. *Programming-language specification*
2. *Application-programming interface*
3. *Virtual-machine specification*
4. *JDK (Java Development Kit)*

*Java Development Kit* (JDK) adalah produk Sun Microsystems yang ditujukan untuk pengembang Java. Sejak diperkenalkannya Java, telah jauh SDK Java yang paling banyak digunakan. Pada tanggal 17 November 2006, Sun mengumumkan bahwa akan dirilis di bawah GNU *General* *Public* *License* (GPL), sehingga membuat perangkat lunak bebas. Hal ini terjadi sebagian besar pada tanggal 8 Mei 2007. Sun kontribusi kode sumber untuk JDKOpen. JRL (*Java* *Research* *License*) ini dibuat khusus untuk universitas dan peneliti yang ingin menggunakan teknologi Java sebagai subyek pembelajaran dan penelitian. Ia ditulis untuk menyederhanakan dan bersantai persyaratan bagian yang ada penelitian dari SCSL saat ini. JRL adalah lisensi yang diciptakan khusus untuk universitas dan peneliti yang ingin menggunakan teknologi Java sebagai subyek pembelajaran dan penelitian. NetBeans mengacu pada kedua kerangka *platform* untuk aplikasi desktop Java, dan sebuah lingkungan pengembangan terpadu (IDE) untuk pengembangan dengan Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure, dan lain-lain. NetBeans IDE ditulis dalam Java dan berjalan di mana-mana. Di mana JVM yang diinstal, termasuk Windows, Mac OS, Linux, dan Solaris. Sebuah JDK diperlukan untuk pengembangan fungsionalitas Java, tetapi tidak diperlukan untuk pembangunan di bahasa pemrograman lain. *Platform* NetBeans memungkinkan aplikasi untuk dikembangkan dari satu set komponen perangkat lunak moduler yang disebut modul. Aplikasi berbasis *platform* NetBeans (termasuk IDE NetBeans) dapat diperpanjang oleh pengembang pihak ketiga.

**2.5.1** [**Karakteristik Java**](http://aalfyansyah.blogspot.com/2012/09/karakteristik-java.html)

Kelebihan dan Karakteristik Java Sintaks bahasa pemograman java adalah pengembangan dari bahasa pemograman C/C++, sehingga dengan mereka yang sudah terbiasa dengan C/C++ tidak akan mengalami kesulitan mempelajari bahasa pemograman Java. Java adalah bahasa pemograman yang sederhana dan tangguh. Berikut ini adalah beberapa karakteristik dari Java sesuai dengan *white paper* dari SUN.

1. Berorientasi Objek, Java telah menerapkan konsep pemograman berorientasi objek yang modern dalam implementasinya
2. *Robust*, Java mendorong pemograman yang bebas dengan kesalahan yang bersifat *strongly typed* dan memiliki *run time checking*.
3. *Protable*, pemograman Java berjalan pada sistem operasi apapun yang memiliki *Java* *Virtual* *Machine*.
4. *Multithreading*, Java mendukung pemograman *multithreading* dan terintegrasi secara langsung dalam bahasa Java.
5. Dinamis, program Java dapat melakukan sesuatu tindakan yang ditentukan pada saat eksekusi program dan pada saat kompilasi.
6. Sederhana, Java menggunakan bahasa yang sederhana dan mudah dipelajari.
7. Terdistribusi, Java didesain untuk berjalan pada lingkungan yang terdistribusi seperti halnya internet.
8. Aman, aplikasi yang dibuat dengan bahasa Java lebih dapat dijamin keamanannya terutama untuk aplikasi internet.
9. Netral secara arsitektur, Java tidak terkait pada suatu mesin atau mesin operasi tertentu.
10. *Interpreted*, aplikasi java bisa dieksekusi pada *platform* yang berbeda-beda karena melakukan interpretasi pada *bytecode*.
11. Berkinerja tinggi, *byte* kode Java telah teroptimasi dengan baik sehingga eksekusi program dapat dilakukan dengan cepat.
    1. **Spring Framework**

Spring merupakan sebuah *framework* (kerangka kerja) yang digunakan untuk membangun sebuah aplikasi *Enterprise*. Spring termasuk *framework* yang *lightweight* (ringan) untuk mendukung secara penuh dalam pengembangan aplikasi *Enterprise* siap pakai. Spring dapat digunakan untuk melakukan pengaturan deklarasi manajemen transaksi, *remote* *access* dengan menggunakan RMI atau layanan *web* lainnya, fasilitas *mailing*, dan beragam opsi untuk pengaturan data ke *database*. Spring juga memungkinkan untuk menggunakan hanya modul-modul tertentu sehingga tidak perlu menggunakan semua modul Spring dalam aplikasi apabila tidak diperlukan. Fitur-fitur dari Spring *Framework* :

1. *Transaction Management* : Spring *framework* menyediakan sebuah layer

yang generik untuk manajemen transaksi, sehingga memudahkan para *developer* dalam melakukan manajemen transaksi.

1. JDBC *Exception Handling* : layer abstrak JDBC menawarkan *exception*

yang bersifat hierarki sehingga memudahkan penanganan *error*.

1. *Integration* with Hibernate, JDO, and iBatis : Spring menawarkan layanan

integrasi terbaik dengan Hibernate, JDO dan iBatas

1. AOP *Framework* : Spring merupakan *framework* AOP terbaik yang

pernah ada.

1. MVC *Framework* : Spring hadir dengan *framework* aplikasi *web* MVC, yang dibangun di atas inti Spring. Spring merupakan *framework* yang sangat fleksibel dalam pengaturan strategi *interface*, dan mengakomodasi beberapa teknologi *view* seperti JSP, Velocity, Tiles, iText, dan POI.
   1. **Hibernate**

Hibernate adalah salah satu teknologi *Object* *Relational* *persistence* *Mapping* (ORM) dan query *service*. Transformasi *persistent* data dari sebuah representasi data (*relational database*) ke sebuah representasi data yang lain (*java objects*) ataupun sebaliknya adalah fungsi yang sebenarnya dilakukan oleh ORM.

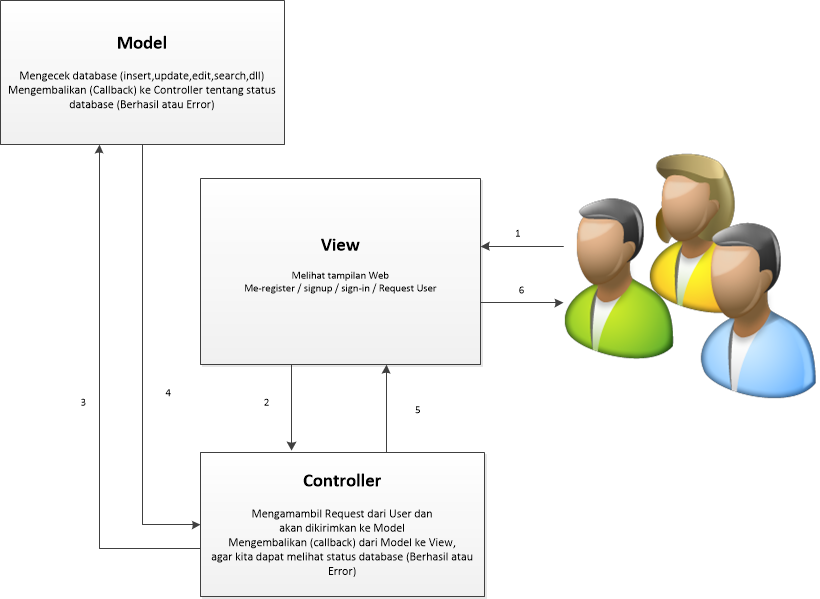
*Persistent object* dapat dibuat menggunakan hibernate dengan idiom bahasa pemrograman java termasuk didalamnya *association*, *inheritance*, *polymorphism*, *composition*, *java* *collection* *framework* seperti Set, List, Map. Hibernate adalah *free* *software* yang menggunakan lisensi LGPL. Hibernate mendukung banyak DBMS (*Database* *Management* *System*) antara lain Oracle, DB2, MySQL, PostgreSQL, Sybase, SAP DB, HypersonicSQL, Microsoft SQL *Server*, Posgress, McKoi SQL, *Pointbase* dan *Interbase*. Hibernate dapat diintegrasikan dengan *Framework-framework* java yang lain seperti JBoss (hibernate sekarang merupakan bagian dari JBoss Project), Spring *Framework*, Struts, JSF maupun *framework* java yang lain sebagai *persistence* *layer*.

* 1. **MVC**

*Model View Controller* merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi *web*, berawal pada bahasa pemrograman *Small Talk* MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, *user interface*, dan bagian yang menjadi kontrol aplikasi. Terdapat 3 jenis komponen yang membangun suatu MVC *pattern* dalam suatu aplikasi yaitu:

1. *View*, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi *web* bagian ini biasanya berupa *file* *template* HTML, yang diatur oleh *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada *user*. Bagian ini tidak memiliki akses langsung terhadap bagian model.
2. Model, biasanya berhubungan langsung dengan *database* untuk memanipulasi data (*insert, update, delete, search*), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.
3. *Controller*, merupakan bagian yang mengatur hubungan antara bagian model dan bagian *view*, *controller* berfungsi untuk menerima *reques*t dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.

Singkat kata Model untuk mengatur alur *database*, *View* untuk menampilkan *web*, sedangkan *Controller* untuk mengatur alur kerja antara *Model* dan *View*.

[](http://situsali.com/wp-content/uploads/2015/07/mvc_konsep.png)

**Gambar 2.1** Proses *Model*-*View*-*Controller*

* 1. **HTML *(Hypertext markup Language)***

*Hypertext Markup Language* (HTML) adalah bahasa yang digunakan untuk menulis halaman *web*. HTML merupakan pengembangan dari standar pemformatan dokumen teks yaitu *Standard Generalized Markup Language* (SGML). HTML sebenarnya adalah dokumen ASCII atau teks biasa, yang dirancang untuk tidak tergantung pada suatu sistem operasi tertentu.

HTML dibuat oleh Tim *Berners-Lee* ketika masih bekerja untuk CERN dan dipopulerkan pertama kali oleh *browser* Mosaic. Selama awal tahun 1990 HTML mengalami perkembangan yang sangat pesat. Setiap pengembangan HTML pasti akan menambahkan kemampuan dan fasilitas yang lebih baik dari versi sebelumnya. Namun perkembangan resmi dikeluarkan pada bulan November 1995 oleh IETF (*Internet Engineering Task Force*).

* 1. **CSS (*Cascading Style Sheet*)**

*Cascading Style Sheet* (CSS) merupakan salah satu bahasa pemrograman *web* untuk mengendalikan beberapa komponen dalam sebuah *web* sehingga akan lebih terstruktur dan seragam.

Sama halnya *styles* dalam aplikasi pengolahan kata seperti *Microsoft Word* yang dapat mengatur beberapa *style*, misalnya *heading*, *subbab*, *bodytext*, *footer*, *images*, dan *style* lainnya untuk dapat digunakan bersama-sama dalam beberapa berkas (*file*). Pada umumnya CSS dipakai untuk memformat tampilan halaman *web* yang dibuat dengan bahasa HTML dan XHTML.

CSS dapat mengendalikan ukuran gambar, warna bagian tubuh pada teks, warna tabel, ukuran border, warna border, warna *hyperlink*, warna *mouse over*, spasi antar paragraf, spasi antar teks, margin kiri, kanan, atas, bawah, dan parameter lainnya. CSS adalah bahasa *stylesheet* yang digunakan untuk mengatur tampilan dokumen. Dengan adanya CSS memungkinkan untuk menampilkan halaman yang sama dengan format yang berbeda.

* 1. **Javascript**

Javascript merupakan bahasa *scripting* sederhana yang secara khusus diciptakan untuk digunakan di dalam *web browser* yang membuat *website* menjadi lebih dinamis. Bila HTML berjalan sendirian, hanya mampu menghasilkan halaman-halaman statis. Setelah di *load*, tampilan halaman HTML tak banyak berubah sampai mengklik sebuah link menuju halaman lain.

Banyak yang mengira Javascript berhubungan dengan Java padahal keduanya tidak sama.  Java, yang diciptakan oleh  Sun Microsystems, merupakan bahasa pemrogaman komputer sepenuhnya seperti C++, cocok untuk penulisan program skala besar dan lengkap. Javascript, di sisi lain, diciptakan oleh  Netscape. Ini berbasis pada suatu tingkatan di dalam Java *syntax code* nya amat mirip, namun sangat jarang dipergunakan untuk keperluan di luar lingkup *browser*.

JavaScript ditulis dengan cara yang sama dengan HTML dalam *editor teks* (Wordpad, Notepad). Penerapan JS sama  persis dengan CSS ,yang bisa mentautkan *link* ke file eksternal dengan ekstensi file .js, atau menulis blok kodenya langsung dalam dokumen HTML dengan *tag*.

* 1. **UML (Unified Modeling Language)**

UML (*Unified* *Modeling* *Language*) merupakan sebuah bahasa pemodelan yang digunakan untuk menspesifikasikan, menvisualisasikan, membangun serta mendokumentasikan seluruh rancangan sistem perangkat lunak (*software*). Pemodelan ini digunakan untuk menyederhanakan permasalahan agar lebih mudah dipahami. Penggunaan model ini juga bertujuan untuk mengidentifikasi bagian-bagian yang termasuk ke dalam lingkungan sistem dan bagaimana hubungan antara sistem dengan subsistem maupun lingkungan luarnya.

Selain itu UML (*Unified* *Modeling* *Language*) merupakan metodologi kolaborasi antara metoda-metoda Booch, OMT (*Object* *Modeling* *Technique*), serta OOSE (*Object* *Oriented* *Software* *Engineering*) dan beberapa metoda lainnya. UML merupakan metodologi yang paling sering digunakan saat ini untuk mengadaptasi maraknya penggunaan bahasa pemograman berorientasi objek (OOP). UML menyediakan notasi-notasi yang mampu membantu dalam memodelkan sistem dari berbagai sudut.

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek. Namun yang akan penulis bahas hanya 4 diagram yaitu:

1. *Use* *Case* *Diagram,* merupakan diagram yang digunakan untuk memodelkan proses bisnis.
2. *Sequence* *Diagram*, merupakan diagram yang digunakan untuk memodelkan pengiriman pesan (*messages*) antar objek.
3. *Activity* *Diagram*, merupakan diagram yang digunakan untuk memodelkan perilaku *Use* *Cases* dan *Objects* di dalam sistem.
4. *Class* *Diagram*, merupakan diagram yang digunakan untuk memodelkan struktur kelas.

Berikut akan dijelaskan 4 macam diagram yang paling sering digunakan dalam membangun aplikasi berorientasi objek, yaitu *use* *case* *diagram*, *activity* *diagram*, *sequence* *diagram*, dan *class* *diagram*.

* + 1. ***Use*** ***Case*** ***Diagram***

*Use* *case* pada dasarnya merupakan unit fungsionalitas korehen yang diekspresikan sebagai transaksi-transaksi yang terjadi antara aktor dan sistem. *Use* *case* diagram digunakan untuk mendefinisikan suatu bagian perilaku sistem yang bersifat korehen tanpa perlu menyingkap struktur internal sistem / perangkat lunak yang sedang dikembangkan. Adapun simbol-simbol yang digunankan dalam *Use Case* Diagram:

1. ***Actor***

*Actor* merupakan suatu simbol yang menunjukkan pelaku yang berinteraksi dengan sistem. Berikut adalah simbol *Actor*:



**Gambar 2.2** Simbol *Actor* dalam *Use* *Case* *Diagram*

1. ***Use Case***

*Use Case* merupakan simbol yang menunjukkan suatu fungsi yang bisa dilakukan. Berikut adalah simbol Use Case:



**Gambar 2.3** Simbol *Use Case* dalam *Use Case Diagram*

1. ***Association***

*Association* merupakan suatu simbol yang digunakan untuk menghubungkan antara objek satu dengan objek yang lainnya. Berikut adalah simbol *Association*:



**Gambar 2.4** Simbol *Association* dalam *Use Case Diagram*

1. ***Extend***

*Extend* merupakan simbol relasi *use* *case* tambahan ke sebuah *use* *case*. Dimana *use* *case* yang ditambahkan tersebut dapat berdiri sendiri walau tanpa *use* *case* tambahan. Berikut adalah simbol *Extend*:



**Gambar 2.5** Simbol *Extend* dalam *Use Case Diagram*

1. ***Include***

*Include* merupakan suatu simbol yang digunakan untuk menggambarkan relasi *use* *case* tambahan ke sebuah *use case*. Dimana *use* *case* yang ditambahkan tersebut memerlukan *use* *case* ini untuk menjalankan fungsinya. Berikut merupakan simbol *Include*:



**Gambar 2.6** Simbol *Include* dalam *Use Case Diagram*

1. ***System Boundary***

*System Boundary* merupakan suatu simbol yang menunjukkan batasan dari suatu sistem. Berikut adalah simbol *System Boundary*:



**Gambar 2.7** Simbol *System Boundary* dalam *Use Case Diagram*

1. ***Generalization***

*Generalization* merupakan suatu simbol yang menggambarkan hubungan generalisasi dan spesialisasi antar dua use case. menunjukkan objek induk yang berbagi perilaku dengan objek anak**.**

Berikut adalah simbol *Generalization*:



**Gambar 2.8** Simbol *Generalization* dalam *Use Case Diagram*

* + 1. ***Activity*** ***Diagram***

*Activity diagram* merupakan bentuk khusus dari *state machine* yang bertujuan memodelkan komputasi-komputasi dan aliran-aliran kerja yang terjadi dalam sistem / perangkat lunak yang sedang dikembangkan. *Activity* *diagram* menggambarkan aliran kerja (*workflow*) atau aktivitas kerja dari sebuah sistem.

Keunggulan dari *activity* *diagram* adalah bahwa diagram lebih mudah dipahami dibandingkan dengan skenario. Pada dasarnya *activity* *diagram* menggambarkan skenario secara grafis. *Activity* *diagram* juga sesungguhnya mirip / serupa dengan diagram alir (*flowchart*). Yang membedakan hanyalah *swimlane* (garis tegas dari atas ke bawah) yang menunjukan suatu *state* berada pada objek / kelas tertentu. Berikut simbol yang digunakan dalam *Activity Diagram*.

1. ***Initial*** ***State***

*Initial* *State* merupkan suatu simbol yang menunjukkan titik awal memulai suatu aktivitas. Berikut adalah *Initial* *State* simbol:



**Gambar 2.9** Simbol *Initial* *State* dalam *Activity Diagram*

1. ***Final*** ***State***

*Final* *State* merupakan suatu simbol yang menunjukkan titik akhir untuk mengakhiri suatu aktivitas. Berikut adalah simbol *Final* *State*:



**Gambar 2.10** Simbol *Final* *State* dalam *Activity Diagram*

1. ***Decision***

*Decision* merupakan suatu simbol yang menunjukkan percabangan, menunjukan pilihan (sesuatu yang harus dipilih). Berikut adalah simbol *Decision*:



**Gambar 2.11** Simbol *Decision* dalam *Activity Diagram*

1. ***Fork And Join***

*Fork* *And* *Join*merupakan suatu simbol yang menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu. Berikut adalah simbol *Fork* *And* *Join*:



**Gambar 2.12** Simbol *Fork* *And* *Join* dalam *Activity Diagram*

1. ***Activity***

*Activity* merupakan suatu simbol yang menggambarkan aktivitas yang berjalan, yang dilakukan oleh sistem. Berikut adalah simbol *Activity*:



**Gambar 2.13** Simbol *Activity* dalam *Activity Diagram*

1. ***Swimlane***

*Swimlane* merupakan suatu simbol yang memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi. Berikut adalah simbol *Swimlane*:



**Gambar 2.14** Simbol *Swimlane* dalam *Activity Diagram*

* + 1. ***Sequence*** ***Diagram***

*Sequence diagram* adalah suatu diagram yang menggambarkan interaksi antar objek dan mengindikasikan komunikasi diantara objek-objek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh objek-objek yang melakukan suatu tugas atau aksi tertentu.

*Sequence* *diagram* mendeskripsikan bagaimana entitas dalam sistem berinteraksi, termasuk pesan yang digunakan saat interaksi. *Sequence* *diagram* berhubungan erat dengan *Use* *Case* diagram, dimana 1 *Use* *Case* akan menjadi 1 *Sequence* Diagram.

Berikut simbol-simbol yang digunakan dalam *Sequence* *Diagram*.

1. ***Actor***

*Actor* merupakan suatu simbol yang menunjukkan pelaku yang berinteraksi dengan sistem. Berikut adalah simbol *Actor*:



**Gambar 2.15** Simbol *Actor* dalam *Sequence Diagram*

1. ***Object***

*Object* atau yang biasa disebut partisipan merupakan *instance* dari sebuah *class*. Berikut adalah *Object* simbol:



**Gambar 2.16** Simbol *Object* dalam *Sequence Diagram*

1. ***Lifeline***

*Lifeline* merupakan suatu simbol yang menunjukkan waktu mulai dan berakhirnya pesan. Mengindikasikan keberadaan sebuah objek dalam basis waktu. Berikut adalah simbol *Lifeline*:



**Gambar 2.17** Simbol *Lifeline* dalam *Sequence Diagram*

1. ***Activation***

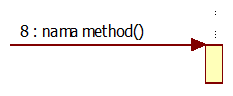
*Activation* merupakan suatu simbol yang berfungsi untuk mengindikasikan sebuah objek yang melakukan sebuah aksi. Berikut adalah simbol *Activation*:



**Gambar 2.18** Simbol *Activation* dalam *Sequence Diagram*

1. ***Call Message***

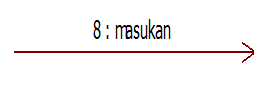
*Call* *Message* merupakan suatu simbol yang menyatakan suatu objek atau memanggil operasi / *method* yang ada pada objek lain atau dirinya sendiri. Berikut adalah simbol *Call* *Message*:



**Gambar 2.19** Simbol *Call* *Message* dalam *Sequence Diagram*

1. ***Send Message***

*Send Message* merupakan suatu simbol yang menyatakanbahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah kepada objek yang dikirim. Berikut adalah simbol *Send Message*.



**Gambar 2.20** Simbol *Send Message* dalam *Sequence Diagram*

1. ***Return Message***

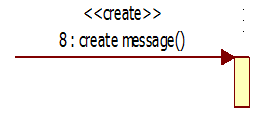
*Return* *Message* merupakan suatu simbol yang menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau *method* menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian. Berikut adalah simbol *Return* *Message*:



**Gambar 2.21** Simbol *Return* *Message* dalam *Sequence Diagram*

1. ***Create Message***

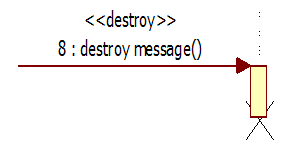
*Create Message* merupakan suatu simbol yang menyatakan suatu objek dapat membuat objek lain, arah panah mengarah pada objek yang dibuat. Berikut adalah simbol *Create Message*:



**Gambar 2.22** Simbol *Create Message* dalam *Sequence Diagram*

1. ***Destroy Message***

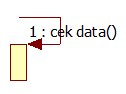
*Destroy Message* merupakan suatu simbol yang menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri. Berikut adalah simbol *Destroy Message*:



**Gambar 2.23** Simbol *Destroy Message* dalam *Sequence Diagram*

1. ***Self- Message***

*Self- Message* merupakan simbol yang menunjukkan komunikasi ke dalam sebuah objek itu sendiri. Berikut adalah simbol:



**Gambar 2.24** Simbol *Self- Message* dalam *Sequence Diagram*

* + 1. ***Class*** ***Diagram***

Diagram kelas atau *class diagram* menunjukkan interaksi antar kelas di dalam sistem. Diagram kelas mengandung informasi dan tingkah-laku segala sesuatu yang berkaitan dengan informasi tersebut. Berikut kegunaan diagram kelas.

1. Mengelompokkan obyek-obyek menjadi kelas - kelas berarti mengabstraksikan masalah yang sedang dihadapi.
2. Definisi-definisi *common* (seperti nama kelas dan atribut) cukup disimpan sekali per instan kelas (objek). Berikut simbol-simbol yang digunakan dalam diagram kelas.
3. ***Association***

*Association* merupakan suatu simbol yang menghubungkan objek satu dengan objek yang lainnya. Berikut adalah simbol *Association*:



**Gambar 2.25** Simbol *Association* dalam *Class Diagram*

1. ***Aggregation***

*Aggregation* merupkan suatu simbol yang menunjukkan hubungan kelas induk dan anak, kelas anak masih bisa berdiri sendiri. Berikut adalah simbol *Aggregation*:



**Gambar 2.26** Simbol *Aggregation* dalam *Class Diagram*

1. ***Composition***

*Composition* merupakan suatu simbol yang menunjukkan hubungan kelas induk dan anak, kelas anak tidak bisa berdiri sendiri. Berikut adalah simbol *Composition*:



**Gambar 2.27** Simbol *Composition* dalam *Class Diagram*

1. ***Dependency***

*Dependency* merupakan suatu simbol yang menunjukkan suatu kelas yang bergantung pada kelas lainnya, perubahan pada kelas yang berdiri sendiri dapat mempengaruhi kelas yang bergantung padanya. Berikut adalah simbol *Dependency*:



**Gambar 2.28** Simbol *Dependency* dalam *Class Diagram*

1. ***Realization***

*Realization* merupakan suatu simbol yang menunjukkan realisasi dari suatu kelas, digunakan untuk menunjukkan kelas implement dari suatu *interface*. Berikut adalah simbol *Realization*:



**Gambar 2.29** Simbol *Realization* dalam *Class Diagram*

1. ***Class***

*Class* merupakan himpunan objek-objek yang berbagi atribut serta operasi yang sama. Berikut adalah simbol *Class*:



**Gambar 2.30** Simbol *Class* dalam *Class* *Diagram*

* 1. **Basis Data**

Basis data adalah kumpulan [informasi](https://id.wikipedia.org/wiki/Informasi) yang disimpan di dalam [komputer](https://id.wikipedia.org/wiki/Komputer) secara sistematik sehingga dapat diperiksa menggunakan suatu [program komputer](https://id.wikipedia.org/wiki/Program_komputer) untuk memperoleh informasi dari basis data tersebut. [Perangkat lunak](https://id.wikipedia.org/wiki/Perangkat_lunak) yang digunakan untuk mengelola dan memanggil [kueri](https://id.wikipedia.org/wiki/Kueri) (*query*) basis data disebut [sistem manajemen basis data](https://id.wikipedia.org/wiki/Sistem_manajemen_basis_data) (*database management system*, DBMS). Sistem basis data dipelajari dalam [ilmu informasi](https://id.wikipedia.org/wiki/Ilmu_informasi).

Istilah *basis data* mengacu pada koleksi dari data-data yang saling berhubungan, dan perangkat lunaknya seharusnya mengacu sebagai sistem manajemen basis data (*database management system / DBMS*). Jika konteksnya sudah jelas, banyak *administrator* dan *programer* menggunakan istilah basis data untuk kedua arti tersebut.

Jadi secara konsep basis data atau *database* adalah kumpulan dari data-data yang membentuk suatu berkas (*file*) yang saling berhubungan (*relation*) dengan tatacara yang tertentu untuk membentuk data baru atau informasi. Atau basis data (*database*) merupakan kumpulan dari data yang saling berhubungan (relasi) antara satu dengan yang lainnya yang diorganisasikan berdasarkan skema atau struktur tertentu. Pada komputer, basis data disimpan dalam perangkat *hardware* penyimpan, dan dengan *software* tertentu dimanipulasi untuk kepentingan atau kegunaan tertentu. Hubungan atau relasi data biasanya ditunjukkan dengan kunci (*key*) dari tiap file yang ada. Data merupakan fakta atau nilai (*value*) yang tercatat atau merepresentasikan deskripsi dari suatu objek. Data yang merupakan fakta yang tercatat dan selanjutnya dilakukan pengolahan (proses) menjadi bentuk yang berguna atau bermanfaat bagi pemakainya akan membentuk apa yang disebut informasi. Bentuk informasi yang kompleks dan terintegrasi dan pengolahan sebuah *database* dengan komputer akan digunakan untuk proses pengambilan keputusan pada manajemen akan membentuk Sistem Informasi Manajemen (SIM), data dalam basis data merupakan item terkecil dan terpenting untuk membangun basis data yang baik dan valid. Data dalam basis data bersifat *integrated* dan *shared*:

1. Terpadu (*integrated*), berkas-berkas data yang ada pada basis data saling terkait (terjadi dependensi data);
2. Berbagi data (*shared*), data yang sama dapat dipakai oleh sejumlah pengguna dalam waktu yang bersamaan. Sering dinamakan sebagi sistem *multiuser*

Data merupakan suatu sumber yang sangat berguna bagi hampir disemua organisasi. Dengan tersedianya data yang melimpah, maka masalah pengaturan data secara efektif menjadi suatu hal yang sangat penting dalam pengembangan sistem informasi manajemen. Oleh karena itu, tujuan dari diadakannya pengaturan data adalah sebagai berikut:

1. Menyediakan penyimpanan data untuk dapat digunakan oleh organisasi saat sekarang dan masa akan datang.
2. Sebagai cara pemasukan data sehingga sehingga memudahkan tugas operator dan menyangkut pula waktu yang diperlukan oleh pemakai untuk mendapatkan data serta hak-hak yang dimiliki terhadap data yang ditangani
3. Pengendalian data untuk setiap siklus agar data selalu *up to date* dan dapat mencerminkan perubahan spesifik yang terjadi di setiap sistem.
4. Pengamanan data terhadap kemungkinan penambahan, modifikasi, pencurian, dan gangguan-gangguaan lain.

Suatu bangunan basis data memiliki jenjang sebagai berikut:

1. Karakter, merupakan bagian data terkecil yang berupa angka, huruf, atau karakter khusus yang membentuk sebuah item data atau *field*. Contoh A,B,X,Y,2,1,2,9,0,=,<,> dan sebagainya.
2. *Field* / item, merupakan representasi suatu atribut dan record (rekaman/tupel) yang sejenis yang menunjukkan suatu item dari data. Contoh *field* nama (berisi data nama-nama pegawai), *field* departemen (berisi data bagian atau spesifikasi pekerjaan), dan lain sebagainya.
3. *Record* / rekaman / tupel: Kumpulan dari *field* membentuk suatu *record* atau rekaman. *Record* menggambarkan suatu unit data individu yang tertentu. Contoh: file pegawai, dimana tiap-tiap *record*nya berisi kumpulan data nama, alamat, departemen, yang dapat mewakili tiap-tiap data.
4. *File*, merupakan kumpulan dari *record-record* yang menggambarkan satu kesatuan data yang sejenis. Contoh *file* pegawai berisi data tentang semua yang berhubungan dengan pegawai seperti nama pegawai, alamat pegawai, departemen, yang dapat mewakili tiap-tiap data.
5. *Database*, merupakan kumupan dari *file* atau tabel yang membentuk suatu database. Contoh *database* pegawai PT Maju Terus terdiri atas *file* pegawai, *file* gaji, *file* golongan, dan sebagainya.

Dalam satu *file* terdapat *record-record* yang sejenis, sama besar, sama bentuk, yang merupakan satu kumpulan entitas yang seragam. Satu *record* terdiri dari *field* yang saling berhubungan menunjukkan bahwa *field* tersebut dalam satu pengertian yang lengkap dan direkam dalam satu *record*. Setiap nilai atau isi *field* memiliki kapasitas ruang atau lebar yang sama. Jenis isi data sebuah *field* harus sesuai dengan tipe datanya. Nama sebuah *file* harus menggambarkan isi dari data *file* tersebut. Untuk melengkapi definisi tentang *file*, dalam database dikenal nama entitas (*entity*) dan atribut. Entetis adalah orang, tempat, kejadian, atau konsep yang informasinya direkam. setiap entitas memiliki atribut atau sebutan untuk mewakili suatu entitas. Sebagai contoh dalam sistem perkuliahan; mahasiwa, matakuliah, pembayaran, dosen adalah sebagai entitas. Sedangkan entitas mahasiswa memiliki atribut nomor induk, nama, jurusan, dan sebagainya. Atau dari contoh di atas entitasnya adalah pegawai, yang memilik atribut NIP, nama, alamat, tgl\_lahir, jns\_kel. Sistem basis data merupakan perpaduan antara basis data dan sistem manajemen basis data (SMBD). *Database* yang kompleks dan disertai dengan teknik pendokumentasian dan prosedur manipulasinya akan membentuk Sistem Manajemen Basis Data (*Database Management System / DBMS*). Singkatnya DBMS adalah *database* dan program untuk mengaksesnya.

Definisi di atas dapat menggambarkan bahwa basis data mempunyai beberapa kriteria penting, yaitu:

1. Berorientasi data dan bukan berorientasi program.
2. Dapat digunakan oleh beberapa program aplikasi tanpa perlu mengubah basis datanya.
3. Dapat dikembangkan dengan mudah, baik volume maupun strukturnya.
4. Dapat memenuhi kebutuhan sistem-sistem baru secara mudah.
5. Dapat digunakan dengan cara-cara yang berbeda.

Komponen-komponen sistem basis data adalah:

1. *Hardware*, sebagai pendukung operasi pengolahan data seperti CPU, memori, *disk*, terminal, dan sebagainya.
2. *Software* sistem operasi , (Windows 9x, Windows 2000/XP, Linux, Unix.)
3. *Software* pengelola basis data (DBMS) seperti MS-Access, SQL, Oracle.
4. *Software* program aplikasi misalnya Visual Basic, Delphi, Visual Foxpro.
5. Basis data (semua data yang diperlukan, dipelihara, dikelola oleh sistem Basis Data)
6. Pemakai/pengguna basis data (*user*).
   * 1. **Peran Basis Data**

Basis data penting dalam pengelolaan data dalam menggunakan komputer, karena beberapa alasan yaitu:

1. Sebagai komponen utama atau penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.
2. Menentukan kualitas informasi yang cepat, akurat, relevan, sehingga informasi yang disajikan tidak basi. Informasi dapat dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya.

Penyusunan suatu *database* digunakan untuk mengatasi masalah-masalah pada penyusunan data. Manfaat yang diperoleh dari penyusunan *database* yaitu untuk:

1. Mengatasi kerangkapan (*redundancy*) data. Penyimpanan data yang sama pada beberapa tempat selain bisa meyulitkan pemakai tentang aktualisasi data juga memboroskan tempat penyimpanan, maka basis data akan mendeteksi dan menghindari jika terjadi kerangkapan data.
2. Menghindari terjadinya inkonsistensi data. Akibat lain jika terjadi kerangkapan data, maka jika terjadi perubahan pada data yang satu sedangkan yang lain tidak dirubah akan terjadi ketidak konsistenan data. Untuk itu *database* akan menyesuaikan terhadap data yang sama apabila terjadi perubahan, jika semua data akan selalu ter-*update*
3. Mengatasi kesulitan dalam mengakses data. Memudahkan jika suatu saat akan diambil atau dicetak data yang memiliki kriteria tertentu, misalnya pada tanggal tertentu, alamat tertentu, jumlah tertentu, dan sebagainya.
4. Menyusun format yang standar dari sebuah data. Data yang sama pada *file* yang berbeda harus memiliki format data berupa tipe dan jangkauannya harus sama. Ketidaksamaan format data akan mengakibatkan sulit atau tidak bisanya pengaksesan data yang lain.
5. Penggunaan oleh banyak pemakai (*multiple user*). Sebuah *database* bisa dimanfaatkan sekaligus secara bersama oleh banyak pengguna (*multiuser*)
6. Melakukan perlindungan dan pengamanan data (*data security*). Setiap data hanya bisa diakses atas dimanipulasi oleh pihak yang diberi otoritas dengan memberikan *login* dan *password* terhadap masing-masing data.
7. Menyusun integritas dan independensi data. Basis data merupakan data kompleks yang bisa diintegrasikan, sehingga *user* dapat memanipulasi untuk mendapatkan berbagai bentuk lembar kerja dan laporan yang di inginkan. Namun demikian masing-masing data tidak saling tergantung terhadap data yang lain. Data-data dalam *database* bisa saling berdiri sendiri.
   1. **MySQL**

MySQL adalah sebuah [perangkat lunak](https://id.wikipedia.org/wiki/Perangkat_lunak) sistem manajemen [basis data](https://id.wikipedia.org/wiki/Basis_data) [SQL](https://id.wikipedia.org/wiki/SQL) ([bahasa Inggris](https://id.wikipedia.org/wiki/Bahasa_Inggris): *database management system*) atau DBMS yang [multialur](https://id.wikipedia.org/w/index.php?title=Multialur&action=edit&redlink=1), [multipengguna](https://id.wikipedia.org/wiki/Multipengguna), dengan sekitar 6 juta instalasi di seluruh dunia. [MySQL AB](https://id.wikipedia.org/w/index.php?title=MySQL_AB&action=edit&redlink=1) membuat MySQL tersedia sebagai [perangkat lunak gratis](https://id.wikipedia.org/wiki/Perangkat_lunak_gratis) di bawah lisensi [GNU *General* *Public License*](https://id.wikipedia.org/wiki/GNU_General_Public_License) (GPL), tetapi mereka juga menjual di bawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti [Apache](https://id.wikipedia.org/wiki/Apache_HTTP_Server), di mana perangkat lunak dikembangkan oleh komunitas umum, dan [hak cipta](https://id.wikipedia.org/wiki/Hak_cipta) untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial [Swedia](https://id.wikipedia.org/wiki/Swedia) [MySQL AB](https://id.wikipedia.org/w/index.php?title=MySQL_AB&action=edit&redlink=1), di mana memegang hak cipta hampir atas semua kode sumbernya.

* 1. **Xampp**

XAMPP adalah [perangkat lunak bebas](https://id.wikipedia.org/wiki/Perangkat_lunak_bebas), yang mendukung banyak [sistem operasi](https://id.wikipedia.org/wiki/Sistem_operasi), merupakan kompilasi dari beberapa [program](https://id.wikipedia.org/wiki/Program). Fungsinya adalah sebagai [*server*](https://id.wikipedia.org/wiki/Server) yang berdiri sendiri ([*localhost*](https://id.wikipedia.org/w/index.php?title=Localhost&action=edit&redlink=1)), yang terdiri atas program [Apache HTTP *Server*](https://id.wikipedia.org/wiki/Apache_HTTP_Server), [MySQL](https://id.wikipedia.org/wiki/MySQL) [*database*](https://id.wikipedia.org/wiki/Database), dan [penerjemah](https://id.wikipedia.org/wiki/Penerjemah) [bahasa](https://id.wikipedia.org/wiki/Bahasa) yang ditulis dengan [bahasa](https://id.wikipedia.org/wiki/Bahasa) [pemrograman](https://id.wikipedia.org/wiki/Pemrograman) [PHP](https://id.wikipedia.org/wiki/PHP) dan [Perl](https://id.wikipedia.org/wiki/Perl). Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), [Apache](https://id.wikipedia.org/wiki/Apache), [MySQL](https://id.wikipedia.org/wiki/MySQL), [PHP](https://id.wikipedia.org/wiki/PHP) dan [Perl](https://id.wikipedia.org/wiki/Perl).

**BAB III**

**OBJEK PENELITIAN DAN ANALISIS**

* 1. **Objek Penelitian**

**3.1.1 Sejarah Universitas Nasional Pasim**

YAPASIM berdiri pada tanggal [11](https://id.wikipedia.org/wiki/11) [Juni](https://id.wikipedia.org/wiki/Juni) [1996](https://id.wikipedia.org/wiki/1996) dengan akta notaris No 41 didepan [notaris](https://id.wikipedia.org/wiki/Notaris) Dr Wiratni Ahmadi SH. Pendirian YAPASIM ini ditujukan untuk menjadi lembaga pengelola [pendidikan](https://id.wikipedia.org/wiki/Pendidikan) tinggi. Oleh karena itu, setelah YAPASIM berdiri tidak berapa lama sesudah itu pendiri mengajukan izin kepada Direktur Jenderal Pendidikan Tinggi Departemen [Pendidikan](https://id.wikipedia.org/wiki/Pendidikan) dan Kebudayaan untuk mendirikan Sekolah Tinggi Ilmu Ekonomi yang diberi nama STIE PASIM.

Dan pada akhirnya permohonan tersebut dikabulkan oleh Menteri pendidikan dan kebudayaan pada tanggal [4](https://id.wikipedia.org/wiki/4) [Desember](https://id.wikipedia.org/wiki/Desember) [1996](https://id.wikipedia.org/wiki/1996) dengan dikeluarkannya SK Menteri Pendidikan dan Kebudayaan RI No.80/D/0/1996 tentang izin STIE PASIM yang terdiri dari dua jurusan yaitu jurusan [manajemen](https://id.wikipedia.org/wiki/Manajemen) dan jurusan [akuntansi](https://id.wikipedia.org/wiki/Akuntansi).

Seiring dengan berjalannya waktu dan perkembangan [teknologi informasi](https://id.wikipedia.org/wiki/Teknologi_informasi), di [Indonesia](https://id.wikipedia.org/wiki/Indonesia) khususnya banyak berdiri [sekolah tinggi](https://id.wikipedia.org/wiki/Sekolah_tinggi) bidang [informatika](https://id.wikipedia.org/wiki/Informatika) dan [komputer](https://id.wikipedia.org/wiki/Komputer). Untuk merespon perkembangan teknologi informasi tersebut serta memenuhi kebutuhan masyarakat, YAPASIM mengajukan izin kepada menteri pendidikan [nasional](https://id.wikipedia.org/wiki/Nasional) untuk mendirikan Sekolah Tinggi Manajemen Informatika dan Komputer yang diberi nama STMIK PASIM. Berdasarkan pada SK Menteri Pendidikan Nasional RI No 164/D/0/2000 tanggal [23](https://id.wikipedia.org/wiki/23) [Agustus](https://id.wikipedia.org/wiki/Agustus) [2000](https://id.wikipedia.org/wiki/2000) secara resmi berdirilah STMIK PASIM, sehingga di bawah pengelolaan YAPASIM ada dua sekolah tinggi yaitu STIE PASIM dan STMIK PASIM.

47

Tidak hanya berhenti sampai disitu, setelah beberapa tahun kemudian pendiri YAPASIM sangat berkeinginan untuk mendirikan [universitas](https://id.wikipedia.org/wiki/Universitas), namun pada saat itu izin untuk mendirikan universitas baru tidak dikeluarkan lagi oleh [Departemen Pendidikan Nasional](https://id.wikipedia.org/wiki/Departemen_Pendidikan_Nasional). Namun hal ini tidak menghalangi keinginan pendiri YAPASIM untuk mengelola universitas, tepatnya pada tanggal [25](https://id.wikipedia.org/wiki/25) [September](https://id.wikipedia.org/wiki/September) [2006](https://id.wikipedia.org/wiki/2006) pengelola Yayasan Pendidikan Kader Keuangan dan Perbankan yang mengelola Universitas Nasional Bandung menyerahkan pengelolaan Universitas Nasional Bandung kepada pendiri YAPASIM. Penyerahan tersebut dituangkan dalam akta notaris No [25](https://id.wikipedia.org/wiki/25) tanggal 25 [September](https://id.wikipedia.org/wiki/September) [2006](https://id.wikipedia.org/wiki/2006) didepan Notaris R. Sabar Partakoesoema, SH.MH.

Setelah berada di bawah pengelolaan YAPASIM, Universitas Nasional Bandung diajukan perubahan namanya kepada Menteri Pendidikan Nasional [RI](https://id.wikipedia.org/wiki/RI) menjadi Universitas Nasional Pasim (Pengembangan Sistem Informasi Manajemen). Pengajuan perubahan nama tersebut disetujui oleh Menteri Pendidikan Nasional RI dengan dikeluarkannya SK Menteri Pendidikan Nasional RI No 172/D/0/2007 pada tanggal [6](https://id.wikipedia.org/wiki/6) [September](https://id.wikipedia.org/wiki/September) [2007](https://id.wikipedia.org/wiki/2007) di bawah Yayasan Pendidikan Kader Keuangan Perbankan.

Merger dilakukan dikarenakan pada tahun sebelumnya UNB (YPKKP) memiliki defisit mahasiswa yang cukup besar. seluruh [mahasiswa](https://id.wikipedia.org/wiki/Mahasiswa) UNB yang terbagi kedalam 5 fakultas tidak mencapai 150 orang. Dalam keadaan tersebut pemilik yayasan (YPKKP) menjual salah satu gedung milik UNB kepada Infomedia. keadaan ini makin memperuncing konflik antara mahasiswa UNB dan pemilik yayasan, dengan dijualnya aset tersebut ada kemungkinan aset-aset lain akan dijual jika mahasiswa UNB kembali berkurang pada tahun berikutnya.

Wancana menjual UNB pun bergulir, beberapa tawaran semisal, dari UK (Uniersitas Kebangsaan) untuk menjual fakultas ekonomi yang dimiliki UNB ditolak karena yayasan ingin menjual seluruh fakultas yang dimilikinya. Pada perkembangan selanjutnya, UNB akhirnya dibeli oleh Yapasim. Seluruh mahasiswa ex-UNB diberikan opsi untuk ikut bergabung dengan management baru atau keluar selamanya dari UNB (atau Universitas Nasional Pasim).

**3.1.2 Mahasiswa D3 Manajemen Informatika**

Objek Penelitian yang kami lakukan di Universitas Nasional Pasim ditujukan unutk Mahasiswa D3 Manajemen Informatika. Dalam jurusan ini terdapat dua jalur mahasiswa. Yang pertama yaitu Mahasiwa jalur Beasiswa Pemberdayaan Umat Berkelanjutan  dan yang kedua adalah Mahasiswa jalur Reguler.

1. Mahasiswa Pemberdayaan Umat Berkelanjutan (PUB)

Program pemberdayaan umat berkelanjutan (PUB) adalah program kerjasama Universitas Nasional Pasim dengan Yayasan Marlina Buchari untuk menampung lulusan setingkat SMA yang Kurang mampu untuk melanjutkan pendidikannya di perguruan tinggi tetapi memiliki kecerdasan yang baik.

Untuk diterima sebagai mahasiswa program PUB, mereka harus mengikuti seleksi yang sangat ketat dan melibatkan para psikolog, mengingat program PUB ini dirancang untuk mencetak lulusan yang unggul dan memiliki mental yang baik.

Selama mengikuti program PUB, mahasiswa PUB ditempatkan di asrama dan seluruh biaya pendidikan dan akomodasi ditanggung Universitas Nasional Pasim, dalam mengikuti program PUB, mereka diwajibkan mengikuti aturan-aturan dan pendidikan ketat mulai dari kedisiplinan, mental spiritual, kemampuan khusus dibidang Teknologi Informasi, kemampuan bahasa asing, program magang, kemampuan bekerja dan kemampuan lainnya.

Setiap harinya mahasiswa PUB melakukan pelatihan pemograman , bahasa inggris dan lain-lain. Pelatihan ini ditujukan untuk membantu mahasiswa dalam belajar di perkuliahan agar mahasiswa dapat mendapatkan IPK diatas 3.25.

1. Mahasiswa Reguler

Mahasiswa Reguler adalah mahasiswa yang mengikuti pembelajaran seperti biasa namun yang penulis maksud yang tidak mengikuti jalur beasiswa melainkan jalur umum. Dilihat dari kegiatannya sendiri mahasiswa PUB lebih banyak kegiatan belajar seperti pelatihan , namun untuk mahasiswa Reguler tidak. Sehingga mahasiswa memiliki lingkungan yang berbeda yang dapat mempengaruhi cara belajar.

**3.1.3 Visi Universitas Nasional PASIM**

Visi Universitas Nasional Pasim adalah : “To Make World Class Quality of Human Resources Development in Global Era of Information System” yang artinya membangun Sumber daya manusia berkelas dunia di era global sistem informasi.

**3.1.4 Misi Universitas Nasional PASIM**

Misi Universitas Nasional Pasim adalah : “The best curriculum, the best lecturers, the best facilities” yang artinya menyediakan kurikulum terbaik, pengajar terbaik, dan fasilitas terbaik.

**3.1.5 Struktur Organisasi**

Dibawah ini adalah struktur organisasi Universitas Nasional Pasim 

**Gambar 3.1** Struktur Organisasi Universitas Nasional Pasim

**3.2 Analisis Sistem**

Analisis sistem merupakan pendefinisian suatu sistem menjadi beberapa bagian dengan tujuan untuk mengidentifikasi suatu permasalahan serta mengevaluasinya agar didapatkan suatu sistem yang sesuai dengan kebutuhan.

Berdasarkan data-data yang telah diperoleh, kemudian dilakukan analisis pada sistem yang berjalan saat ini, kelemahan sistem yang berjalan, dan analisis sistem yang diinginkan serta analisis kebutuhan sistem.

* + 1. **Analisis Sistem yang Berjalan**

Sistem yang sedang berjalan merupakan sistem yang sedang di pakai selama ini, Sedangkan Analisis Sistem yang sedang berjalan merupakan cara penulis untuk memahami sistem yang sedang berjalan, kalau penulis sudah memahami tentang sistem yang sedang berjalan maka akan terpikirkan oleh penulis mengenai apa yang dibutuhkan untuk membangun sistem yang memudahkan dan membantu sistem yang berjalan saat ini.

Salah satu manfaat dari Analisis yang sedang berjalan adalah untuk mendefinisikan kebutuhan fungsional dari sistem dan mencari tahu kebutuhan apa saja yang di butuhkan oleh pemakai dan kelemahan oleh sistem yang sedang berjalan tersebut.

Berdasarkan penjelasan dari dosen yang menjadi objek penelitian penulis sekaligus aktor yang menjalankan sistem, penulis mendapatkan beberapa objek inti dari penelitian masalah pada universitas dalam membangun *e-learning* dan fitur standar yang dibutuhkan dalam implementasi *e-learning* pada universitas. Berikut ini adalah analisis dari sistem yang sedang berjalan saat ini :

* 1. **Analisis Kegiatan Belajar Mengajar yang sedang Berjalan.**
  2. Prosedur Kegiatan Belajar Mengajar di Kelas.
* Mahasiswa memasuki kelas.
* Dosen mulai memasuki ruangan Kelas dan memberikan *quiz* kepada mahasiswa.
* Jika dosen ingin melaksanakan *quiz* maka mahasiswa mengerjakan *quiz* dan dalam waktu tertentu kemudian ketika waktu selesai maka mahasiswa menyerahkan *quiz* yang telah dikerjakan kepada dosen.
* Jika tidak maka dosen langsung memberikan materi dan latihan kepada mahasiswa.
* Mahasiswa menyimak dan mengerjakan latihan dari dosen.

**C:\Users\Dyah_Nuraeni\Downloads\Flowmap1.png**

**Gambar 3.2** Flowmap Prosedur Kegiatan Belajar Mengajar di Kelas

* 1. Prosedur Kegiatan Belajar Mengajar di Lab
* Mahasiswa memasuki Lab.
* Asisten Dosen memasuki Lab dan memberi latihan dan juga materi
* Mahasiswa menerima materi , menyalakan komputer masing – masing dan mulai mengerjakan latihan yang diberikan Asisten Dosen.
* Asisten dosen memeriksa output latihan mahasiswa di komputer.

**C:\Users\Dyah_Nuraeni\Downloads\Flowmap2.png**

**Gambar 3.3** Flowmap Prosedur Kegiatan Belajar Mengajar di Lab

1. Prosedur pelaksanaan pemberian tugas

* Dosen membuat tugas untuk mahasiswa.
* Kemudian mahasiswa mengerjakan tugas yang diberikan oleh dosen.
* Dosen mengoreksi tugas dan mendapatkan nilai mahasiswa dan diarsipkan.

C:\Users\Dyah_Nuraeni\Downloads\Flowmap3 (1).png

**Gambar 3.4** Flowmap Prosedur pelaksanaan pemberian tugas

1. Prosedur Pelaksanaan *Quiz*.

* Dosen membagikan *quiz* kepada mahasiswa.
* Mahasiswa mengerjakan *quiz* dalam waktu tertentu dan mengumpulkan *quiz* yang telah dikerjakan ketika waktunya sudah selesai.
* Jika *quiz* perlu dibahas maka dosen membahasnya. Tapi jika tidak, maka soal *quiz* di arsipkan

C:\Users\Dyah_Nuraeni\Downloads\Flowmap4.png

**Gambar 3.5** Flowmap Prosedur Pelaksanaan *Quiz*.

1. Prosedur Mengoreksi *Quiz*

* Dosen mengambil soal *quiz* yang telah dikerjakan Mahasiswa dan mulai mengoreksi
* Lalu soal *quiz* dan hasil *quiz* di arsipkan

C:\Users\Dyah_Nuraeni\Downloads\Flowmap5.png

**Gambar 3.6** Flowmap Prosedur Mengoreksi *Quiz*.

1. Hubungan dosen dengan Asisten dosen.

Asumsikan jika Asisten dosen sudah mendapatkan nilai ujian lab, yang didapatkan dari hasil kegiatan belajar di lab.

* Asisten dosen menyerahkan nilai ujian lab kepada dosen.
* Jika menurut dosen, banyak mahasiswa mendapatkan nilai yang bagus maka nilai tersebut akan digabungkan sebagai komponen nilai akhir
* Jika tidak maka dosen tidak memasukkan sebagai komponen tugas akhir. C:\Users\Dyah_Nuraeni\Downloads\Flowma6.png **Gambar 3.7** Flowmap Hubungan dosen dengan Asisten dosen.

1. Prosedur pelaksanaan Asisten dosen membantu mengoreksi *quiz*

* Dosen mengambil *quiz* dan memberikan kepada asisten dosen.
* Asisten dosen mengoreksi soal dan memberi nilai *quiz* yang sudah dikoreksi.
* Dosen mengarsipkan nilai dan soal *quiz*
* Dan dosen juga memasukkan nilai *quiz* ke daftar nilai mahasiswa.

C:\Users\Dyah_Nuraeni\Downloads\Flowma7.png

**Gambar 3.8** Flowmap pelaksanaan Asisten dosen

membantu mengoreksi *quiz*

1. Prosedur pelaksanaan mengamati kemampuan mahasiswa berdasarkan nilai mahasiswa.

* Dosen menilai *quiz*, uts, dan uas.
* Dosen menyorting nilai secara *descending*.

C:\Users\Dyah_Nuraeni\Downloads\Flowma8.png

**Gambar 3.9** Flowmap Pelaksanaan mengamati kemampuan

mahasiswa berdasarkan nilai mahasiswa.

1. Prosedur pelaksanaan mengamati kemampuan mahasiswa berdasarkan pemahaman mahasiswa.

* Dosen memberikan Soal dan Menyuruh Mahasiswa Langsung Mengerjakan Soal di Kelas.
* Mahasiswa langsung mengerjakan soal di papan tulis.

C:\Users\Dyah_Nuraeni\Downloads\Flowma9.png

**Gambar 3.10** Flowmap pelaksanaan mengamat kemampuan mahasiswa berdasarkan pemahaman mahasiswa.

1. Prosedur pelaksanaan mengamati kemampuan mahasiswa secara langsung.

* Dosen mengamati mahasiswa secara langsung, apakah mahasiswa itu aktif atau pasif, dan suka mengajari yang lainnya atau tidak

C:\Users\Dyah_Nuraeni\Downloads\Flowma10.png

**Gambar 3.11** Flowmap pelaksanaan mengamati kemampuan

mahasiswa secara langsung

1. Prosedur penarikan kesimpulan baik buruk mahasiswa dalam

belajar.

* Mengumpulkan variabel dari prosedur 9,10,11
* Jika nilai bagus dan mahasiswa bisa mengerjakan soal dengan baik, dan mahasiswa aktif , suka mengajari temannya. Maka dosen menilai mahasiswa itu bagus
* Jika tidak , maka mahasiswa kurang baik.

C:\Users\Dyah_Nuraeni\Downloads\Flowmap11 (1).png

**Gambar 3.12** Flowmap penarikan kesimpulan baik buruk mahasiswa

**3.2.2 Analisis Kelemahan Sistem yang Sedang Berjalan**

1.Tidak ada modul khusus yang disediakan oleh universitas untuk latihan atau belajar mahasiswa dalam mata kuliah Logika dan Algoritma sehingga arah pembelajaran tidak terarah dengan baik.

2.    Tidak adanya SOP (Standar Operasional Prosedur) khusus untuk Asisten Dosen. Sehingga, asisten dosen tidak terlalu berkonsentrasi ke mahasiswa sebanyak sebagaimana asisten dosen yang membantu dosen dengan adanya SOP (*read*: profesional). Kurangnya *feedback* (tentang personal mahasiswa) yang didapatkan oleh dosen dari asisten dosennya. Sehingga hal tersebut menjadi salah satu faktor dosen tidak mengetahui secara pasti kemampuan masing-masing mahasiswa. Sedangkan untuk mata kuliah lanjutan, Pemrograman Teks 2, sudah tidak ada mahasiswa yang mempunyai kualifikasi untuk menjadi asisten dosen Proteks 2 yang masih tinggal di Pasim. Karena para mahasiswa PUB tahun akhir yang biasanya menjadi asdos, sudah selesai berkuliah. Hal tersebut bisa ditangani dengan meminta orang-orang IT alumni PUB untuk menjadi asisten dosen, tapi itu akan lebih menyusahkan dosen, yaitu mengajari mereka terlebih dahulu untuk mengajar peserta didik.

3.    Kelas laboratorium tidak profesional dan kurangnya fasilitas pendukung.

4.    Tahun demi tahun, penambahan mahasiswa semakin banyak. Hal tersebut membuat ruang kelas semakin terisi. Meskipun kelas Algoritma dan Logika sudah dipecahkan menjadi dua, mahasiswa masih tetap banyak. Akibatnya adalah kurangnya personalisasi pembelajaran terhadap mahasiswa. Semakin banyak mahasiswa dan semakin menurunnya personalisasi pembelajaran, berdampak pada waktu pembelajaran di kelas. Sangat baik dan merupakan metode yang efektif jika personalisasi pembelajaran diterapkan. Tapi jika personalisasi pembelajaran tetap diterapkan di kondisi tadi, akan membutuhkan waktu lebih. Sehingga terkadang sampai ada kelas lain yang harus masuk ke ruang itu karena waktu pembelajaran sudah selesai. Ketika dibutuhkan untuk pindah kelas, tidak ada kelas yang kosong. Selain itu, akibat yang timbul dari personalisasi pembelajaran yang rendah adalah dosen yang tidak mengetahui secara pasti mana mahasiswa *Good Learner*.

**c. Analisis PIECES**

|  |  |  |
| --- | --- | --- |
| **Analisis** | **Sistem Lama** | **Sistem Baru** |
| Kinerja (*Performance*) | * Tahun demi tahun, penambahan mahasiswa semakin banyak. Hal tersebut membuat ruang kelas semakin terisi. Meskipun kelas Algoritma dan Logika sudah dipecahkan menjadi dua, mahasiswa masih tetap banyak. Akibatnya adalah kurangnya personalisasi pembelajaran terhadap mahasiswa. Semakin banyak mahasiswa dan semakin menurunnya personalisasi pembelajaran, berdampak pada kurangnya pengetahuan dosen terhadap personal masing-masing mahasiswa. * Tidak adanya SOP (Standar Operasional Prosedur) khusus untuk Asisten Dosen. Sehingga, asisten dosen tidak terlalu berkonsentrasi ke mahasiswa sebanyak sebagaimana asisten dosen yang membantu dosen dengan adanya SOP (*read*: profesional). * Kurangnya feedback (tentang personal mahasiswa) yang didapatkan oleh dosen dari asisten dosennya. Sehingga hal tersebut menjadi salah satu faktor dosen tidak mengetahui secara pasti kemampuan masing-masing mahasiswa. * Kelas laboratorium tidak profesional dan kurangnya fasilitas pendukung. | Penerapan personalisasi dan adaptif pembelajaran pada sistem aplikasi *E-Learning* , dimana Dosen dapat mengetahui Kemampuan Mahasiswa dari sistem *E-Learning* dan Sistem dapat Mengadaptasikan Latihan sesuai kemampuan Mahasiswa. |
| Informasi (*Information*) | * Kurangnya informasi yang diterima oleh dosen tentang performa dan kemampuan masing-masing mahasiswa. | Sistem dapat merekam data-data interaksi mahasiswa dengan sistem dan dengan adanya data tambahan dimana akan menjadi laporan untuk dosen. |
| Ekonomi (*Economic*) | * Jika dosen mendekati mahasiswa dan/atau bertanya pada asisten dosen untuk mendapatkan informasi dari masing-masing mahasiswa akan memakan biaya yang banyak. Misalnya transportasi, dll. | - Dosen dapat melihat data serta kemampuan Mahasiswa di Sistem apabila mahasiswa telah mengerjakan *Quiz* di Sistem *E-Learning*. |
| Pengendalian (*Control*) |  | - |
| Efisiensi (*Efficiency*) | * Penerapan personalisasi pembelajaran di dalam kelas yang memakan waktu jika hanya dengan seorang dosen dan mahasiswa yang terlampau banyak. * Membutuhkan waktu yang sangat banyak jika dosen mendekati mahasiswa dan/atau bertanya pada asisten dosen untuk mendapatkan informasi dari masing-masing mahasiswa. | Penerapan personalisasi Mahasiswa dan adaptif pembelajaran pada sistem sehingga Dosen dapat melihat data dan Kemampuan Mahasiswa dari Sistem |
| Pelayanan (*Service*) | * Kelas laboratorium kurang profesional dan kurangnya fasilitas pendukung. | Sistem menyediakan latihan-latihan soal dimana akan membantu kelas laboratorium. |

**Tabel 3.1** Analisis PIECES

**3.2.3 Analisis Kebutuhan Sistem**

**3.2.3.1 Analisis Kebutuhan Fungsional**

Dibutuhkan sebuah aplikasi *E-Learning* untuk mendampingi dosen. Dimana aplikasi tersebut memiliki kemampuan sebagai berikut:

* + 1. Menyediakan Soal-soal mata kuliah Logika dan Algoritma dasar untuk belajar mahasiswa
  1. Memberikan *feedback* personal mahasiswa yang akan membantu dosen mengetahui kemampuan personal mahasiswa. Sehingga dosen dapat mengajar lebih baik.
  2. Aplikasi mampu beradaptasi dan bersifat personal terhadap penggunanya.
  3. Menemukan jalur dari *Good Learner*. Sehingga aplikasi bisa mengetahui mana yang *Good Learner* dan merekomendasikan kepada dosen, jalur *Good Learner* mana yang sesuai bagi seorang mahasiswa lainnya.

**3.2.3.2 Analisis Kebutuhan Non Fungsional**

Analisis kebutuhan Non Fungsional yang dibutuhkan untuk membangun aplikasi adalah sebagai berikut :

* 1. **Kebutuhan Perangkat Keras**

Tipe jaringan yang penulis gunakan yaitu jaringan yang berbasis server (*Client-Server*) dan perangkat keras yang dibutuhkan adalah sebagai berikut:

1. Personal Computer(PC), untuk menjalankan aplikasi dengan spesifikasi sebagai berikut:
   * + 1. *RAM* minimal 2GB.
       2. *Harddisk* minimal250GB.
       3. *Processor* Intel *Core* i3 / AMD A4.
2. Personal Computer (PC) untuk server dengan spesifikasi sebagai berikut:
   * + 1. *RAM* minimal 8GB.
       2. *Harddisk* minimal 1TB.
       3. *Processor* Intel *Core* i7 2600
3. Koneksi internet yang bagus.
   1. **Kebutuhan Perangkat Lunak**
4. Netbeans Untuk mengetikkan kode - kode yang dibutuhkan untuk

membuat aplikasi.

1. Xampp, sebagai server database  lokal yang penulis gunakan
2. Browser : Mozila Firefox , Chrome ataupun yang lainnya.
   1. **Kebutuhan Pengguna**

Merupakan penjelasan tentang pengguna yang akan menjalankan aplikasi yang akan di bangun, dan kriteria pengguna tersebut adalah sebagai berikut :

1. Mahasiswa / Dosen.
2. Bisa mengoperasikan komputer.
3. Memiliki pengalaman dan bisa mengoperasikan *web browser*.
4. Mempunyai akses internet.

**BAB IV**

**PERANCANGAN SISTEM**

1. **Perancangan Sistem**

Perancangan sistem merupakan suatu proses pembuatan rancangan bangun suatu sistem yang bertujuan untuk memetakan setiap kebutuhan sistem sehingga proses pembangunannya dapat berjalan dengan baik. Dalam proses ini didefinisikan kebutuhan - kebutuhan fungsional dan teknikal dari sistem yang akan dibangun.

* + 1. **Prosedure Pengaksesan Aplikasi**

1. **Prosedur Kategori *Administrator***

Hak akses administrator diberikan kepada petugas untuk mengelola data Dosen seperti Menambah, mengubah dan menghapus Data Dosen.

1. **Prosedur Kategori Dosen**

Hak akses Dosen adalah mengubah materi apa yang akan dikerjakan oleh Mahasiswa pada *quiz* maupun *Exercise*, materi tersebut adalah *Sequential*, *Conditional* dan *Looping* ,Menentukan jumlah *quiz* yang dikerjakan Mahasiswa dan dapat Melihat data mahasiswa. Dosen juga mempunyai akses untuk mengelola setiap data master antara lain data soal (*question*), data jawaban (*answer*) , menghapus data mahasiswa yang sudah lulus atau keluar , kelola data soal - jawaban (question – answer).

70

1. **Prosedur Kategori Mahasiswa**

Hak Akses bagi Mahasiswa adalah mengerjakan *quiz* dan mengerjakan *exercise*, dan Mengubah Data *profile*.

* 1. **Metode Perancangan Sistem**

Perancangan aplikasi *web* ini menggunakan desain berorientasi objek dengan menggunakan pendekatan *object oriented* method. Untuk membantu proses perancangan sistem berorientasi objek, maka digunakan 4 (empat) diagram yaitu *Use Case Diagram*, *Activity Diagram*, *Sequence Diagram* dan *Class Diagram*.

* + 1. **Use Case Diagram**

**

**Gambar 4.1** *Use Case Diagram*

**Use Case Spesification**

1. **Nama Use Case :** Daftar

**Deskripsi Singkat**

Use Case Daftar dapat diakses oleh Mahasiswa. Mahasiswa yang ingin mengakses Aplikasi *E-Learning* ini harus memiliki akun dengan cara daftar.

**Basic Flow**

1. Ketika aplikasi pertama kali dijalankan maka aplikasi akan menampilkan *form login*.
2. Mahasiswa memilih buat akun
3. Sistem menampilkan *Form* pengisian data untuk membuat akun
4. Mahasiswa mengisi data dan memilih tombol *Submit*
5. Apabila data lengkap maka dan sesuai maka Sistem akan menyimpan data Mahasiswa yang dinputkan

**Alternate Flow:**

* + - 1. Jika mahasiswatidak menginputkan data atau tidak menginputkan data dengan benar.
         1. Sistem meminta mahasiswa untuk mengisi data dengan benar

**Pre-Condition :** Mahasiswa dalam keadaaan *login*.

**Post-Condition :** Data telah dimanipulasi

**Requirement :** Mahasiswa telah *login* terlebih dahulu

1. **Nama Use Case :** *Login*

**Deskripsi Singkat**

Use Case *Login* dapat diakses oleh *Admin*, Dosen dan Mahasiswa. Namun menunya berbeda untuk masing-masing *user*.

**Basic Flow**

1. Ketika aplikasi pertama kali dijalankan maka aplikasi akan menampilkan *form login*.
2. *User* mengisi *username* dan *password*.
3. *Username* dan *password* valid, sistem menampilkan halaman utama dengan beberapa menu.

**Alternate Flow :** Jika *username* dan *password* tidak valid maka

pengguna tidak dapat mengakses sistem.

**Pre-Condition :** Pengguna dalam keadaan logout.

**Post-Condition :** Tampil halaman utama.

**Requirement :** Memiliki data *User* untuk *login*

1. **Nama *Use Case* :** Mengerjakan *Quiz*

**Deskripsi Singkat**

Use case ini digunakan oleh Mahasiswa dalam mengerjakan Mengerjakan. Kegiatannya adalah Mengerjakan dan *Submit* *Quiz.*

***Basic Flow***

**a) Mengerjakan dan *Submit* soal**

(1) Mahasiswa membuka halaman utama

* + 1. Sistem menampilkan halaman utama untuk Mahasiswa
    2. Mahasiswa memilih menu *Quiz*
    3. Sistem menampilkan soal yang memiliki jumlah sama setiap levelnya. Contoh 3 soal level *low*(H), 3 soal level *medium*(M), 3 soal level *high*(H)
    4. Mahasiswa mengerjakan soal dan tidak diperbolehkan kembali ke soal sebelumnya
    5. Setiap mahasiswa melanjutkan soal
    6. Sistem akan menyimpan soal yang telah dijawab mahasiswa dengan jawaban yang benar
    7. Mahasiswa mensubmit soal apabila telah selesai menjawab semua soal
    8. Mahasiswa akan memiliki nilai yang tergolong *Poor* (nilai>=0 dan nilai<=33) , *Good* (nilai>33 dan nilai<=66), dan *Fair* (nilai>66 dan nilai <100)
    9. Data akan tersimpan
    10. Kemudian jika sebelumnya terdapat data nilai maka akan di bandingkan. Jika selisih lebih besar maka akan dijadikan *Good Learner*
    11. Soal yang tersimpan akan digunakan kembali oleh Mahasiswa yang mendapatkan *Good Learner* tersebut sehingga dapat mengikuti *Good Learner* tersebut.

**Alternate Flow :** Tidak ada.

**Pre-Condition :** Mahasiswa dalam keadaaan *login.*

**Post-Condition :** *Quiz* telah di*submit*.

**Requirement :** Mahasiswa telah *login* terlebih dahulu

1. **Nama *Use Case* :** Mengerjakan *Exercise*

**Deskripsi Singkat**

*Use case* ini digunakan oleh Mahasiswa dalam mengerjakan *Random Exercise*. Kegiatannya adalah Mengerjakan dan *Submit Random Exercise*.

**Basic Flow**

**a) Mengerjakan dan *Submit* soal**

* 1. Mahasiswa membuka halaman utama
  2. Sistem menampilkan Halaman Utama
  3. Mahasiswa membuka Menu *Exercise* sesuai dengan pembahasan yang sedang dipelajari
  4. Sistem menampilkan soal yang telah dirandom
  5. Mahasiswa mengerjakan soal
  6. Jika mahasiswa telah selesai mengerjakan soal maka mahasiswa men*submit* jawaban
  7. Sistem menampilkan jawaban yang benar dari soal yang telah dijawab,
  8. Apabila mahasiswa telah selesai menjawab soal maka Sistem akan mentotal jawaban yang benar dan menampilkan nilai yang diperoleh mahasiswa

**Alternate Flow :** Tidak ada.

**Pre-Condition :** Mahasiswa dalam keadaaan *login*.

**Post-Condition :** Jawaban telah di submit.

**Requirement :** Mahasiswa telah *login* terlebih dahulu

1. **Nama Use Case :** Perekaman Soal

**Deskripsi Singkat**

Use case ini digunakan untuk Menemukan *Good learner* bagi mahasiswa yang memiliki nilai dibawah.

**Basic Flow**

1. Mahasiswa membuka Halaman Soal *Quiz*
2. Sistem menampilkan Soal
3. Mahasiswa menjawab soal dan selanjutnya mengklik tombol lanjut (*Next*)
4. Sistem akan menyimpan soal yang telah dijawab mahasiswa dengan jawaban yang benar
5. Mahasiswa men*submit* soal apabila telah selesai menjawab semua soal
6. Mahasiswa akan memiliki nilai yang tergolong *Poor* (nilai>=0 dan nilai<=33) , *Good* (nilai>33 dan nilai<=66), *Fair* (nilai>66 dan nilai <100)

**Alternate Flow :** Tidak ada.

**Pre-Condition :** Mahasiswa dalam keadaaan *login*.

**Post-Condition :** Sistem menampilkan data Nilai

**Requirement :** Mahasiswa telah *login* terlebih dahulu

1. **Nama Use Case :** Model *Good Learner*

**Deskripsi Singkat**

Use case ini adalah proses bagaimana mendapatkan *Good Learner*.

**Basic Flow**

1. Mahasiswa membuka Soal
2. Sistem akan menampilkan halaman Soal, dimana soal yang akan dijawab oleh Mahasiswa tersebut adalah Soal dari *Good Learner* yang dipilih untuk mahasiswa tersebut.
3. Mahasiswa menjawab soal dengan mengklik tombol lanjut (*Next*)
4. Sistem akan menyimpan soal yang telah dijawab mahasiswa dengan jawaban yang benar
5. Mahasiswa mensubmit soal apabila telah selesai menjawab semua soal
6. Mahasiswa akan memiliki nilai yang tergolong *Poor* (nilai>=0 dan nilai<=33) , *Good* (nilai>33 dan nilai<=66), dan *Fair* (nilai>66 dan nilai <100)
7. Data akan tersimpan
8. Kemudian jika sebelumnya terdapat data nilai maka akan di bandingkan. Jika selisih lebih besar maka akan dijadikan *Good Learner*
9. Soal yang tersimpan akan digunakan kembali oleh Mahasiswa yang mendapatkan *Good Learner* tersebut sehingga dapat mengikuti *Good Learner* tersebut.

**Alternate Flow :** Tidak ada.

**Pre-Condition :** Mahasiswa dalam keadaaan *login*.

**Requirement :** Mahasiswa telah *login* terlebih dahulu

1. **Nama Use Case :** Kelola Profil

**Deskripsi Singkat**

Use case ini digunakan oleh *Admin*, Dosen, dan Mahasiswa untuk membuat Akun dan mengedit data.

**Basic Flow**

1. Mahasiswa telah *Log in*
2. Sistem menampilkan menu sesuai *role* nya
3. Mahasiswa memilih *Setting*
4. Sistem menampilkan *form* Edit akun
5. Mahasiswa mengubah data dan kemudian mensubmit
6. Sistem mengubah perubahan yang dilakukan *user*

**Alternate Flow:**

1) Jika Mahasiswa tidak menginputkan data atau tidak menginputkan data dengan benar.

* + - * 1. Sistem meminta Mahasiswa untuk mengisi data dengan benar

**Pre-Condition :** Mahasiswa dalam keadaaan *login*.

**Post-Condition :** Data telah dimanipulasi

**Requirement :** Mahasiswa telah *login* terlebih dahulu

1. **Nama Use Case :** Kelola Data *Question*

**Deskripsi Singkat**

Use case ini digunakan oleh Dosen untuk mengelola *Question* / Soal yang mana meliput Soal *Quiz*, dan Soal *Exercise*.

**Basic Flow**

* 1. **Membuat *Question***
     1. Dosen membuka Menu *Question*
     2. Sistem menampilkan Data *Question*
     3. Dosen memilih Tambah *Question*
     4. Sistem menampilkan *form* Tambah
     5. Dosen menginputkan data soal *Question* dan apabila sudah selesai memilih tombol *Submit*
     6. Sistem menyimpan data Soal
  2. **Mengedit *Question***

1. Dosen membuka Menu *Question* pada menu Master Data.
2. Sistem menampilkan Data *Question*
3. Dosen memilih tombol Edit Data *Question*
4. Sistem menampilkan *form* Ubah yang berisi data yang dipilih *Admin*
5. Dosen mengedit data soal yang dipilih dan tombol *Submit*
6. Apabila data telah lengkap dan sesuai maka Sistem akan menyimpan data Soal
   1. **Hapus Data**
      1. Dosen membuka Menu *Question* pada menu Master Data.
      2. Sistem menampilkan Data *Question*
      3. Dosen mengklik tombol hapus.
      4. Sistem menghapus data *Question* yang dipililh

**Alternate Flow :**

1. Jika Dosentidak menginputkan data atau tidak menginputkan data dengan benar.
2. Sistem meminta Dosen untuk mengisi data dengan benar

**Pre-Condition :** Dosen dalam keadaaan *login*.

**Post-Condition :** Data telah dimanipulasi

**Requirement :** Dosen telah *login* terlebih dahulu.

1. **Nama Use Case :** Kelola Data *Answer*

**Deskripsi Singkat**

Use case ini digunakan oleh Dosen untuk mengelola *Answer* / Jawaban yang mana meliputi *Answer* dari Soal *Quiz*, dan Soal *Exercise*.

* 1. **Membuat *Answer***
     1. Dosen membuka Menu *Answer*
     2. Sistem menampilkan Data *Answer*
     3. Dosen memilih Tambah *Answer*
     4. Sistem menampilkan *form* Tambah
     5. Dosen menginputkan data soal *Answer* dan memilih tombol *Submit*
     6. Sistem akan menyimpan Data *Answer*
  2. **Mengedit *Answer***
     1. Dosen membuka Menu *Answer* pada menu Master Data.
     2. Sistem menampilkan Data *Answer*
     3. Dosen memilih tombol *Edit* Data *Answer*
     4. Sistem menampilkan *form* Ubah yang berisi data yang dipilih oleh Dosen
     5. Dosen mengedit soal data yang dipilih dan memilih tombol Submit
     6. Apabila data telah benar dan sesuai maka Sistem menyimpan perubahan data *Answer*
  3. **Hapus Data**
     1. Dosen membuka Menu *Answer* pada menu Master Data.
     2. Sistem menampilkan Data *Answer*
     3. Dosen mengklik tombol hapus.
     4. Sistem menghapus data *Answer* yang dipilih

**Alternate Flow :**

* + - 1. Jika Dosentidak menginputkan data atau tidak menginputkan data dengan benar.
      2. Sistem meminta Dosen untuk mengisi data dengan benar

**Pre-Condition :** Dosen dalam keadaaan *login*.

**Post-Condition :** Data telah dimanipulasi

**Requirement :** Dosen telah *login* terlebih dahulu.

1. **Nama Use Case :** Kelola Data *Question* *Answer*

**Deskripsi Singkat**

Use case ini digunakan oleh Dosen untuk mengelola *Question Answer* (Soal Jawaban). Di *Form* ini Dosen akan mencocokan Soal dengan Jawaban yang telah di buat di Menu *Question* dan *Answer.*

**Basic Flow**

* 1. **Mencocokkan Soal dan Jawaban**
     + 1. Dosen membuka Menu *Question - Answer*
       2. Sistem menampilkan Data *Question - Answer*
       3. Dosen memilih Tambah *Question - Answer*
       4. Sistem menampilkan *form* *Question - Answer*
       5. Dosen memilih *Question - Answer* yang sesuai dan inputkan jawaban yang benar serta data yang lainnya dan memilih tombol *Submit*
       6. Apabila sudah sesuai maka Sistem menyimpan data *Question - Answer*
  2. **Mengedit Soal dan Jawaban**
     + 1. Dosen membuka Menu *Question - Answer* pada menu Master Data.
       2. Sistem menampilkan Data *Question – Answer*
       3. Dosen memilih tombol Edit Data *Question – Answer*
       4. Sistem menampilkan *form* Ubah yang berisi data yang dipilih Dosen
       5. Dosen mengedit soal data yang dipilih dan memilih tombol *Submit*
       6. Apabila sudah sesuai maka Sistem menyimpan data *Question - Answer*
  3. **Hapus Data**

(1) Dosen membuka Menu *Question - Answer* pada menu Master Data.

* + 1. Sistem menampilkan Data *Question – Answer*
    2. Dosen mengklik tombol hapus.
    3. Sistem menghapus data *Question - Answer* yang dipililh

**Alternate Flow :**

* + - 1. Jika Dosentidak menginputkan data atau tidak menginputkan data dengan benar.
         1. Sistem meminta Dosen untuk mengisi data dengan benar

**Pre-Condition :** Dosen dalam keadaaan *login*.

**Post-Condition :** Data telah dimanipulasi

**Requirement :** Dosen telah *login* terlebih dahulu

1. **Nama Use Case :** Melihat Data Mahasiswa

**Deskripsi Singkat**

Use case ini digunakan oleh Dosen untuk melihat semua data Mahasiswa dan Nilai.

**Basic Flow**

* + - 1. Dosen membuka Menu Lihat Data Mahasiswa
      2. Sistem menampilkan Data Mahasiswa dan Nilai yang diperoleh
      3. Dosen memilih Mahasiswa
      4. Sistem menampilkan *Profile* dari mahasiswa yang dipilih

**Alternate Flow :** Tidak ada.

**Pre-Condition :** Dosen dalam keadaaan *login*.

**Post-Condition :** Data telah tampil

**Requirement :** Dosen telah *login* terlebih dahulu

1. **Nama Use Case :** Menghapus Data Mahasiswa

**Deskripsi Singkat**

*Use case* ini digunakan oleh Dosen untuk menghapus mahasiswa, jika mahasiswa tidak digunakan lagi dikarenakan mahasiswa telah lulus / keluar.

**Basic Flow**

1. Dosen memilih Menu Lihat Data Mahasiswa pada menu Dosen
2. Sistem *form* Data Mahasiswamemuat dan menampilkan data-data Mahasiswa dan Hasil *Quiz* yang terdapat dalam sistem.
3. Dosen mengklik nama Mahasiswa yang dipilih
4. Sistem menampilkan keseluruhan data Mahasiswa yang dipilih
5. Dosen memilih Hapus Data Mahasiswa. Hapus ini dilakukan apabila mahasiswa keluar dari Universitas atau Lulus
6. Sistem menghapus data Mahasiswa yang dipilih

**Alternate Flow :** Tidak ada.

**Pre-Condition :** Dosen dalam keadaaan *login*.

**Post-Condition :** DataMahasiswa sudah dimanipulasi.

**Requirement :** Dosen telah *login* terlebih dahulu.

1. **Nama Use Case :** Mengubah Jumlah Banyak Soal Per Level *Quiz*

**Deskripsi Singkat**

Use case ini digunakan oleh Dosen untuk melihat semua data Mahasiswa dan Nilai.

**Basic Flow**

* 1. Dosen membuka Menu Mengubah Banyak Soal Per Level *Quiz*
  2. Sistem menampilkan Menu Mengubah Banyak Soal Per Level *Quiz*
  3. Dosen memilih Jumlah soal per Level dan memilih Tombol Selesai
  4. Sistem menyimpan jumlah soal yang akan tampil di Latihan *Quiz*

**Alternate Flow :** Tidak ada.

**Pre-Condition :** Dosen dalam keadaaan *login*.

**Post-Condition :** Data telah dimanipulasi

**Requirement :** Dosen telah *login* terlebih dahulu

1. **Nama Use Case :** Mengubah Materi

**Deskripsi Singkat**

Use case ini digunakan oleh Dosen mengubah materi untuk Soal yang di akan muncul di menu *Quiz* dan *Exercise* untuk mahasiswa

**Basic Flow**

* 1. Dosen membuka Menu Status Materi
  2. Sistem menampilkan Menu Status Materi
  3. Dosen memilih Materi dan memilih Tombol Selesai
  4. Sistem menyimpan materi

**Alternate Flow :** Tidak ada.

**Pre-Condition :** Dosen dalam keadaaan *login*.

**Post-Condition :** Data telah dimanipulasi

**Requirement :** Dosen telah *login* terlebih dahulu

**Post-Condition :** Data mahasiswa tersimpan

**Requirement :** Mahasiswa telah *login* terlebih dahulu

1. **Nama Use Case :** Kelola Data Dosen

**Deskripsi Singkat**

Use case ini digunakan oleh *Admin* untuk mengelola Data Dosen, seperti membuat akun *Admin*, mengubah dan Menghapus Data Dosen.

**Basic Flow**

**a) Membuat Data Dosen**

* + 1. *Admin* membuka Menu Kelola DataDosen
    2. Sistem menampilkan Data Dosen
    3. *Admin* memilih Tambah Dosen
    4. Sistem menampilkan *form* Tambah
    5. *Admin* menginputkan data soal Dosen dan memilih tombol *Submit*
    6. Sistem akan menyimpan Data Dosen
  1. **Mengedit Data Dosen**
  2. *Admin* membuka Menu Kelola Data Dosen
  3. Sistem menampilkan Data Dosen
  4. *Admin* memilih tombol Edit
  5. Sistem menampilkan *form* Ubah yang berisi data yang dipilih *Admin*
  6. *Admin* mengedit data Dosen yang dipilih dan tombol *Submit*
  7. Apabila data telah lengkap dan sesuai maka Sistem akan menyimpan data Dosen
  8. **Hapus Data**

(1) *Admin* membuka Menu Kelola Data Dosen

(2) Sistem menampilkan Data Dosen

(3) *Admin* mengklik tombol hapus.

(4) Sistem menghapus Data Dosen yang dipililh

**Alternate Flow :**

* + - 1. Jika *Admin* tidak menginputkan data atau tidak menginputkan data dengan benar.
         1. Sistem meminta Dosen untuk mengisi data dengan benar

**Pre-Condition :** *Admin* dalam keadaaan *login*.

**Post-Condition :** Data telah dimanipulasi

**Requirement :** *Admin* telah *login* terlebih dahulu.

* + 1. **Activity Diagram**
       1. **Activity – Daftar**

****

**Gambar 4.2** *Activity Diagram Daftar*

* + - 1. **Activity - *Login***

****

**Gambar 4.3** *Activity Diagram Login*

* + - 1. **Activity – Mengerjakan *Quiz***

****

**Gambar 4.4** *Activity Diagram* Mengerjakan *Quiz*

* + - 1. **Activity – Mengerjakan *Exercise***

**Gambar 4.5** *Activity Diagram* Mengerjakan *Exercise*

* + - 1. **Activity – Perekaman Soal**

****

**Gambar 4.6** *Activity Diagram* Perekaman Soal

* + - 1. **Activity – Model *Good Learner***

****

[

]

]

[

**Gambar 4.7** *Activity Diagram Model Good Learner*

* + - 1. **Acitivity – Kelola Profil**

****

**Gambar 4.8** *Activity Diagram* Kelola Profil

* + - 1. **Activity – Kelola Data *Question***

****

[Berhasil]

[Gagal]

**Gambar 4.9** *Activity Diagram* KelolaData *Question*

* + - 1. **Activity – Kelola Data *Answer***

****

[Berhasil]

[Gagal]

**Gambar 4.10** *Activity Diagram* KelolaData *Answer*

* + - 1. **Activity – KelolaData *Question Answer***

****

[Gagal]

[Berhasil]

**Gambar 4.11** *Activity Diagram* KelolaData *Question - Answer*

* + - 1. **Acitivity – Melihat Data Mahasiswa**

****

**Gambar 4.12** *Activity Diagram* Melihat Data Mahasiswa

* + - 1. **Activity – Menghapus Data Mahasiswa**

****

**Gambar 4.13** *Activity Diagram* Menghapus DataMahasiswa

* + - 1. **Acitivity – Mengubah Materi**

**

**Gambar 4.14** *Activity Diagram* Mengubah Materi

* + - 1. **Acitivity – Mengubah Jumlah Banyak Jumlah Soal Per Level**

**

**Gambar 4.15** *Activity Diagram* Mengubah Banyak Jumlah Soal *Perlevel*

**4.2.2.15 Activity – Kelola Data Dosen**

****

**Gambar 4.16** *Activity Diagram* Kelola Data Dosen

**4.2.2.16 Activity – Jalankan Fungsi**

****

**Gambar 4.17** *Activity Diagram* Jalankan fungsi

**4.2.2.17 Activity – Penilaian *Good Learner***



**Gambar 4.18** *Activity Diagram* Penilaian *Good Learner*

* + 1. **Sequence Diagram**
       1. **Sequence Diagram – Daftar**



:Mahasiswa

**Gambar 4.19** *Sequence Diagram* Daftar

* + - 1. **Sequence Diagram – *Login***

****

**Gambar 4.20** *Sequence Diagram Login*

* + - 1. **Sequence Diagram – Mengerjakan *Quiz***

**

**Gambar 4.21** *Sequence Diagram* Mengerjakan *Quiz*

* + - 1. **Sequence Diagram – Mengerjakan *Exercise***

**

**Gambar 4.22** *Sequence Diagram* Mengerjakan *Exercise*

* + - 1. **Sequence Diagram – Perekaman Soal**

**

**Gambar 4.23** *Sequence Diagram* Perekaman Soal

* + - 1. **Sequence Diagram – Model *Good Learner***

****

**Gambar 4.24** *Sequence Diagram* Model *Good Learner*

* + - 1. **Sequence Diagram – Kelola Profil : Ubah**



**Gambar 4.25** *Sequence Diagram* Kelola Profil : Ubah

* + - 1. **Sequence Diagram – Kelola Data *Question* : Tambah**



:Dosen

**Gambar 4.26** *Sequence Diagram* KelolaData *Question*: Tambah

* + - 1. **Sequence Diagram – Kelola Data *Question* : Ubah**



:Dosen

**Gambar 4.27** *Sequence Diagram* KelolaData *Question*: Ubah

* + - 1. **Sequence Diagram – Kelola Data *Question*: Hapus**

**Gambar 4.28** *Sequence Diagram* Kelola Data *Question* :Hapus

:Dosen

* + - 1. **Sequence Diagram – Kelola Data *Answer* : Tambah**



:Dosen

**Gambar 4.29** *Sequence Diagram* Kelola Data *Answer* : Tambah

* + - 1. **Sequence Diagram – Kelola Data *Answer*: Ubah**

****

:Dosen

**Gambar 4.30** *Sequence Diagram* Kelola Data *Answer*: Ubah

* + - 1. **Sequence Diagram – Kelola Data *Answer*: Hapus**



:Dosen

**Gambar 4.31** *Sequence Diagram* Kelola Data *Answer*: Hapus

* + - 1. **Sequence Diagram – Kelola *Question Answer* : Tambah**



:Dosen

**Gambar 4.32** *Sequence Diagram* Kelola *Question Answer:* Tambah

* + - 1. **Sequence Diagram – Kelola *Question Answer* : Ubah**



:Dosen

**Gambar 4.33** *Sequence Diagram* Kelola *Question Answer*: Ubah

* + - 1. **Sequence Diagram – Kelola *Question Answer*: Hapus**



:Dosen

**Gambar 4.34** *Sequence Diagram* Kelola *Question Answer*: Hapus

* + - 1. **Sequence Diagram – Melihat Data Mahasiswa**

**

**Gambar 4.35** *Sequence Diagram* Melihat Data Mahasiswa

* + - 1. **Sequence Diagram – Menghapus Data Mahasiswa**



:Dosen

dD

**Gambar 4.36** *Sequence Diagram* Menghapus Data Mahasiswa

* + - 1. **Sequence Diagram – Mengubah Materi**



**Gambar 4.37** *Sequence Diagram* Mengubah Materi

* + - 1. **Sequence Diagram – Mengubah Jumlah Banyak Soal Per Level**



**Gambar 4.38** *Sequence Diagram* Mengubah Jumlah Banyak Soal Per Level

* + - 1. **Sequence Diagram – Kelola Data Dosen : Tambah**



:Admin

**Gambar 4.39** *Sequence Diagram* Kelola Data Dosen :Tambah

* + - 1. **Sequence Diagram – Kelola Data Dosen : Ubah**



:Admin

**Gambar 4.40** *Sequence Diagram* Kelola Data Dosen :Ubah

* + - 1. **Sequence Diagram – Kelola Data Dosen : Hapus**



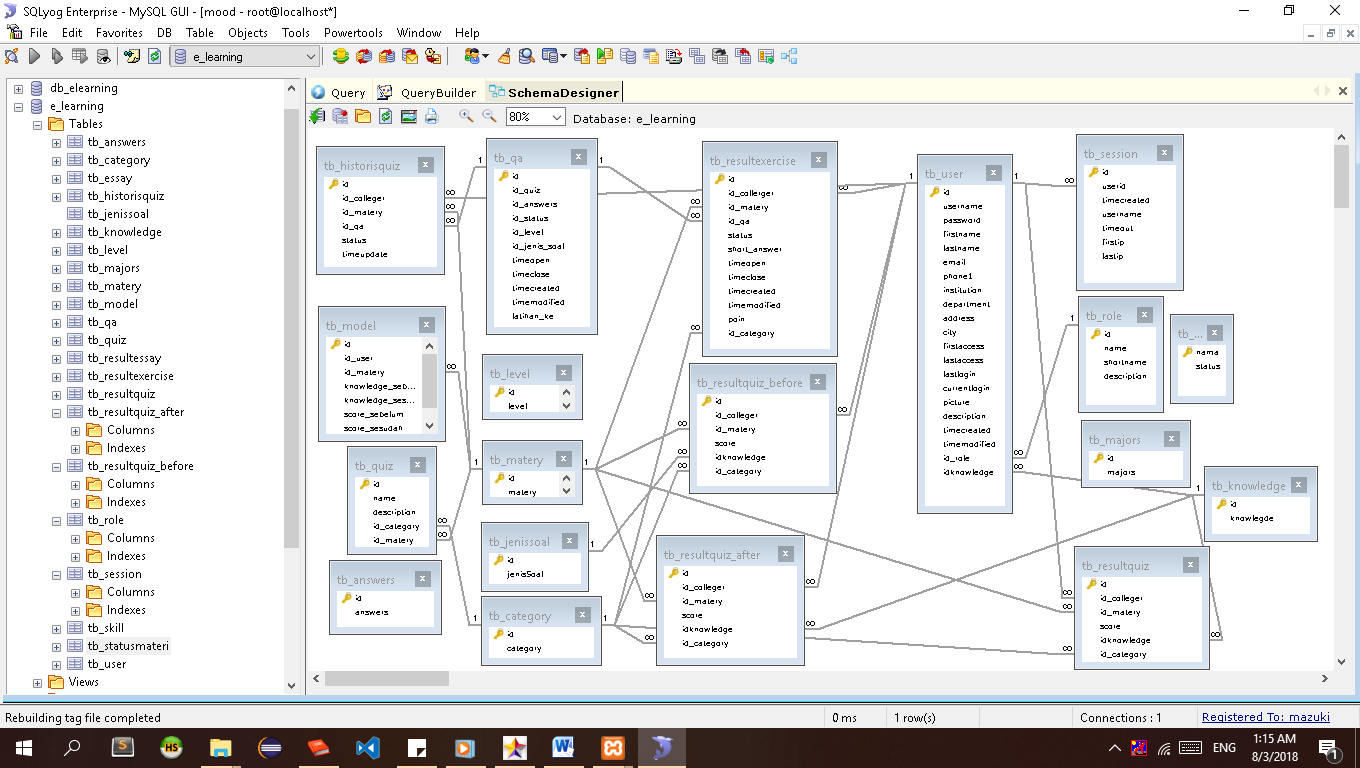
**Gambar 4.41** *Sequence Diagram* Kelola Data Dosen :Hapus

* + 1. **Class Diagram**



**Gambar 4.42** *Class Diagram* Sistem *E-Learning*

* 1. **Perancangan Basis Data**
     1. **Skema Relasi Basis Data**



**Gambar 4.43** *Schema* Sistem *E-Learning*

* + 1. **Deskripsi Tabel**
  1. **Tabel 1 – tb\_user**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT |  | PRIMARY KEY,  AUTO INCREMENT |
| Username | VARCHAR | 255 | - |
| Password | VARCHAR | 255 | - |
| Firstname | VARCHAR | 50 | - |
| Lastname | VARCHAR | 50 | - |
| Phone1 | BIGINT | 13 | - |
| Email | VARCHAR | 50 | - |
| Idrole | INT | 3 | - |
| Idmayor | INT | 3 | - |
| TimeOpen | TIME | - | - |
| TimeClose | TIME | - | - |
| Institution | VARCHAR | 255 | - |
| Department | VARCHAR | 255 | - |
| Address | VARCHAR | 255 | - |
| City | VARCHAR | 255 | - |
| FirstAccess | TIME | - | - |
| LastAccess | TIME | - | - |
| Currentlogin | TIME | - | - |
| Lastlogin | TIME | - | - |
| Picture | Blob | - | - |
| Description | TEXT | - | - |
| Timecreated | TIME | - | - |
| IdKnowledge | INT | - | - |

**Tabel 4.1** Deskripsi Tabel tb\_user

* 1. **Tabel 2 – tb\_session**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | BIGINT | 10 | PRIMARY KEY,  AUTO INCREMENT |
| Iduser | INT | 3 | - |
| Username | VARCHAR | 255 | - |
| Timecreated | BIGINT | 10 | - |
| Timeout | BIGINT | 10 | - |
| Firstip | VARCHAR | 45 | - |
| Lastip | VARCHAR | 45 | - |

**Tabel 4.2** Deskripsi Tabel tb\_session

* 1. **Tabel 3 – tb\_category**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Category | VARCHAR | 100 | - |

**Tabel 4.3** Deskripsi Tabel tb\_category

* 1. **Tabel 4 – tb\_role**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTOINCREMENT |
| Role | BIGINT | 10 | - |

**Tabel 4.4** Deskripsi Tabel tb\_role

* 1. **Tabel 5 – tb\_answer**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Answer | TEXT | - | - |

**Tabel 4.5** Deskripsi Tabel tb\_answer

* 1. **Tabel 6 – tb\_matery**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Matery | VARCHAR | 100 | - |

**Tabel 4.6** Deskripsi Tabel tb\_matery

* 1. **Tabel 7 – tb\_knowledge**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Knowledge | VARCHAR | 50 | - |

**Tabel 4.7** Deskripsi Tabel tb\_knowledge

* 1. **Tabel 8 – tb\_mayor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Mayor | VARCHAR | 255 | - |

**Tabel 4.8** Deskripsi Tabel tb\_mayor

* 1. **Tabel 9 – tb\_level**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | CHAR | 10 | PRIMARY KEY,  AUTO INCREMENT |
| Level | CHAR | 2 | - |

**Tabel 4.9** Deskripsi Tabel tb\_level

* 1. **Tabel 10 – tb\_jenissoal**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | CHAR | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Jenissoal | VARCHAR | 255 | - |

**Tabel 4.10** Deskripsi Tabel tb\_jenissoal

* 1. Tabel 11 – **tb\_quiz**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Name | VARCHAR | 255 | - |
| IdMatery | INT | 3 | FOREIGN KEY  FK\_tb\_quiz\_matery |
| IdCategory | INT | 3 | FOREIGN KEY  FK\_tb\_quiz\_category |
| Description | TEXT | - | - |

**Tabel 4.11** Deskripsi Tabel tb\_question

* 1. Tabel 12 – **tb\_model**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Iduser | INT | 3 | FOREIGN KEY  FK\_tb\_model\_usr |
| Knowledge\_sebelum | INT | 3 | FOREIGN KEY  FK\_tb\_model |
| Knowledge\_sesudah | INT | 3 | FOREIGN KEY  FK\_tb\_model\_user |
| Score\_sebelum | INT | 3 | - |
| Score\_sesudah | INT | 3 | - |
| Idmatery | INT | 3 | FOREIGN KEY  FK\_tb\_model\_matery |

**Tabel 4.12** Deskripsi Tabel tb\_model

* 1. Tabel 13 – **tb\_QA**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Idquiz | INT | 3 | FOREIGN KEY  FK\_tb\_qa\_quiz |
| Idanswer | INT | 3 | FOREIGN KEY  FK\_tb\_qa\_answer123 |
| Id\_Status | INT | 3 | FOREIGN KEY  FK\_tb\_qa\_status |
| Idlevel | INT | 3 | FOREIGN KEY  FK\_tb\_qa\_level |
| Id\_jenissoal | INT | 3 | FOREIGN KEY  FK\_tb\_qa\_jns |
| Timeopen | TIME | - | - |
| Timeclose | TIME | - | - |
| Timecreated | TIME | - | - |
| Timemodified | TIME | - | - |
| Latihan Ke | INT | 3 | - |

**Tabel 4.13** Deskripsi Tabel tb\_qa

* 1. Tabel 14 – **tb\_resultquiz**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Id\_colleger | INT | 3 | FOREIGN KEY  FK\_tb\_resultquiz\_user |
| Idmatery | INT | 3 | FOREIGN KEY  FK\_matery |
| Score | DOUBLE |  |  |
| IdKnowledge | INT | 3 | FOREIGN KEY  FK\_tb\_resultquiz |
| Idcategory | INT | 3 | FOREIGN KEY  FK\_tb\_crq |

**Tabel 4.14** Deskripsi Tabel tb\_resultquiz

* 1. Tabel 15 – **tb\_historiquiz**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Idcolleger | INT | 3 | FOREIGN\_KEY  FK\_tb\_historiquiz\_user |
| Idmatery | INT | 3 | FOREIGN\_KEY  FK\_tb\_historiquiz\_matery |
| Idqa | INT | 3 | FOREIGN\_KEY  FK\_tb\_historiquiz\_qa |
| Status | INT | 3 | - |
| Timeupdate | TIME | - | - |

**Tabel 4.15** Deskripsi Tabel tb\_historiquiz

* 1. Tabel 16 – **tb\_resultexercise**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Idcolleger | INT | 3 | FOREIGN\_KEY  FK\_tb\_resultexercise\_user |
| Idmatery | INT | 3 | FOREIGN\_KEY  FK\_tb\_resultexercise\_matery |
| Idqa | INT | 3 | FOREIGN\_KEY  FK\_tb\_resultexercise\_qa |
| Status | INT | 3 | - |
| Short\_answer | VARCHAR | 100 |  |
| Timecreated | TIME | - | - |
| Timemodified | TIME | - | - |
| TimeOpen | TIME | - | - |
| TimeClose | TIME | - | - |
| Idcategory | INT | 3 | - |

**Tabel 4.16** Deskripsi Tabel tb\_resultexercise

* 1. Tabel 17 – **tb\_resultquiz\_after**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Idcolleger | INT | 3 | FOREIGN KEY  FK\_tb\_rqa\_colleger |
| Id\_matery | INT | 3 | FOREIGN KEY  FK\_tb\_rqa\_matery |
| Score | INT | 3 | - |
| IdKnowledge | INT | 3 | FOREIGN KEY  FK\_tb\_rqak |
| IdCategory | INT | 3 | FOREIGN KEY  FK\_tb\_rqa\_category |

**Tabel 4.17** Deskripsi Tabel tb\_skill

* 1. Tabel 18 – **tb\_resultquiz\_before**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Id | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Idcolleger | INT | 3 | FOREIGN KEY  FK\_tb\_rqa\_colleger |
| Id\_matery | INT | 3 | FOREIGN KEY  FK\_tb\_rqa\_matery |
| Score | INT | 3 | - |
| IdKnowledge | INT | 3 | FOREIGN KEY  FK\_tb\_rqak |
| IdCategory | INT | 3 | FOREIGN KEY  FK\_tb\_rqa\_category |

**Tabel 4.18**  Deskripsi Tabel tb\_resultquiz\_before

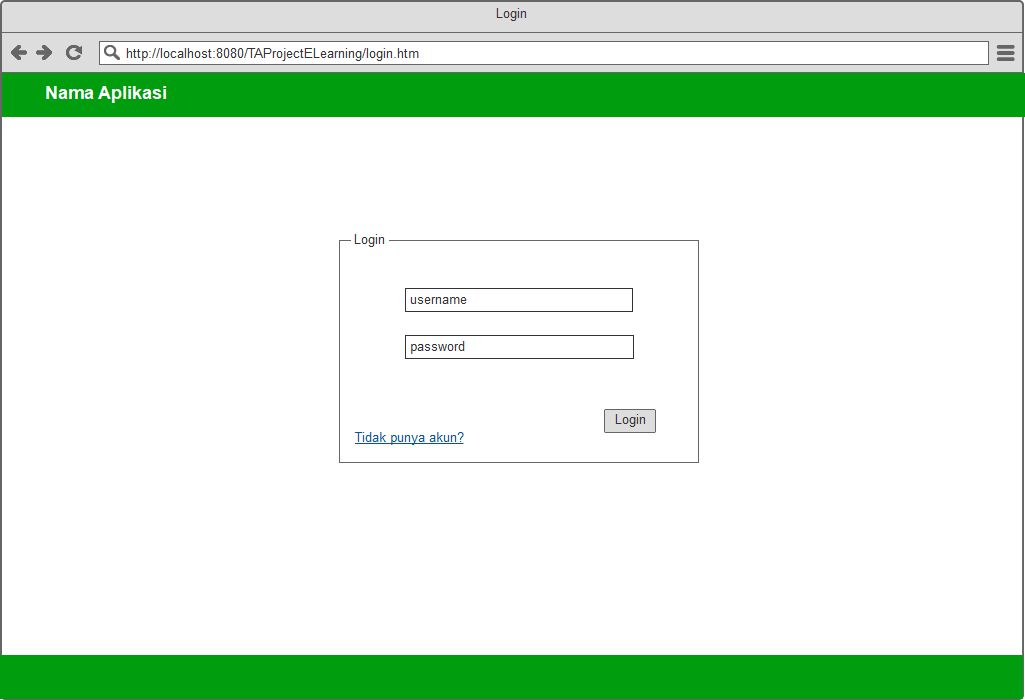
* 1. Tabel 19 – **tb\_statusmatery**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | **Tipe Data** | **Panjang** | **Keterangan** |
| Status | INT | 3 | PRIMARY KEY,  AUTO INCREMENT |
| Nama | VARCHAR | - | - |

**Tabel 4.19** Deskripsi Tabel tb\_statusmatery

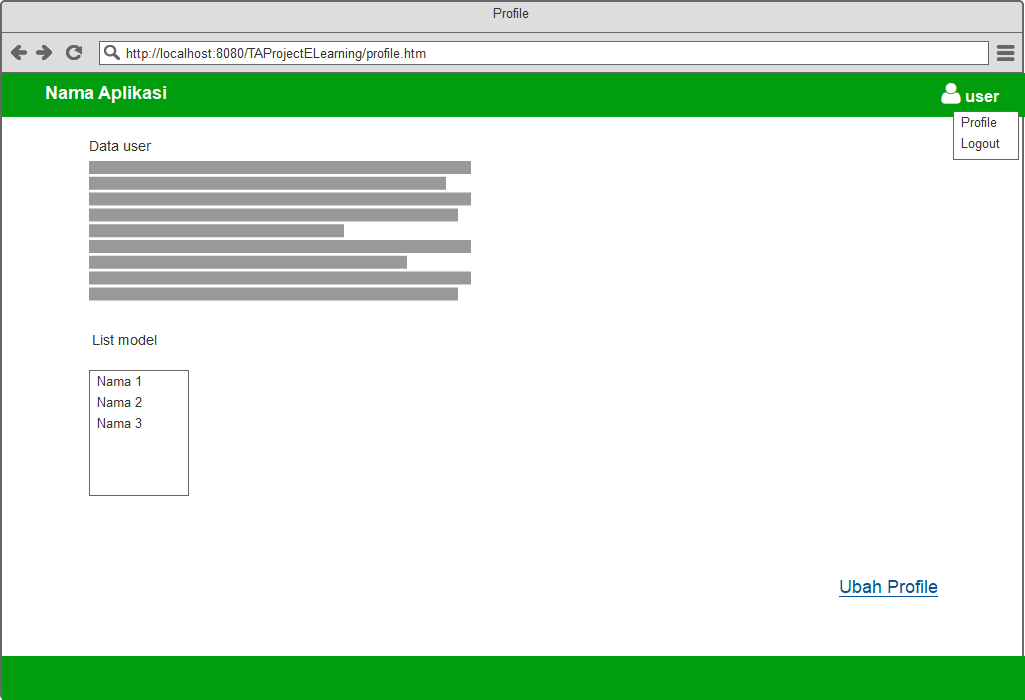
* 1. **Perancangan Antar Muka**

**4.4.1 Tampilan Halaman *Login***

****

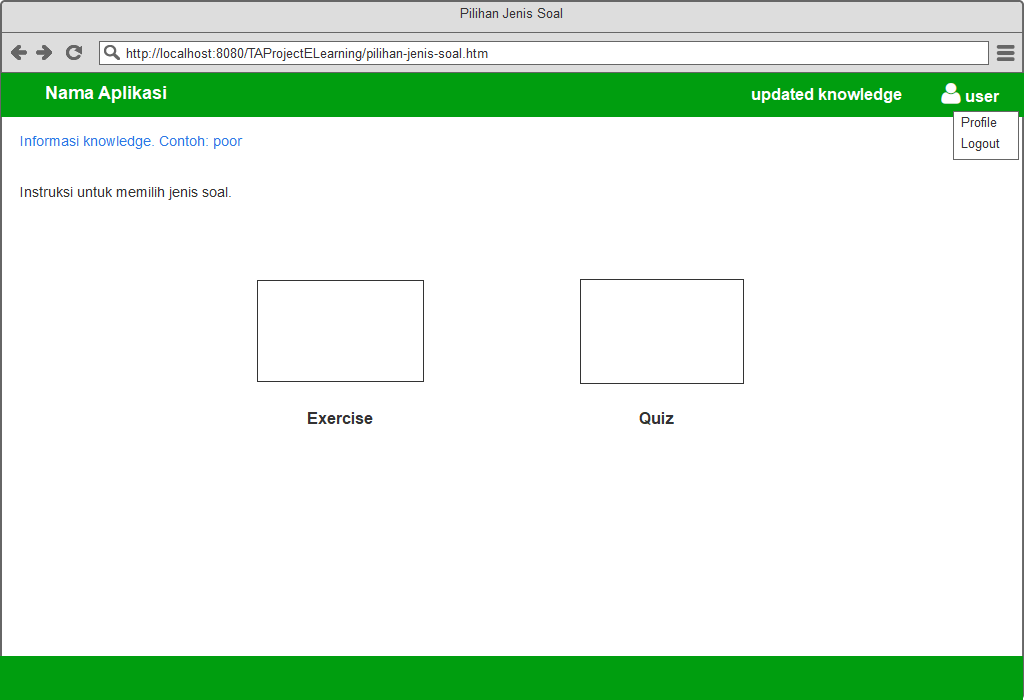
**Gambar 4.44** Tampilan Halaman *Login*

**4.4.2 Tampilan Halaman *Profile***

****

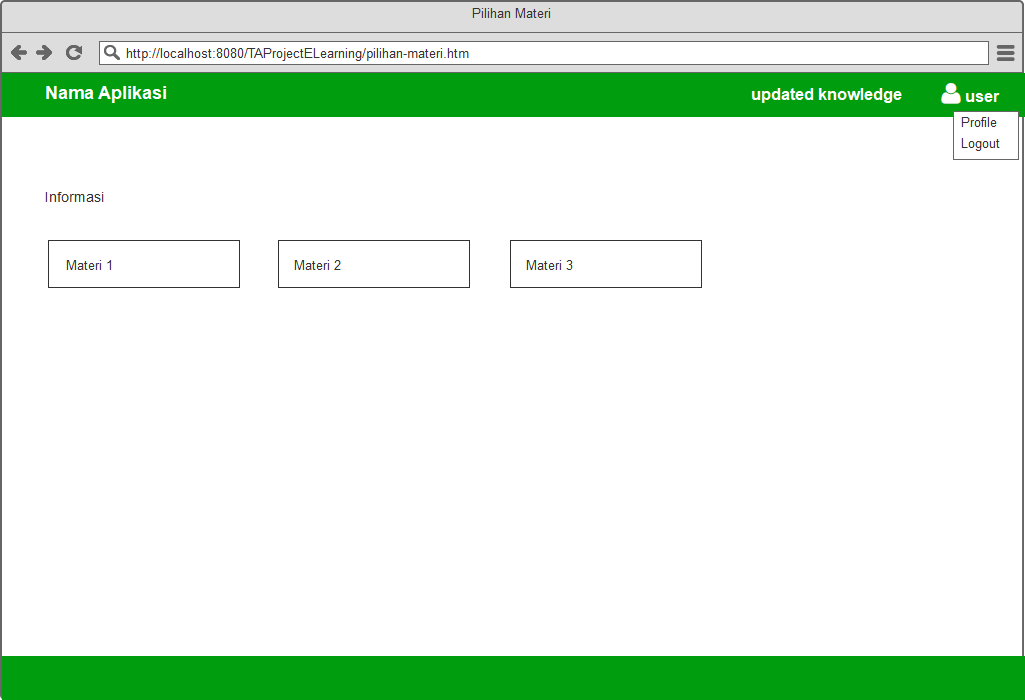
**Gambar 4.45** Tampilan halaman *Profile*

**4.4.3 Tampilan Halaman Pilih Latihan**

****

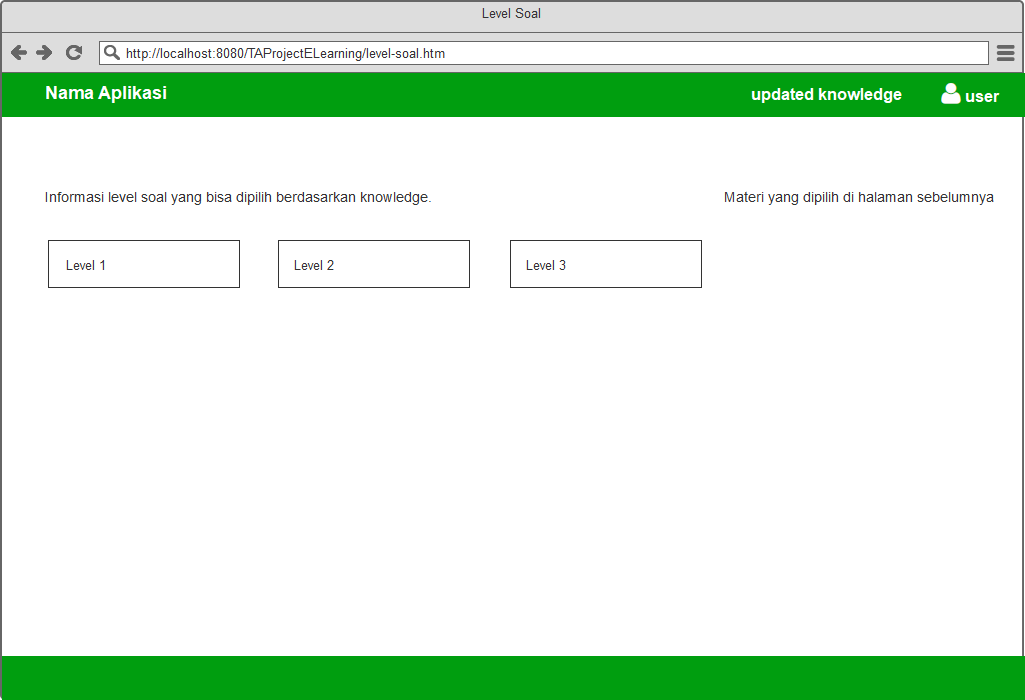
**Gambar 4.46** Tampilan Halaman Pilih *Latihan*

**4.4.4 Tampilan Halaman Latihan per materi**

****

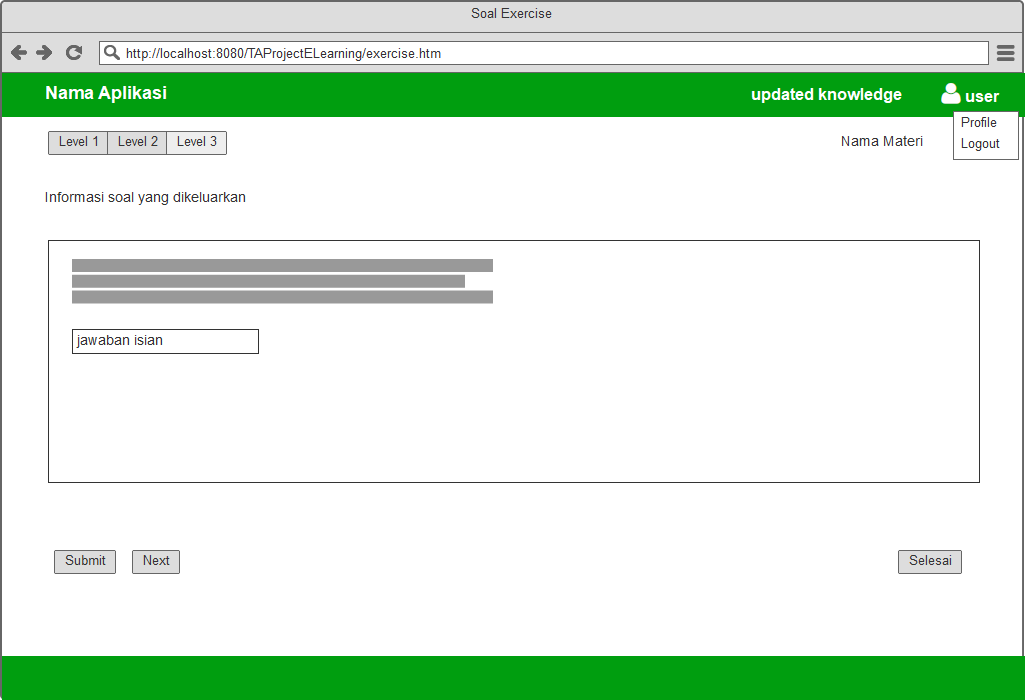
**Gambar 4.47** Tampilan halaman Latihan Per Materi

* + 1. **Tampilan Halaman Pilih Level**

****

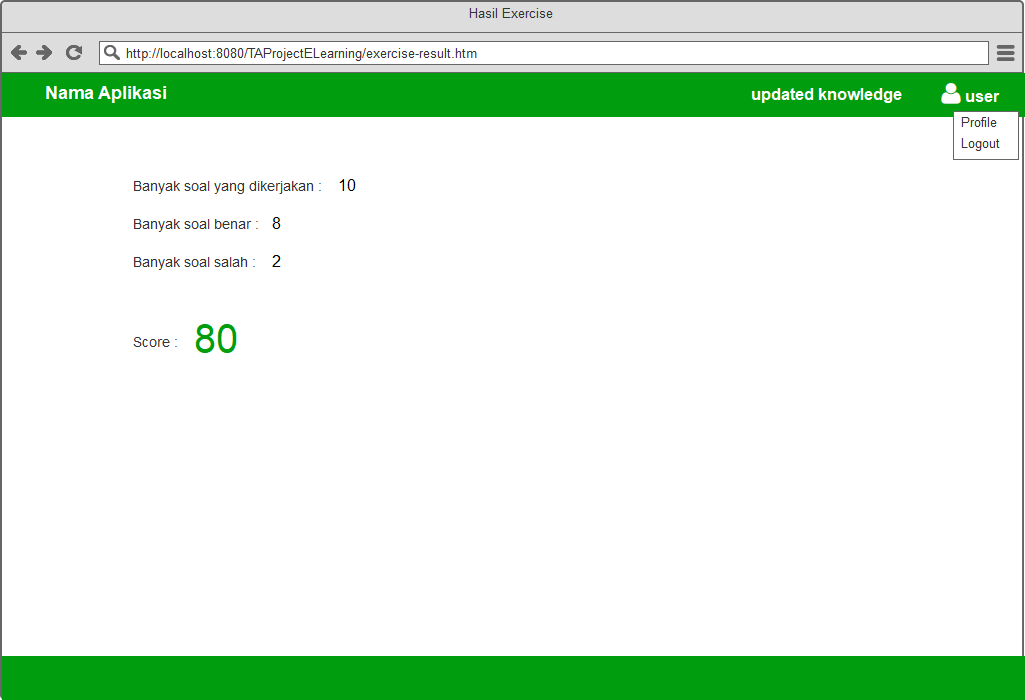
**Gambar 4.48** Tampilan halaman Pilih Level

* + 1. **Tampilan Halaman *Exercise***

****

**Gambar 4.49** Tampilan halaman *Exercise*

* + 1. **Tampilan Halaman Hasil *Exercise***

****

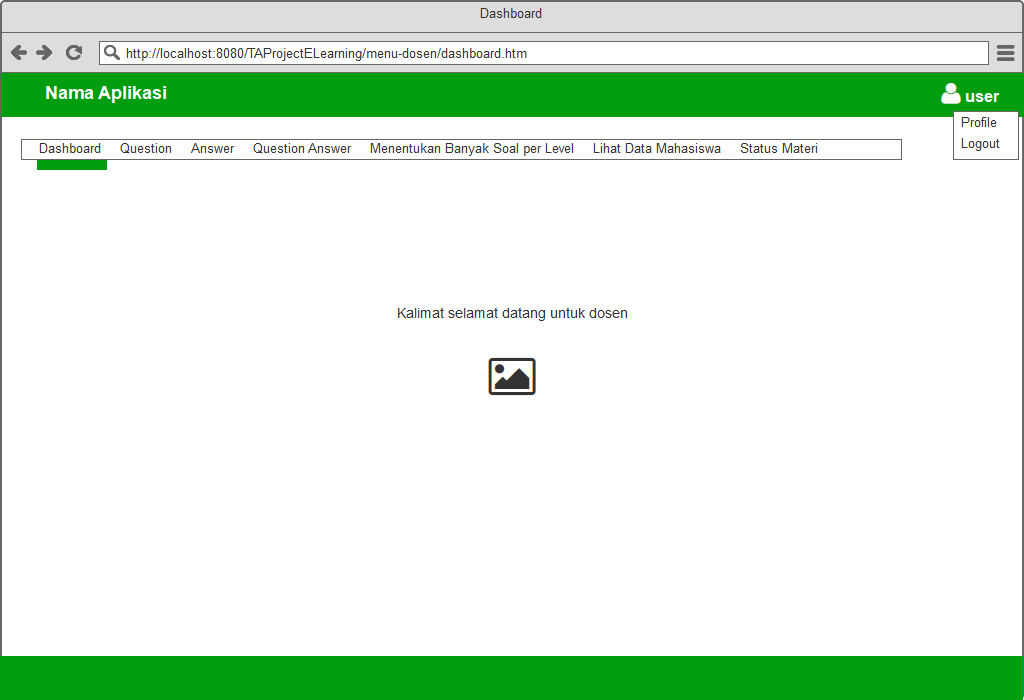
**Gambar 4.50** Tampilan halaman *Result Exercise*

* + 1. **Tampilan Halaman *Quiz***

****

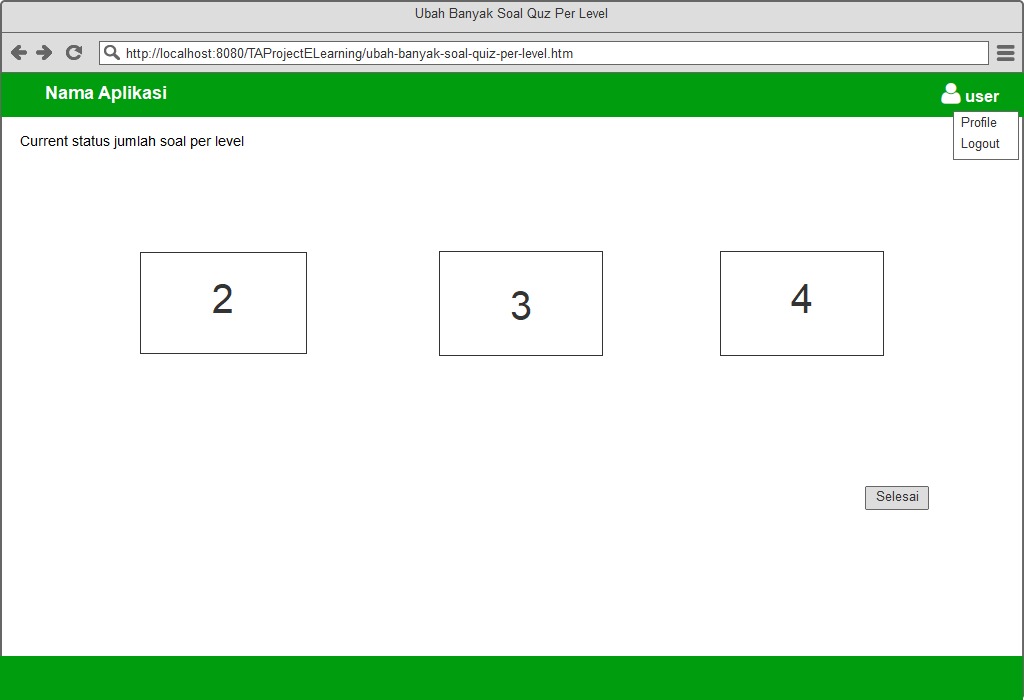
**Gambar 4.51** Tampilan halaman *Quiz*

* + 1. **Tampilan Halaman Menu Dosen**

****

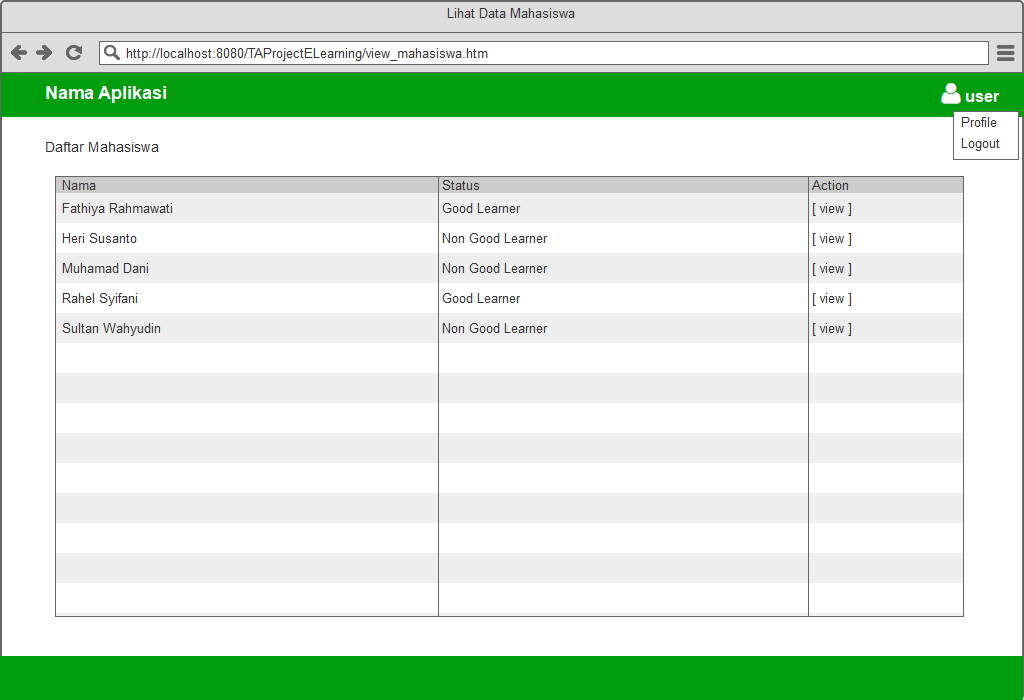
**Gambar 4. 52** Tampilan Halaman Menu Dosen

* + 1. **Tampilan Halaman Menu Dosen : Mengubah jumlah banyak soal per level**

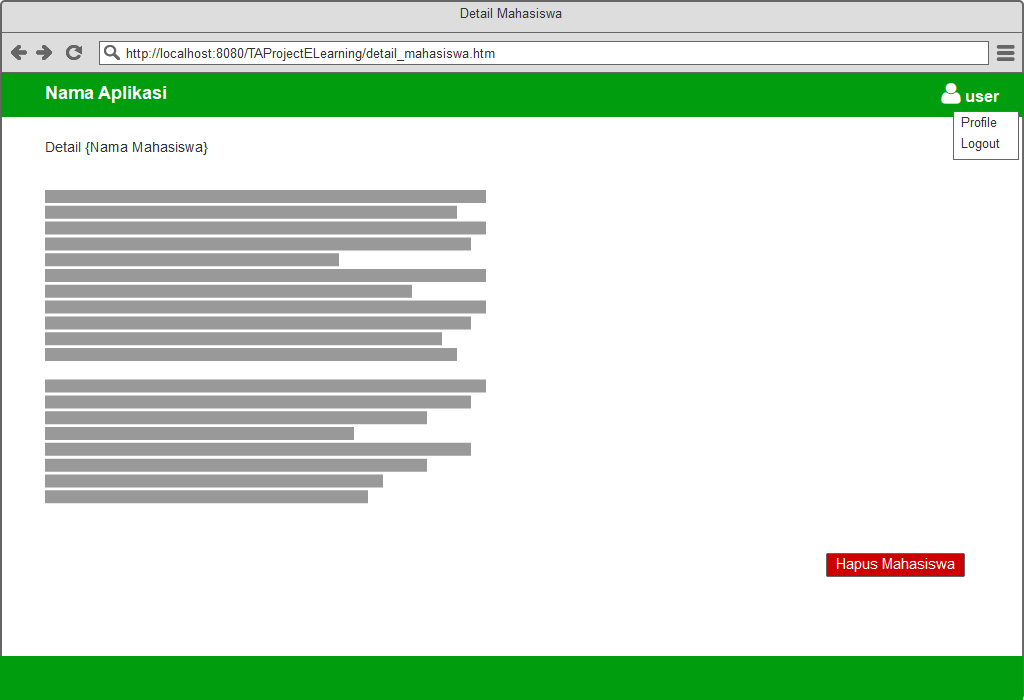
****

**Gambar 4.53** Tampilan Halaman Menu Dosen : Menentukan jumlah soal per level

* + 1. **Tampilan Halaman Menu Dosen : Melihat Data Mahasiswa**

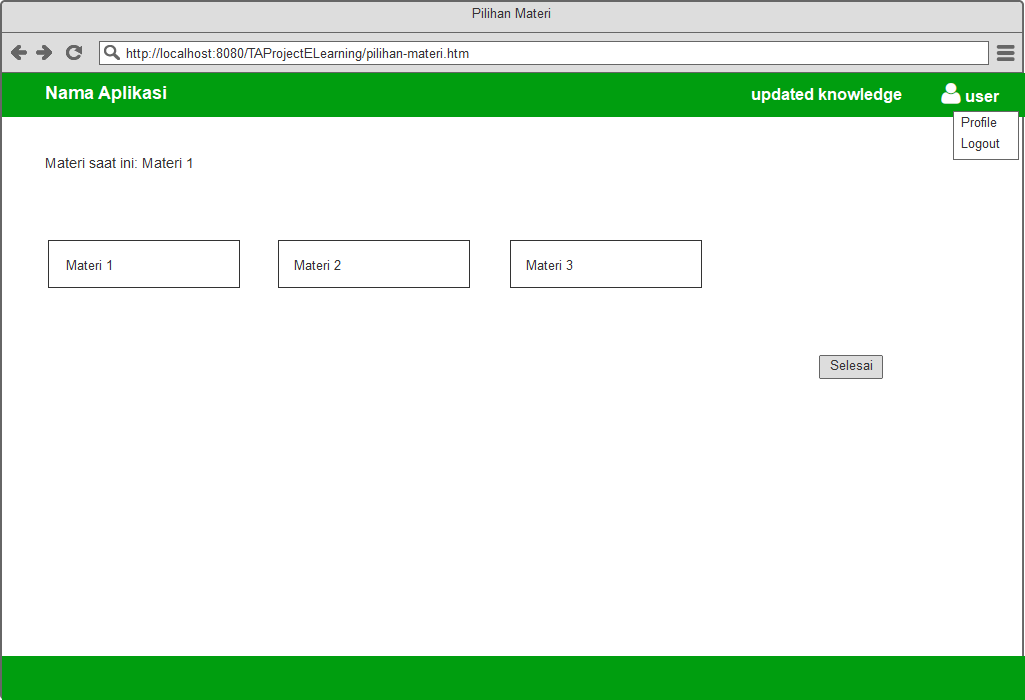
**Gambar 4. 54** Tampilan Halaman Menu Dosen : Melihat Data Mahasiswa

* + 1. **Tampilan Halaman Menu Dosen : Hapus Data Mahasiswa**



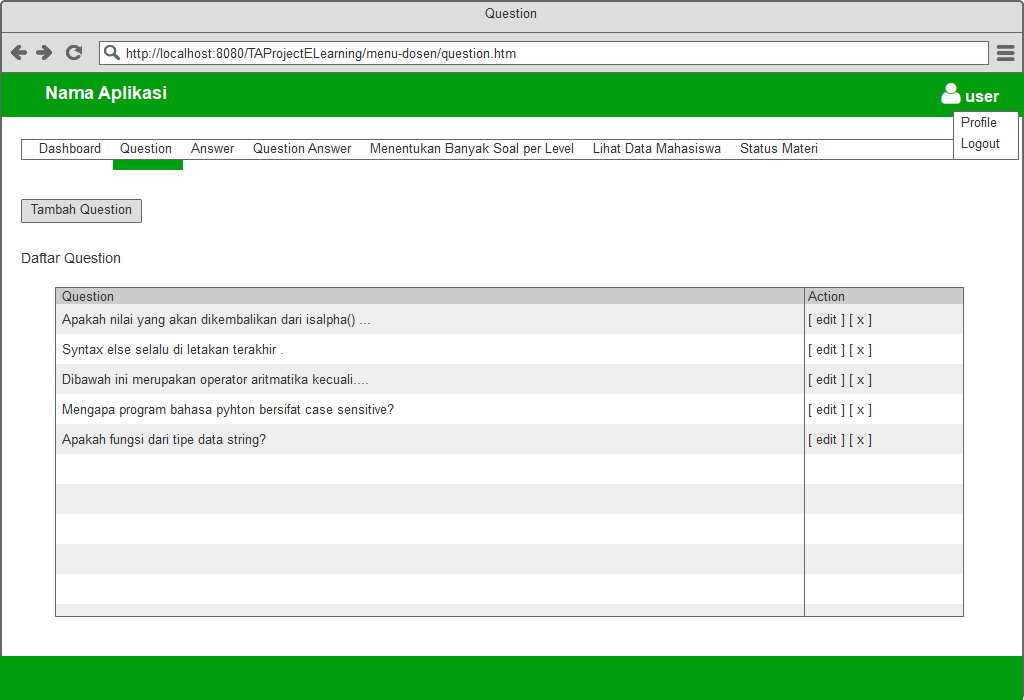
**Gambar 4. 55** Tampilan Halaman Menu Dosen : Hapus Data Mahasiswa

* + 1. **Tampilan Halaman Menu Dosen : Memilih Materi**

****

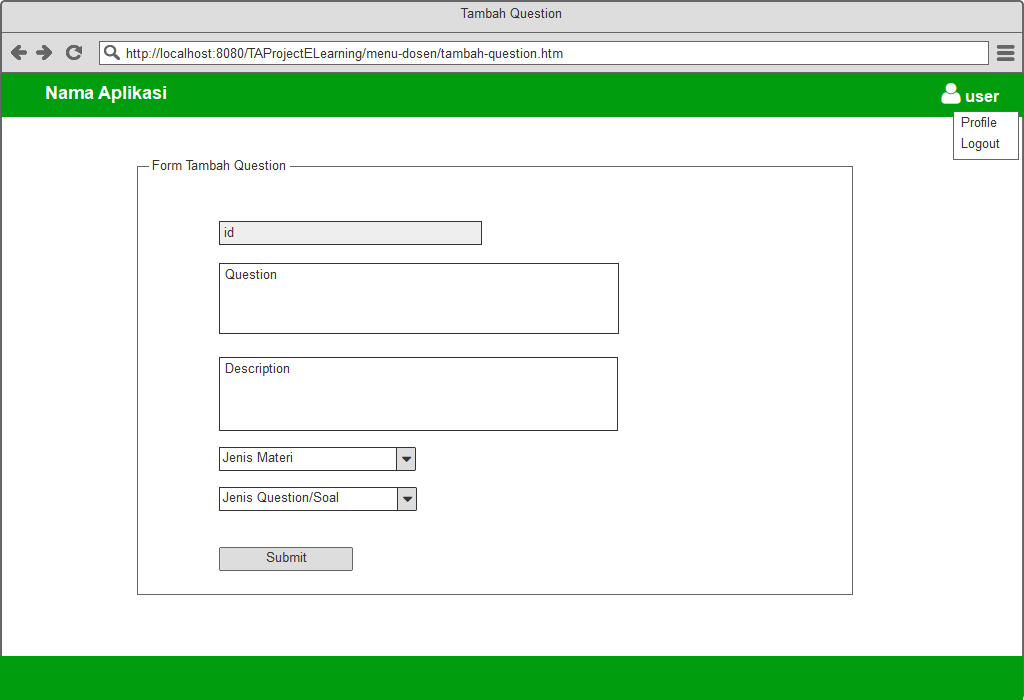
**Gambar 4. 56** Tampilan Halaman Menu Dosen :Memilih Materi

* + 1. **Tampilan Halaman Menu Dosen: Halaman *Question***

****

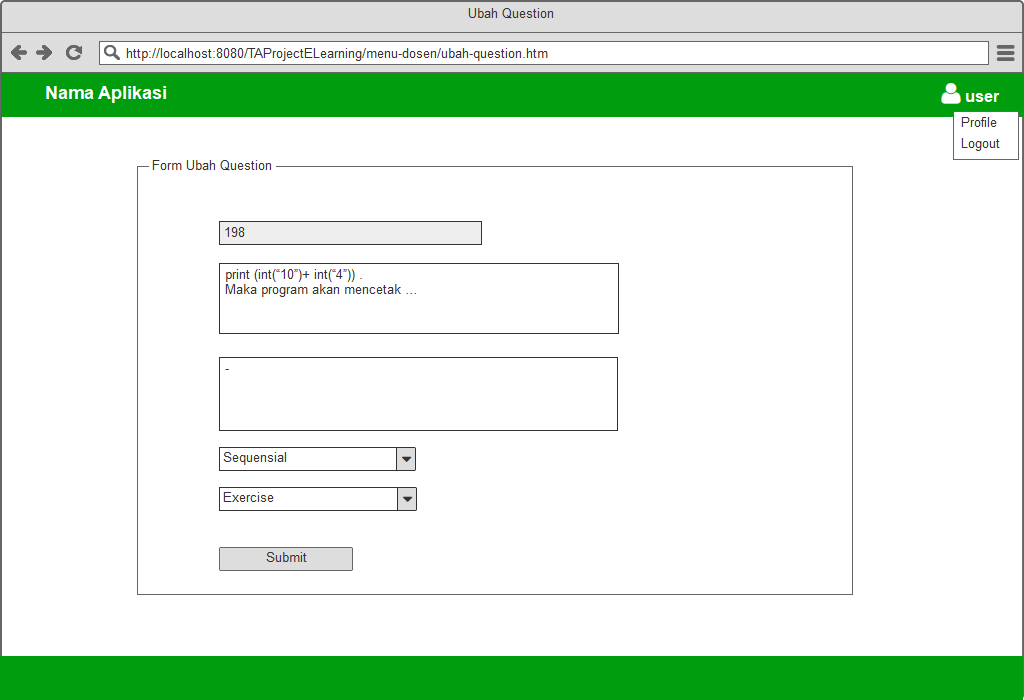
**Gambar 4. 57** Tampilan Halaman Menu Dosen :Halaman *Question*

* + 1. **Tampilan Halaman MenuDosen: Tambah *Question***

**

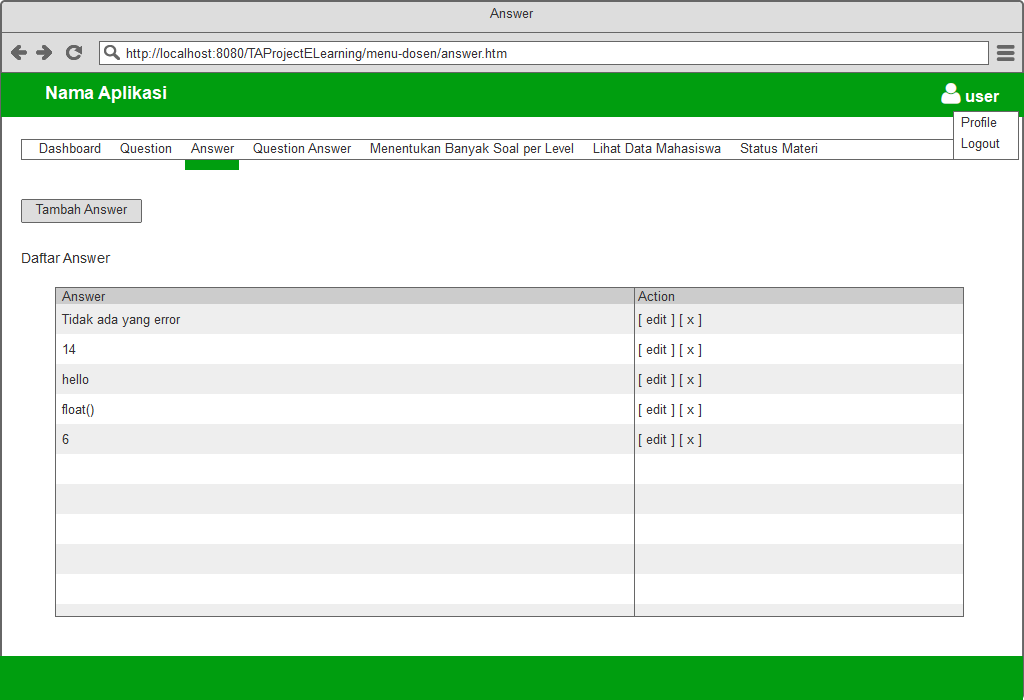
**Gambar 4. 58** Tampilan Halaman Menu Dosen : Tambah *Question*

* + 1. **Tampilan Halaman Menu Dosen: Ubah *Question***

****

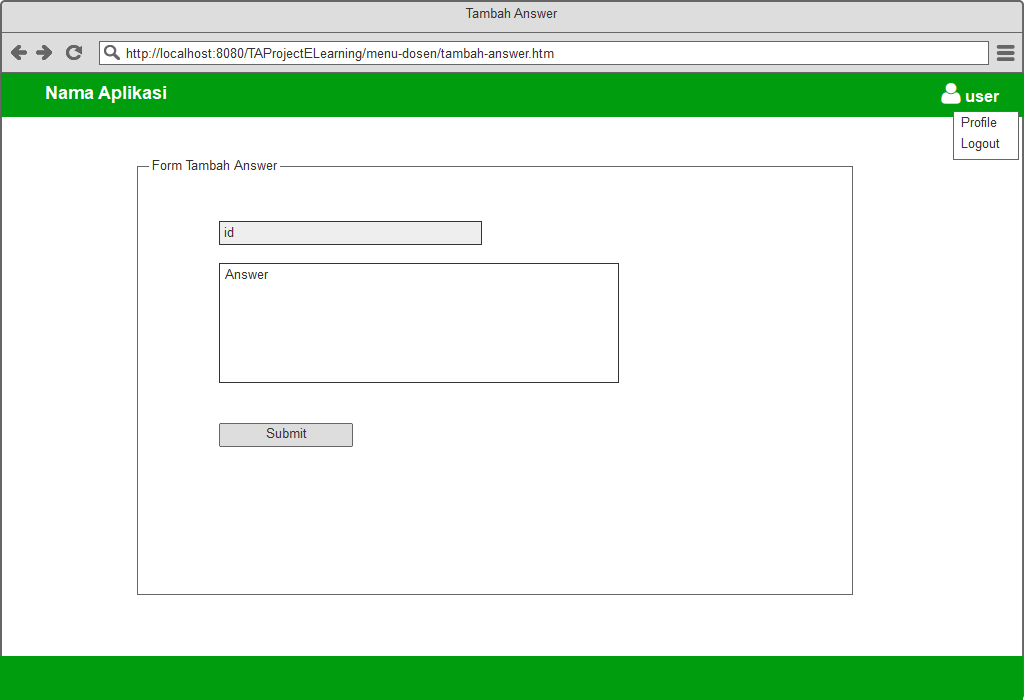
**Gambar 4. 59** Tampilan Halaman Menu Dosen : Ubah *Question*

* + 1. **Tampilan Halaman Menu Dosen: Halaman *Answer***

****

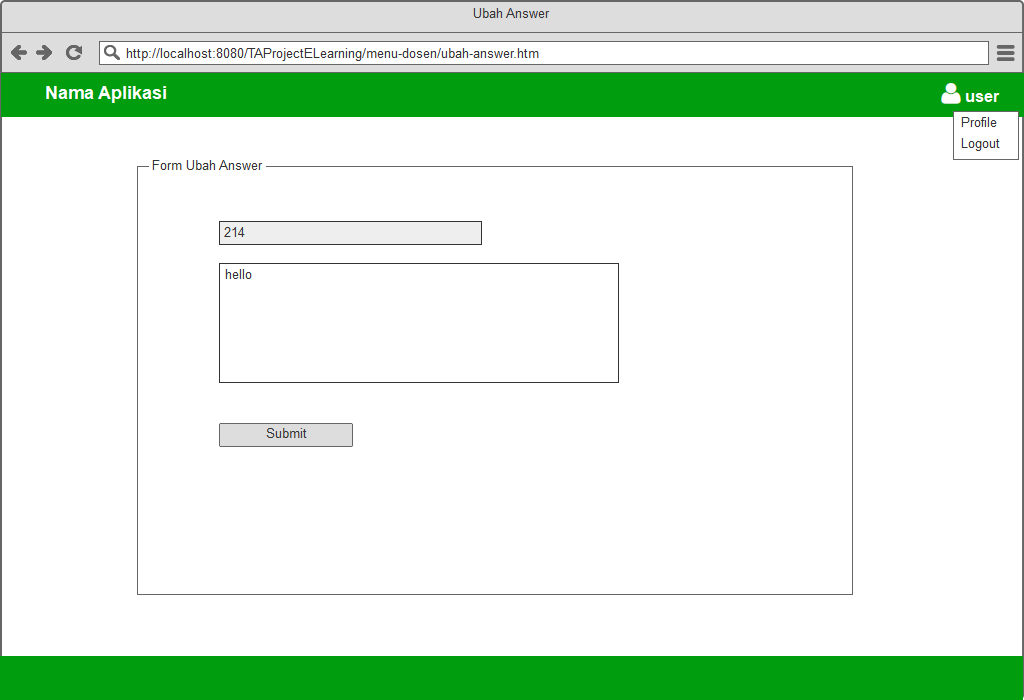
**Gambar 4.60** Tampilan Halaman Menu Dosen: Halaman *Answer*

* + 1. **Tampilan Halaman Menu Dosen: Tambah *Answer***



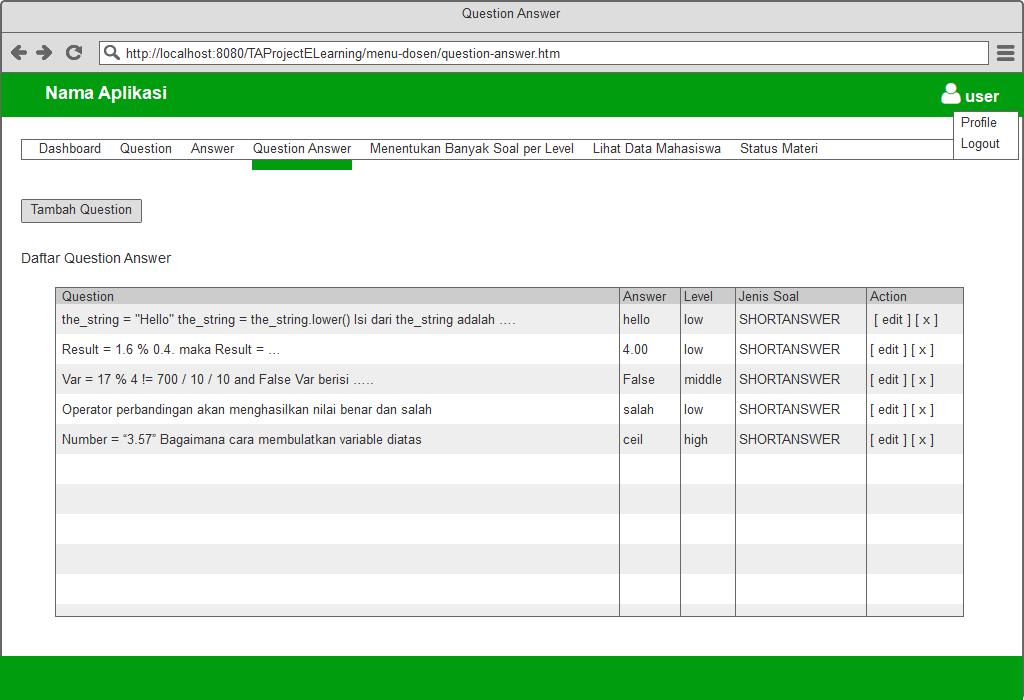
**Gambar 4.61** Tampilan Halaman Menu Dosen: Tambah *Answer*

* + 1. **Tampilan Halaman Menu Dosen: Ubah *Answer***



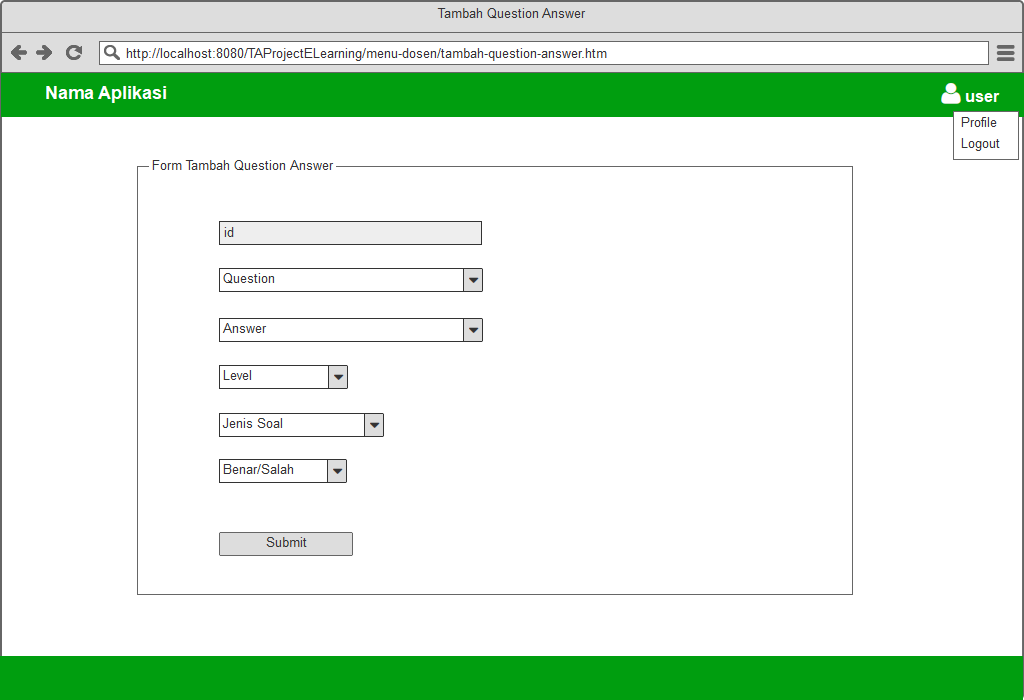
**Gambar 4.62** Tampilan Halaman Menu Dosen: Ubah *Answer*

* + 1. **Tampilan Halaman Menu Dosen: Halaman *Question Answer***



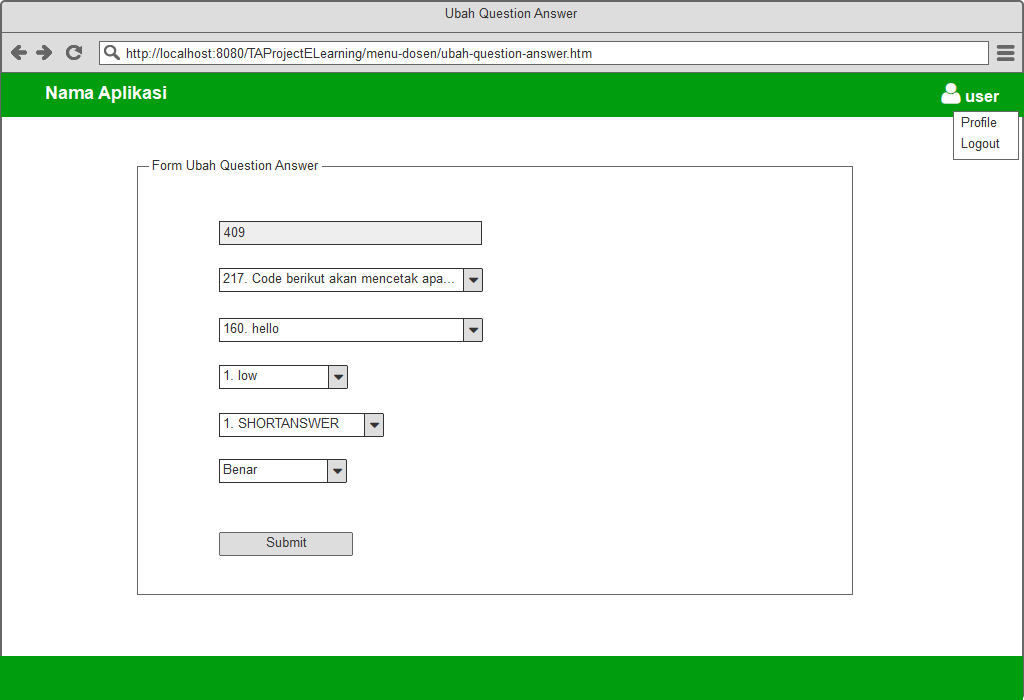
**Gambar 4.63** Tampilan Halaman Menu Dosen: Halaman *Question* *Answer*

* + 1. **Tampilan Halaman Menu Dosen: Tambah Question *Answer***



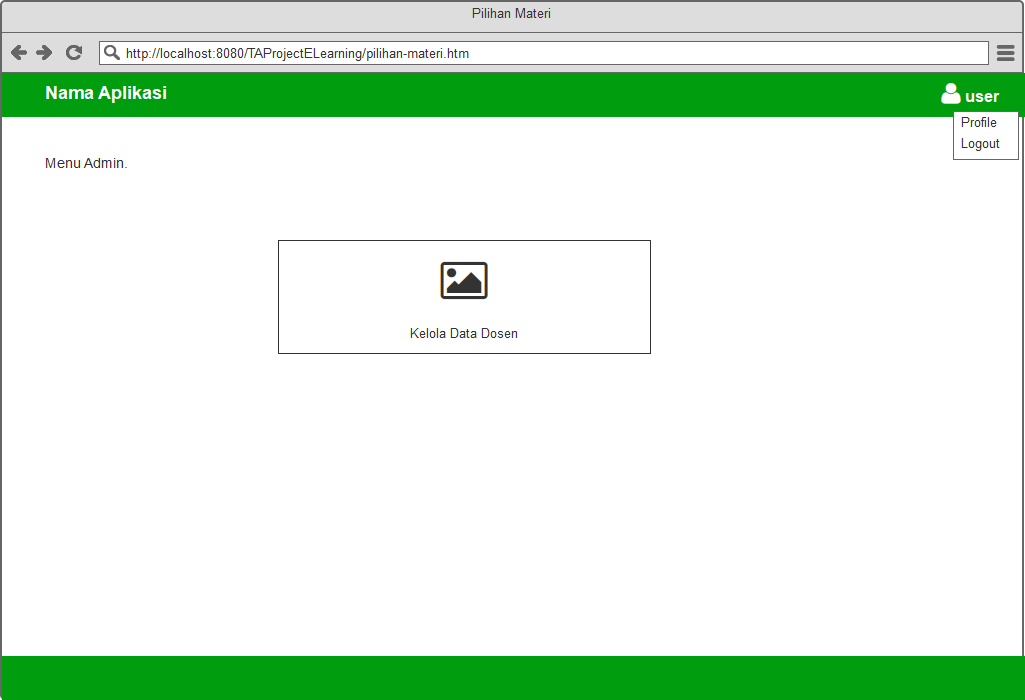
**Gambar 4.64** Tampilan Halaman Menu Dosen: Tambah *Question* *Answer*

* + 1. **Tampilan Halaman Menu Dosen: Ubah Question *Answer***



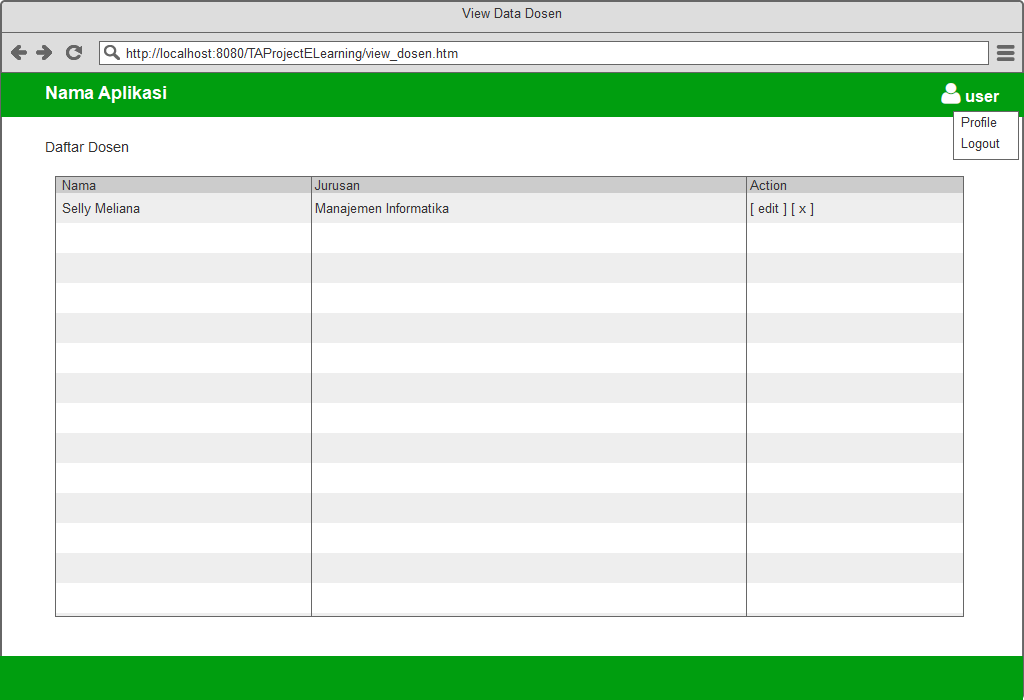
**Gambar 4.65** Tampilan Halaman Menu Dosen: Ubah *Question* *Answer*

* + 1. **Tampilan Halaman Menu *Admin***



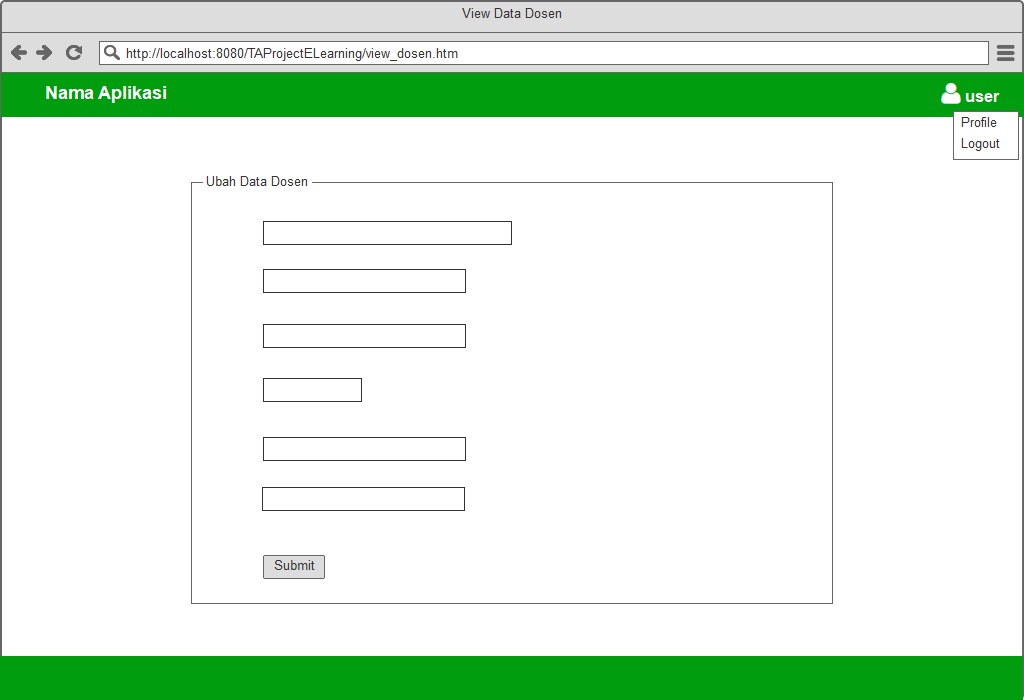
**Gambar 4.66** Tampilan Halaman Menu *Admin*

* + 1. **Tampilan Halaman Menu *Admin* : Kelola Data Dosen**



**Gambar 4.67** Tampilan Halaman Menu *Admin*: Kelola Data Dosen

* + 1. **Tampilan Halaman Menu *Admin* : Tambah Data Dosen**



**Gambar 4.68** Tampilan Halaman Menu *Admin*: Tambah Data Dosen

**BAB V**

**IMPLEMENTASI SISTEM**

Pada bab ini akan diimplementasikan hasil dari perancangan yang telah penulis buat pada bab sebelumnya. Bab implementasi ini meliputi lingkungan implementasi, basis data, implementasi proses dan implementasi antarmuka (*interface*).

1. **Lingkungan Implementasi**

Lingkungan implementasi meliputi lingkungan perangkat keras (*hardware*) dan lingkungan perangkat lunak (*software*).

1. **Lingkungan Perangkat Keras (*Hardware*)**

|  |  |
| --- | --- |
| **Perangkat Keras (Hardware)** | **Keterangan** |
| *Processor* | *Client :*   * *Processor* Intel *Core* i3 / AMD A4.   *Server* :   * *Processor* Intel *Core* i7 2600 |
| RAM  132 | *Client*:   * Minimum : 1 GB * *Recommended* : 2 GB   *Server* :   * Minimum : 2 GB * *Recommended* : 8 GB |
| Hardisk | *Client :*   * Minimal *250GB*   *Server :*   * *Recommended 1TG* |
| Display | Minimum : 1024 x 768 256 Color  *Recommended* : 1366 x 768 High Color |
| Monitor | *Standard* |
| *Mouse* | *Standard* |
| *Keyboard* | *Standard* |

**Tabel 5.1** Lingkungan Perangkat Keras *(hardware)*

1. **Lingkungan Perangkat Lunak (*Software*)**
   * + 1. **Server**

|  |  |
| --- | --- |
| **Perangkat Lunak** | **Keterangan** |
| Mozila Firefox | Sebagai *web* *browser* untuk membuka aplikasi *e-learning*. |
| Netbeans | Untuk mengetikkan kode - kode yang dibutuhkan untuk membuat aplikasi |
| Xampp | *server database*  lokal yang penulis gunakan |

**Tabel 5.2** Lingkungan Perangkat Lunak untuk *Server* *(software)*

* + - 1. ***Client***

|  |  |
| --- | --- |
| **Perangkat Lunak** | **Keterangan** |
| Mozila Firefox / Chrome | Sebagai *web* *browser* untuk membuka aplikasi *e-learning*. |

**Tabel 5.3** Lingkungan Perangkat Lunak untuk *Client* *(software)*

1. **Hasil Implementasi**
2. **Implementasi Basis Data**

Basis data yang digunakan untuk mengimplementasikan Aplikasi *E-learning* ini adalah MySQL. Berikut adalah hasil implementasi tabel yang terdapat dalam aplikasi :

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Tabel** | **Hasil Implementasi** | **Keterangan** |
| 1. | Tb\_*user* | *User* | Mengelola data pengguna Sistem *E-Learning* |
| 2. | Tb\_*Category* | Kategori | Mengelola data Kategory Soal *Exercise* atau *quiz* |
| 3. | Tb\_*Quiz* | Soal | Mengelola data Soal-soal *Quiz* dan *Exercise* |
| 4. | Tb\_*Answer* | Jawaban | Mengelola data Jawaban Soal *Quiz* dan *Exercise* |
| 5. | Tb\_jenissoal | Jenis Soal | Mengelola data jenis jawaban dari soal yaitu Pilihan Ganda atau Jawaban Pendek |
| 6. | Tb\_Level | Level | Mengelola data Level Soal |
| 7. | Tb\_*role* | *Role* | Mengelola data peran pengguna aplikasi |
| 8. | Tb\_*matery* | Materi | Mengelola data Materi Soal |
| 9. | Tb\_*Knowledge* | *Knowledge* | Mengelola data *Knowledge* Mahasiswa |
| 10. | Tb\_QA | *QuestionAnswer* | Mengelola data Soal dan Jawaban |
| 11. | Tb\_*Mayors* | *Mayor* | Mengelola data Jurusan Mahasiswa |
| 12. | Tb\_Model | Model | Mengelola data Mahasiswa yang menjadi *Good learner* |
| 13. | Tb\_HistoriQuiz | HistoriQuiz | Mengelola data *quiz* yang pernah dikerjakan oleh mahasiswa |
| 14 | Tb\_*ResultQuiz* | *ResultQuiz* | Mengelola Hasil *quiz* |
| 15 | Tb\_*ResultExercise* | *ResultExercise* | Mengelola Hasil *Exercise* |
| 16 | Tb\_*Resultquiz\_after* | *Resultquiz\_after* | Mengelola setelah hasil *result quiz* |
| 17 | Tb\_*Resultquiz\_before* | *Resultquiz\_before* | Mengelola sebelum hasil *result quiz* |
| 18. | Tb\_StatusMateri | StatusMateri | Mengelola Status Materi |
| 19 | Tb\_*session* | *Session* | Mengelola historis *user* yang *login* |

**Tabel 5.4** Implementasi Basis Data

**5.2.2 Implementasi Proses**

Sub bab ini menjelaskan *page – page* yang menjadi antar muka (*interface*) yang telah dibuat untuk masing – masing proses yang ada dalam aplikasi. Berikut adalah daftar tombol yang umum digunakan yang terdapat dalam *page* pada aplikasi yang sudah dibuat.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Keterangan** | |
| 1. |  | Tombol untuk menuju ke halaman *exercise* | |
| 2. |  | Tombol untuk menuju ke halaman *quiz* | |
| 3. |  | Tombol untuk memilih materi dari soal yang akan di kerjakan | |
| 4. |  | Tombol untuk memilih level dari soal yang akan di kerjakan | |
| 5. |  | Tombol untuk submit jawaban dari soal *quiz* atau *exercise*. | |
| 6. |  | Tombol untuk melanjutkan ke soal *exercise* berikutnya tanpa menyimpan jawaban. | |
| 7. |  | Tombol yang di tekan ketika selesai mengerjakan soal *exercise*. | |
| 8. |  | | Tombol untuk menampilkan halaman awal. |
| 9. |  | | Tombol profile untuk melihat profil dan logout untuk keluar dari halaman dan kembali ke halaman *login*. |
| 10. |  | | Tombol untuk menampilkan halaman dosen untuk menentukan jumlah soal *quiz* per level |
| 11. |  | | Tombol untuk menampilkan data mahasiswa. |
| 12. |  | | Tombol untuk menampilkan halaman status materi. |
| 13. |  | | Tombol untuk melihat detail mahasiswa , berikut nilai nya. |
| 14. |  | | Tombol untuk memilih jumlah soal per level. |
| 15 |  | | Tombol untuk memilih materi yang di pelajari. |
| 16 |  | | Tombol untuk menampilkan halaman *dashboard*. |
| 17 |  | | Tombol untuk menampilkan halaman *Question*. |
| 18 |  | | Tombol untuk menampilkan halaman *Answers*. |
| 19 |  | | Tombol untuk menampilkan halaman *Question Answers*. |
| 20 |  | | Tombol untuk menampilkan halaman *User*. |
| 21 |  | | Tombol yang berwarna hijau berfungsi untuk mengubah data dan yang merah untuk menghapus data. |
| 22 | C:\Users\Dyah_Nuraeni\Downloads\SS\ss\13.PNG | | Tombol untuk menghapus data Mahasiswa |

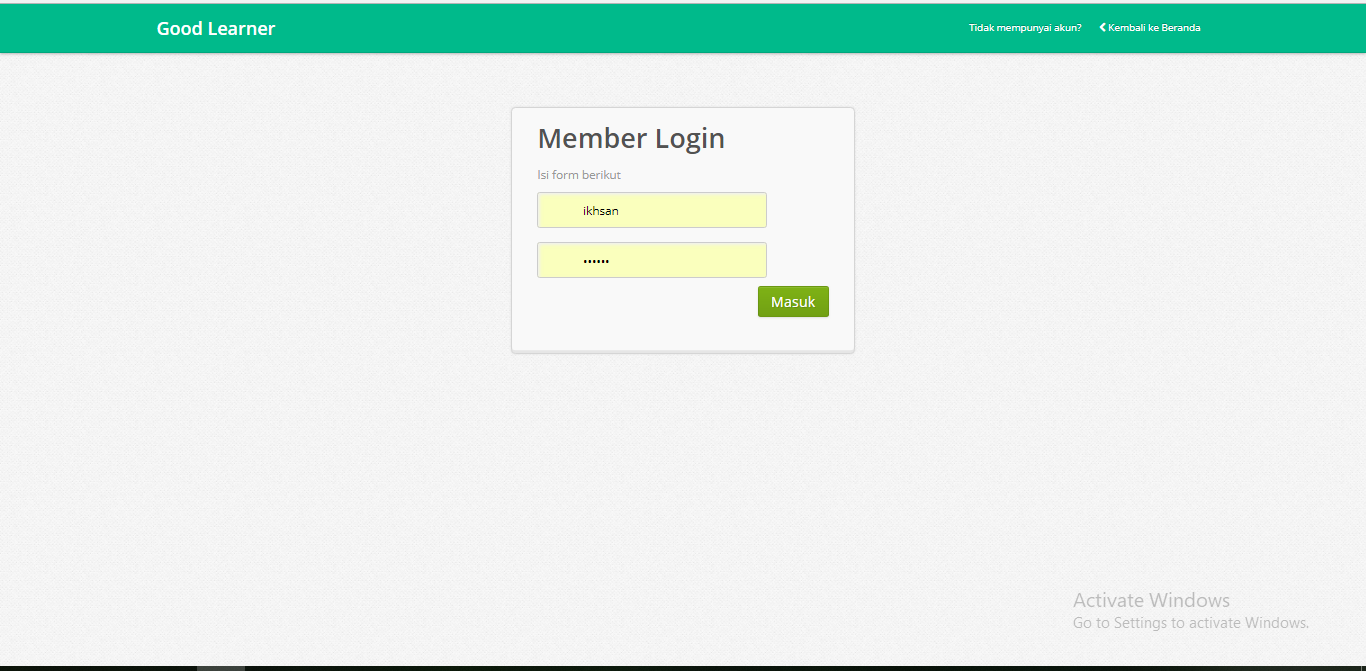
**Tabel 5.5** Daftar Tombol yang Digunakan Pada Aplikasi

1. **Implementasi Antarmuka (*Interface*)**

Sub bab ini bertujuan untuk memberikan informasi mengenai tata cara penggunaan aplikasi *E-Learning*, yang akan dilakukan oleh Mahasiswa, Dosen, maupun *Admin*.

* + 1. **Proses *Login***

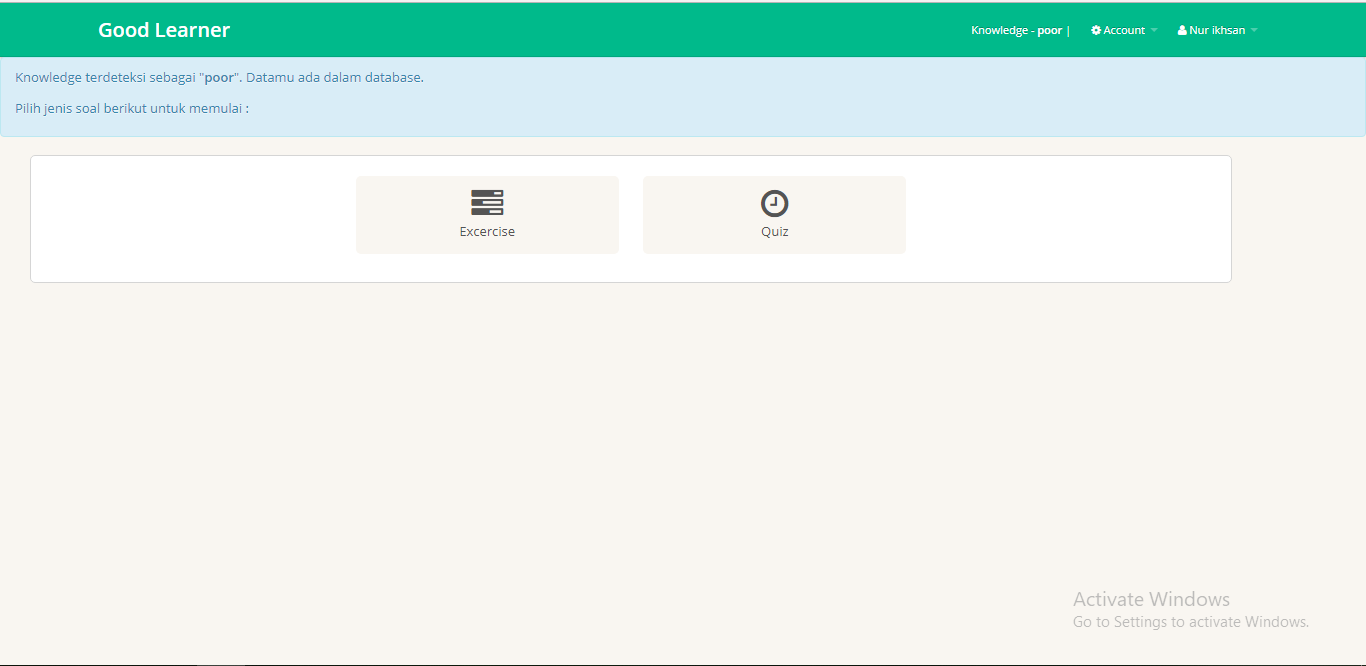
Proses *login* aplikasi dimulai dengan cara mengetikkan alamat *web* aplikasi *E-Learning* pada kolom *url*. Kemudian akan muncul tampilan halaman *login* sebagai berikut :



**Gambar 5.1** Halaman *Login*

Setelah muncul halaman *login*, Pengguna menginputkan *username* dan *password*. Jika *username* dan *password* salah maka Pengguna diminta untuk menginputkan *username* dan *password* lagi.

Jika *username* dan *password* sudah benar (valid), maka akan tampil akan Halaman Utama. Adapun tampilannya sebagai berikut



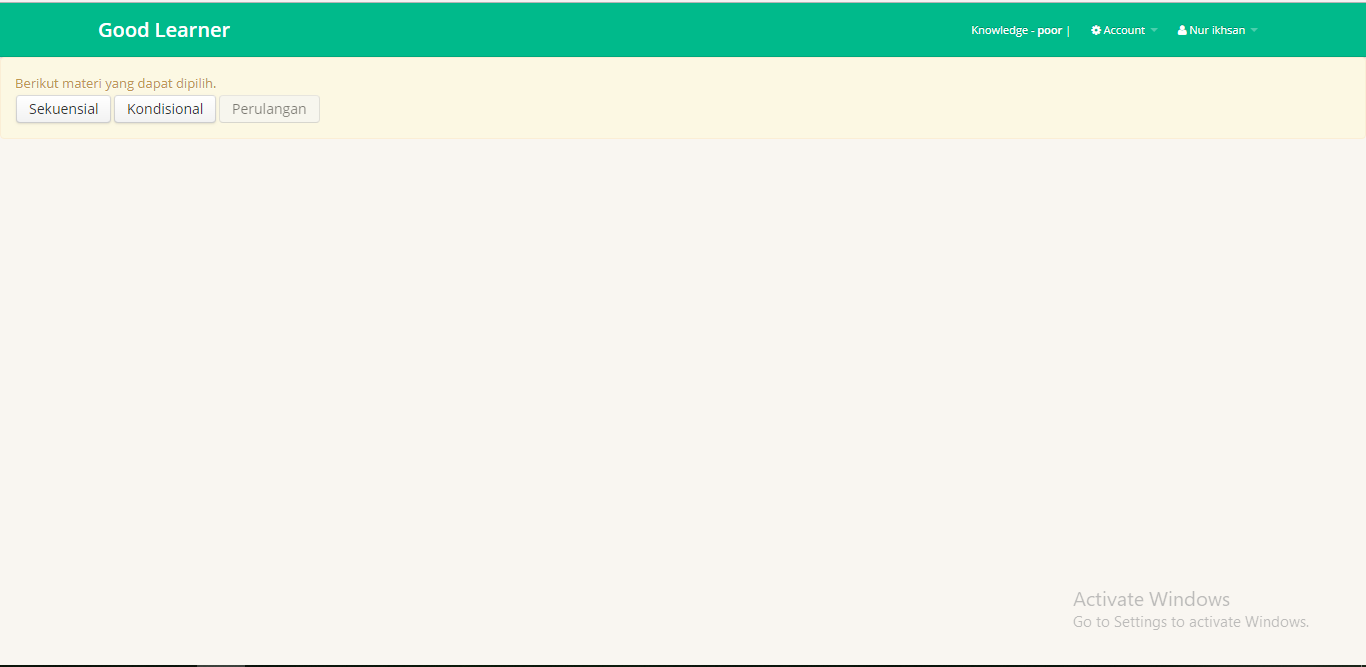
**Gambar 5.2** Halaman Utama

* + 1. **Implementasi Antarmuka (*Interface*) untuk Mahasiswa**

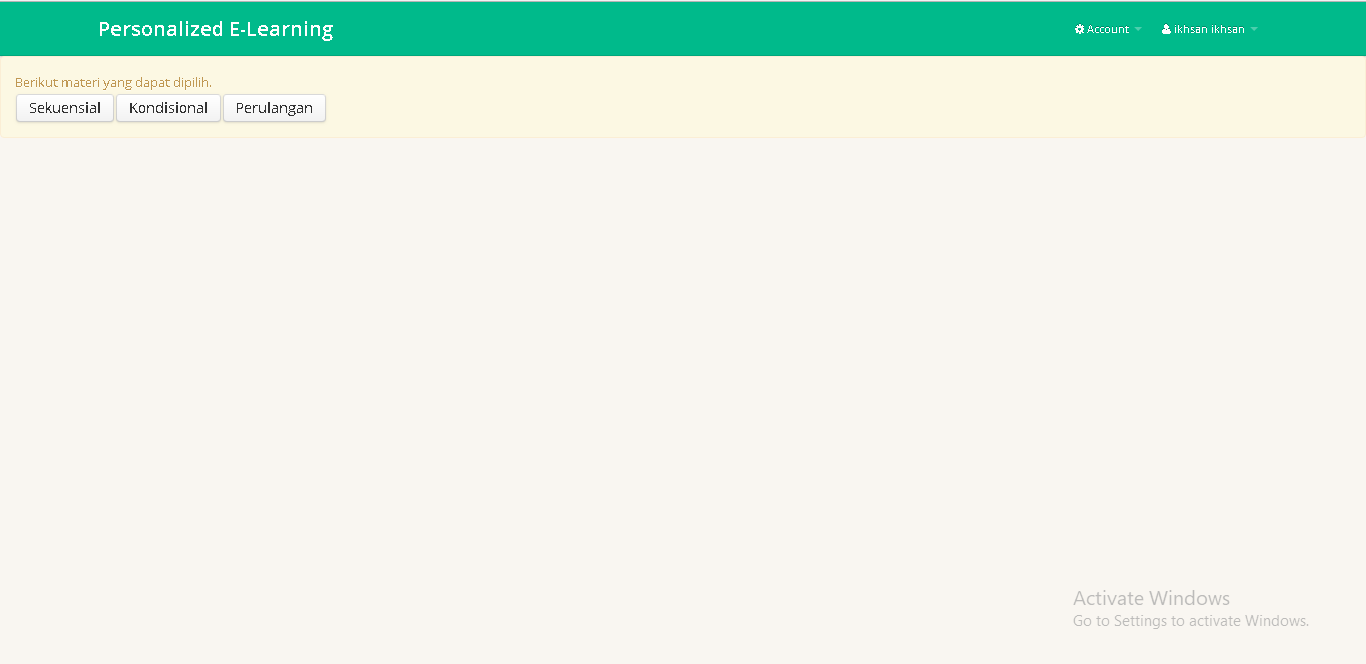
Berikut ini adalah petunjuk penggunaan aplikasi *e-learning* yang digunakan oleh Mahasiswa.

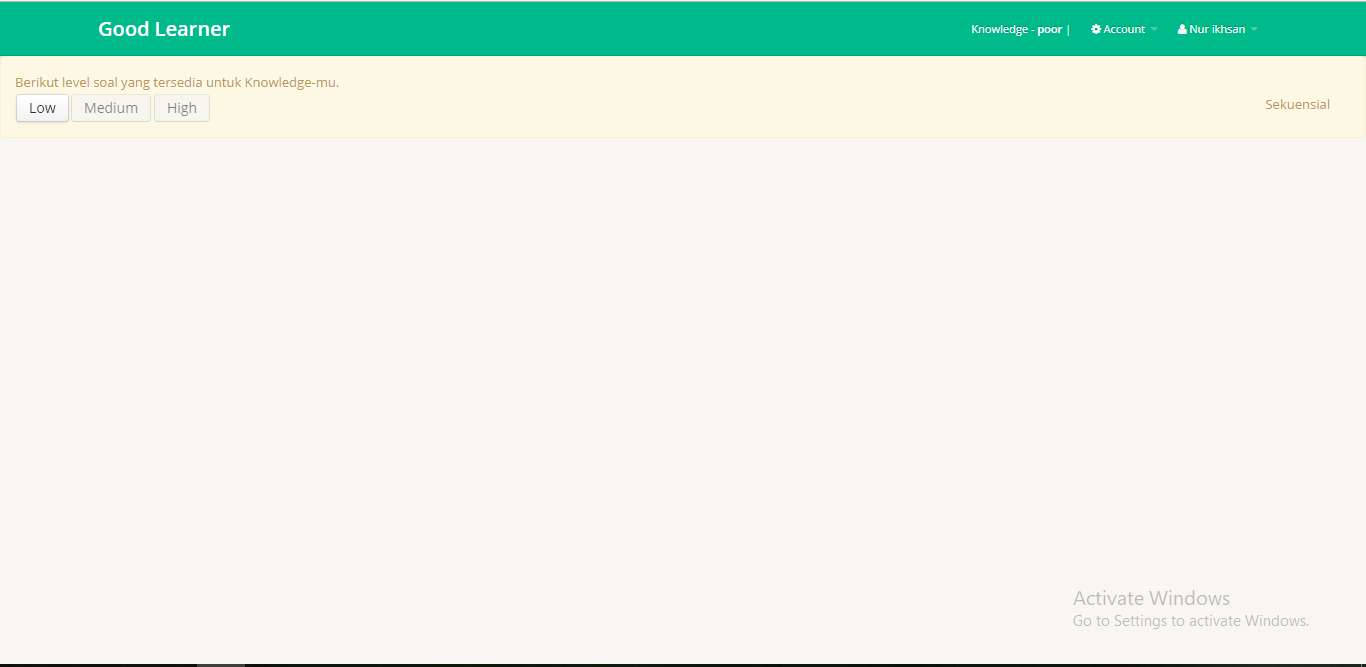
**5.3.2.1 Halaman Utama** 🡪 ***Exercise***

Halaman ini berfungsi untuk mengerjakan *Exercise* untuk Mahasiswa. Mahasiswa akan memilih Materi yang akan menjadi *Exercise*. Kemudian akan muncul Pilihan Level Soal yang akan dikerjakan. Level terdiri dari *low*, *middle* dan *High* . Apabila mahasiswa sudah mendapatkan nilai yang meningkat maka level yang lebih tinggi dari level sebelumnya akan dapat diakses. Setelah itu Mahasiswa akan mengerjakan *Exercise* kemudian jika telah selesai maka akan muncul *Form* Nilai yang menampilkan *Score* yang didapat mahasiswa. Halaman ini dapat diakses dengan cara memilih menu **Halaman Utama 🡪 *Exercise***. Adapun tampilannya sebagai berikut:

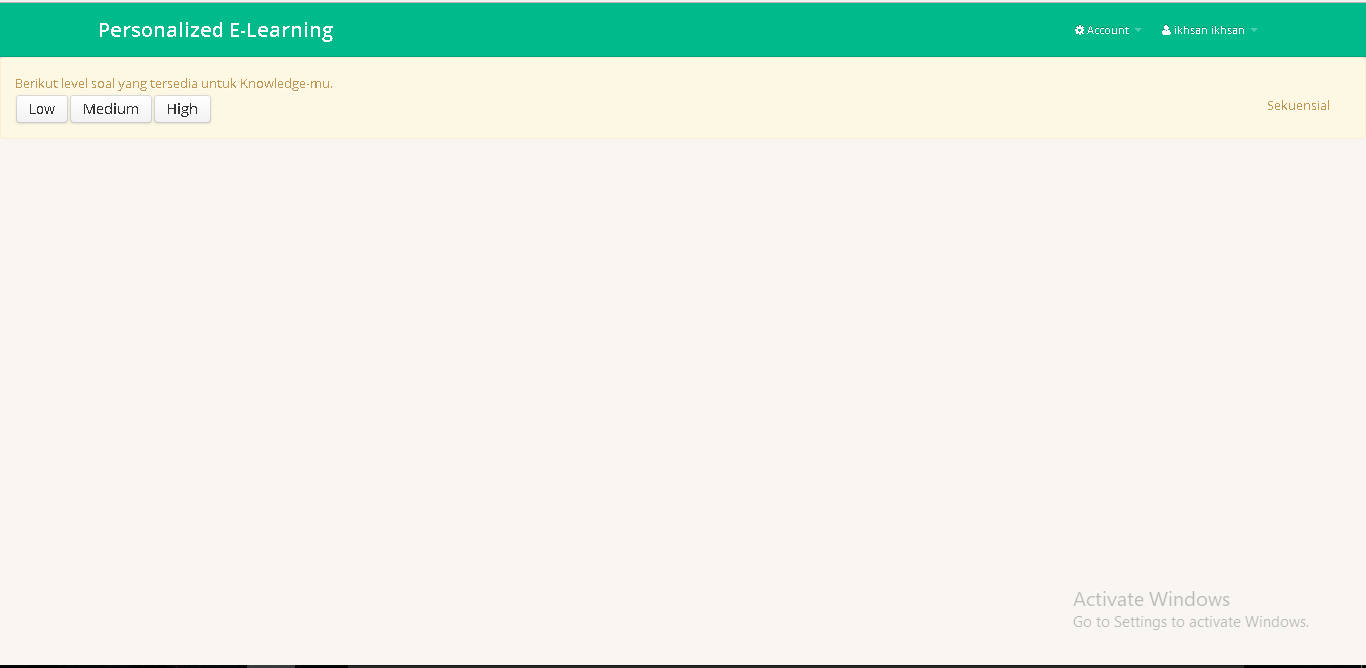


**Gambar 5.3** Halaman Materi Soal

Mahasiswa memilih Materi Soal dengan mengklik tombol berikut . Selanjutnya akan muncul tampilan seperti berikut .



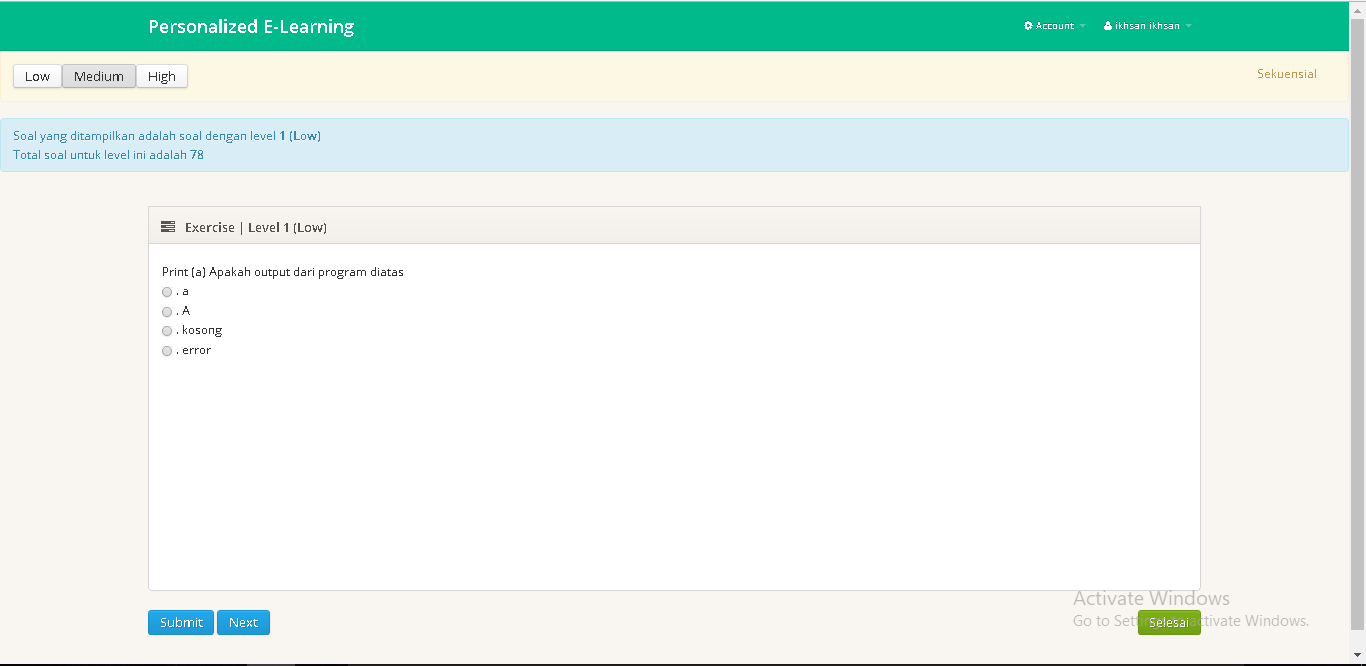
**Gambar 5.4** Halaman *Level* Soal

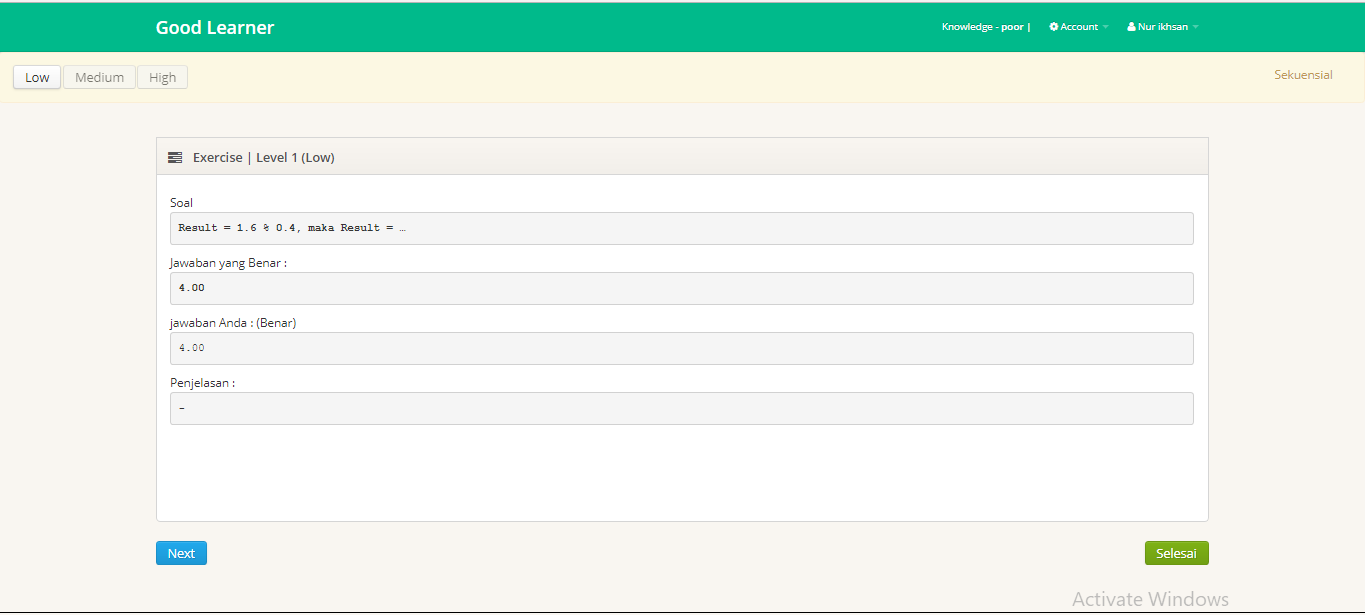
Selanjut mahasiswa memilih *Level*  Soal dengan mengklik tombol berikut . Maka soal yang muncul akan sesuai dengan level yang dipilih. Kemudian muncul Tampilan Soal



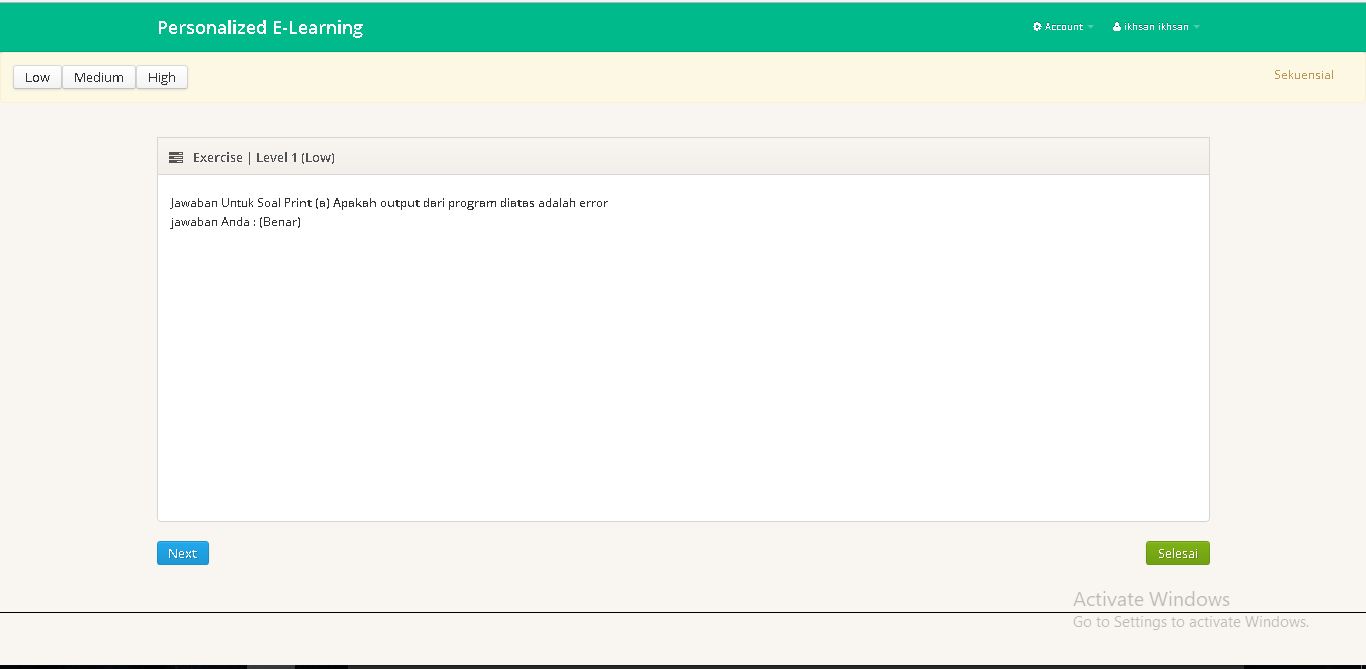
**Gambar 5.5** Halaman Soal

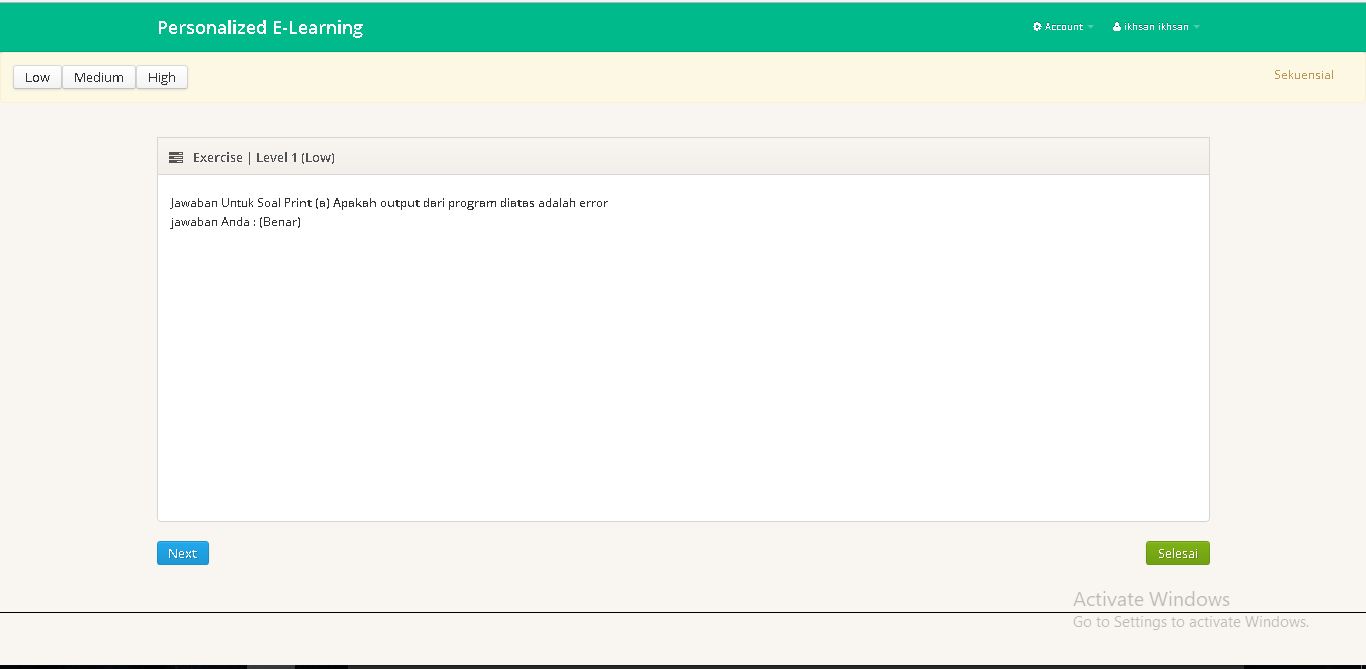
Ditampilan ini, mahasiswa akan menjawab soal-soal yang muncul, dan jika mahasiswa telah menjawab soal maka mahasiswa klik tombol

 , Maka muncul *Form* berikut :



**Gambar 5.6** Halaman JawabanSoal

Namun mahasiwa juga bisa melanjutkan tanpa menjawab soal dengan mengklik tombol , maka akan muncul soal selanjutnya.

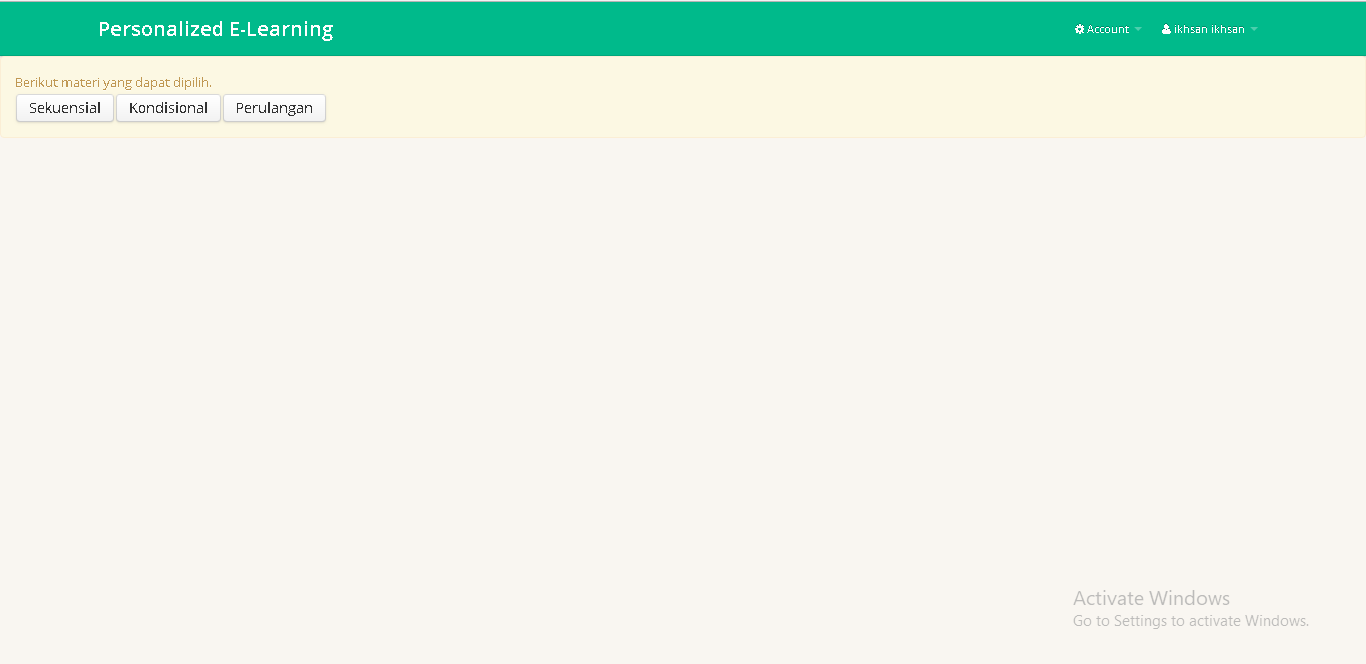
Mahasiswa dapat keluar dari *Form Quiz* dengan mengklik tombol . Maka tampilan yang akan muncul selanjutnya adalah menampilkan yang di dapat mahasiwa



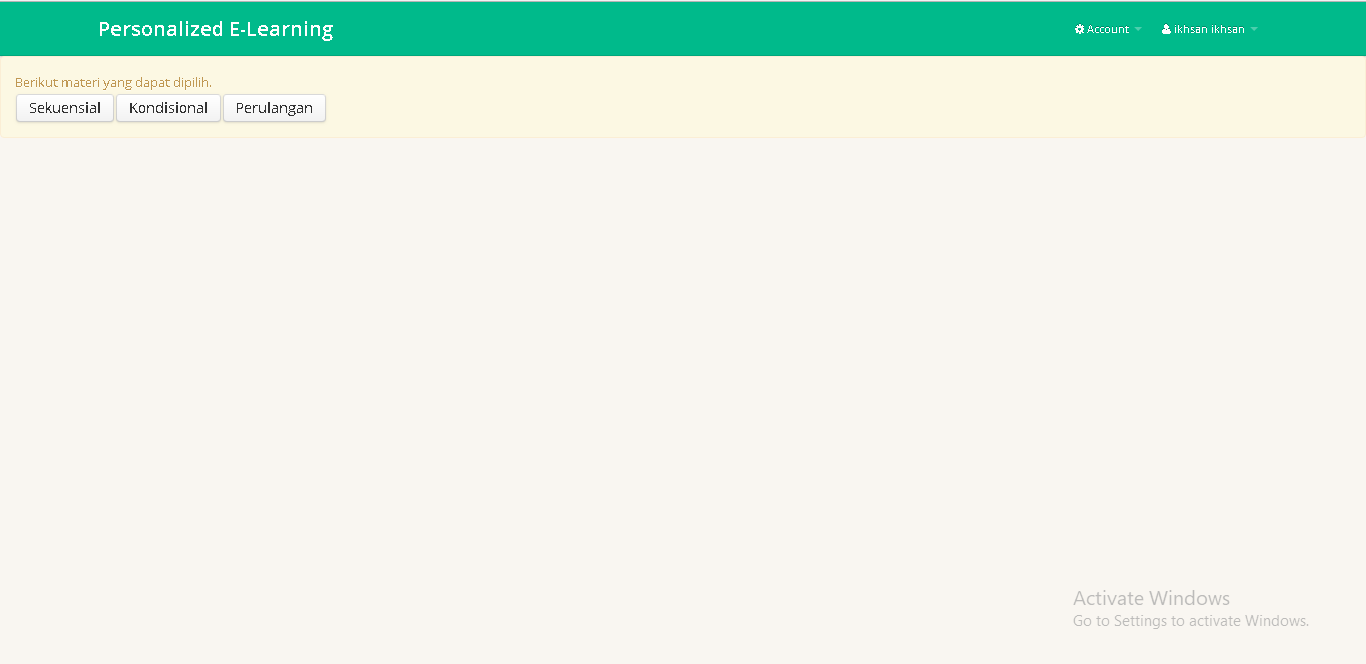
**Gambar 5.7** Halaman Hasil

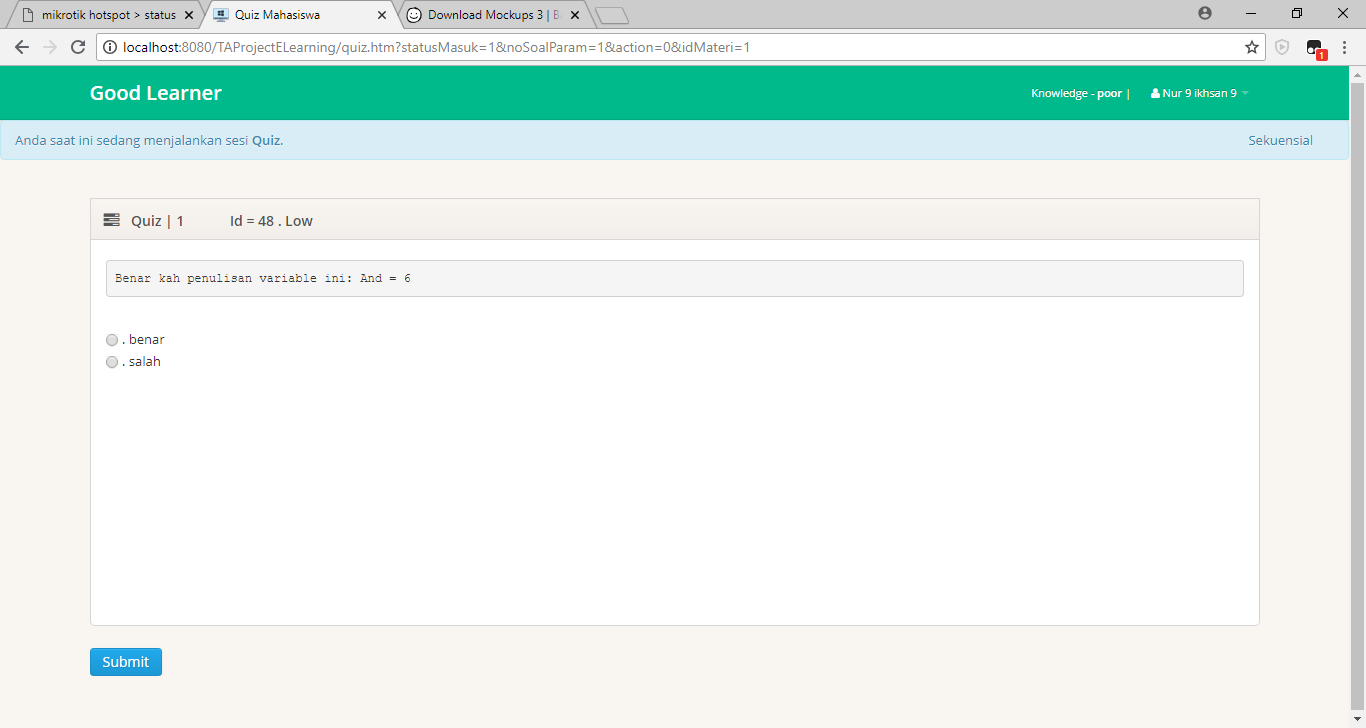
**5.3.2.2 Halaman Utama** 🡪 ***Quiz***

Halaman ini berfungsi untuk mengerjakan *Quiz* untuk Mahasiswa. Mahasiswa akan memilih Materi yang akan menjadi *Quiz*, setelah itu Mahasiswa akan mengerjakan *Quiz*, kemudian jika telah selesai maka akan muncul *Form* Nilai yang menampilkan *Score* yang didapat mahasiswa, Untuk soal yang di tampilkan adalah soal random atau soal dari *Good Learner*. Halaman ini dapat diakses dengan cara memilih menu **Halaman Utama 🡪 *Quiz***. Adapun tampilannya sebagai berikut:



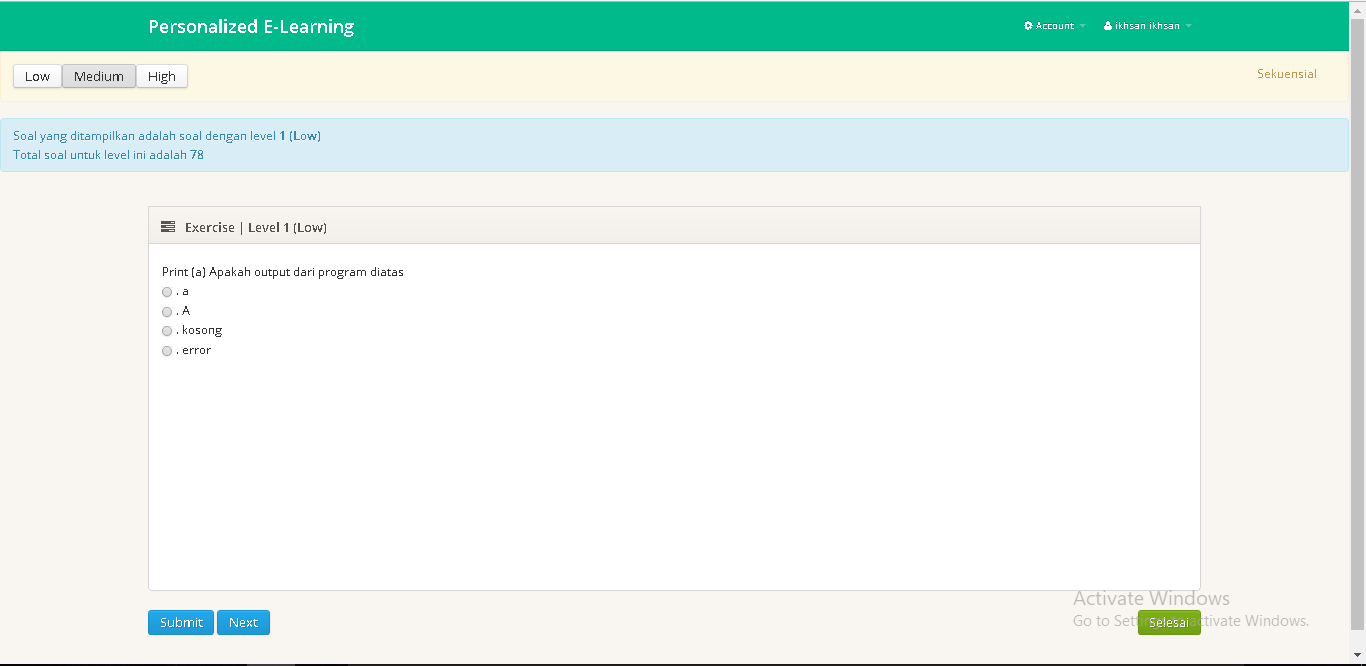
**Gambar 5.8** Halaman Materi Soal

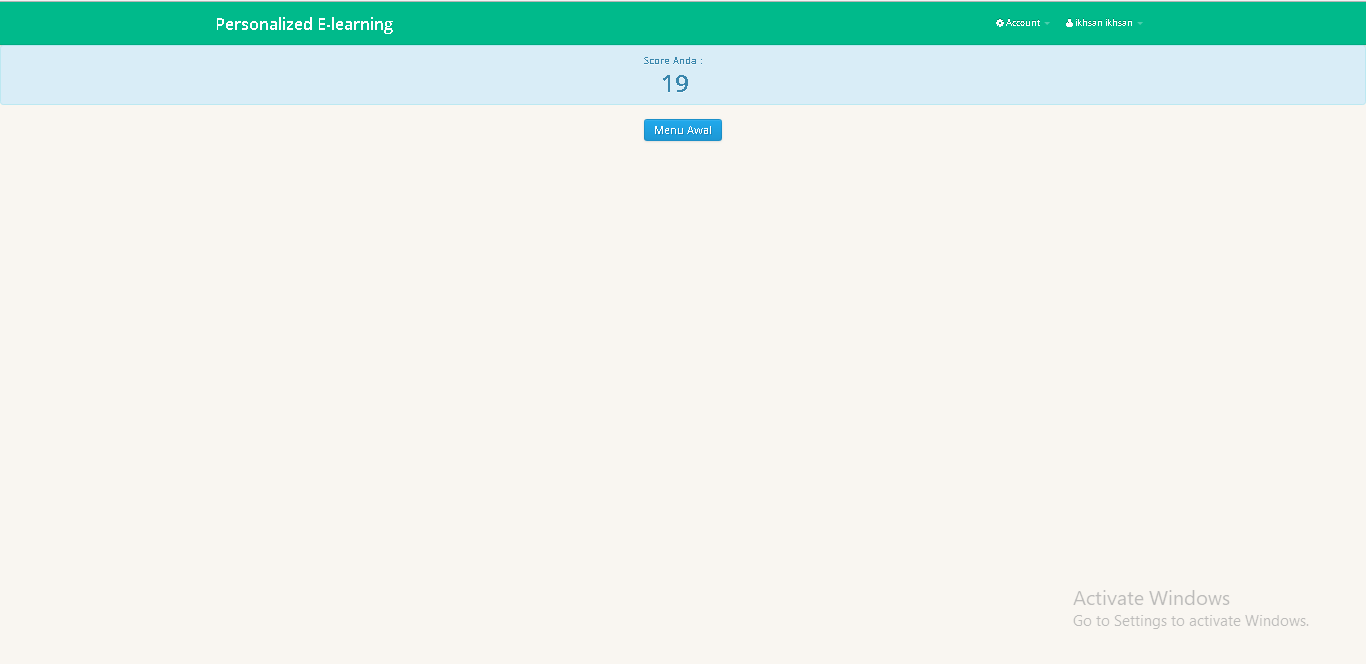
Mahasiswa memilih Materi Soal dengan mengklik tombol berikut .

Kemudian muncul Tampilan Soal *Quiz*.

**Gambar 5.9** Halaman Soal *Quiz*

Di tampilan ini, mahasiswa akan menjawab soal-soal yang muncul, dan jika mahasiswa telah menjawab soal maka mahasiswa klik *tombol*

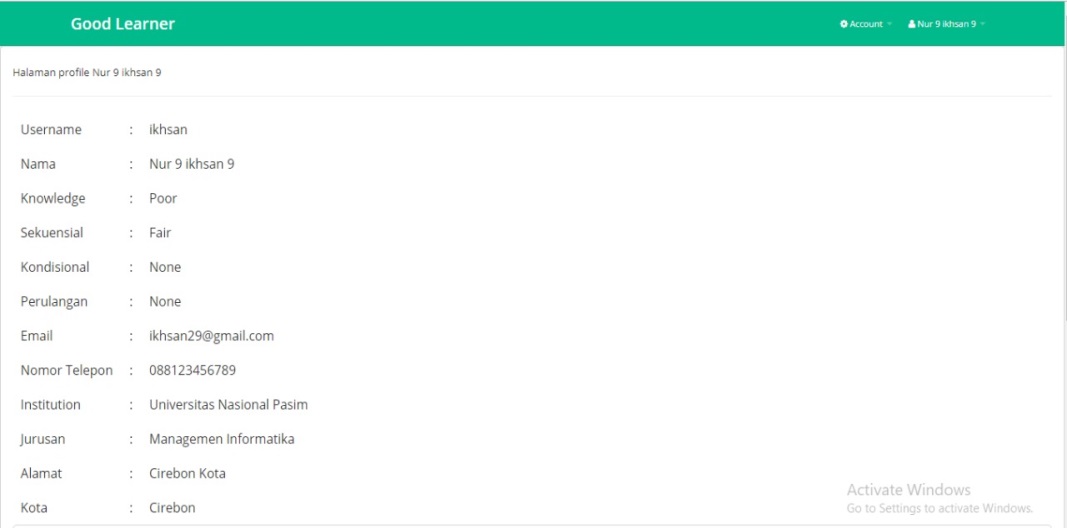




**Gambar 5.10** Halaman Hasil

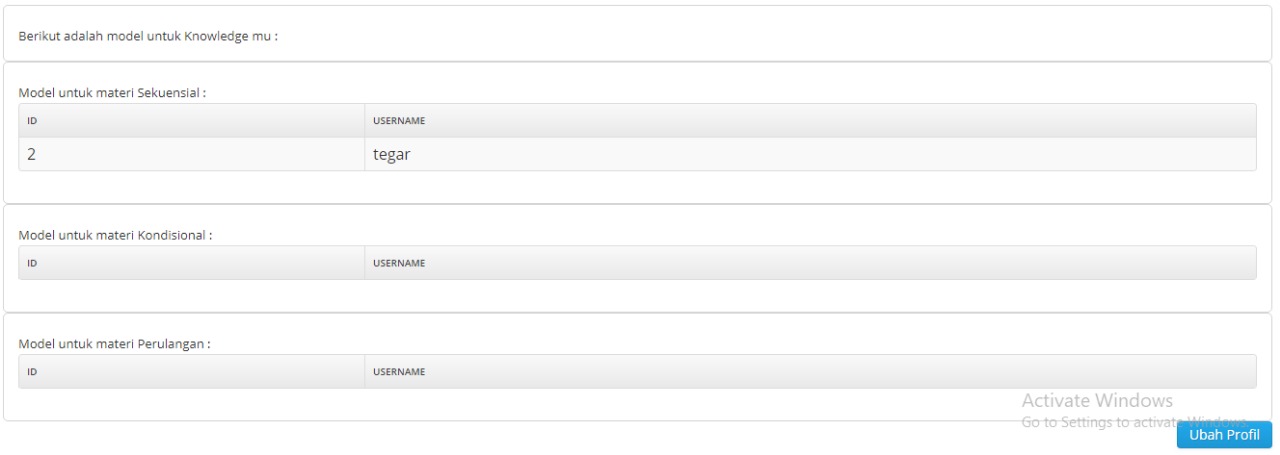
**5.3.2.3 Halaman Utama** 🡪 ***Profile***

Halaman ini berfungsi untuk menampilkan data Mahasiswa. Didalam menu ini akan ditampilkan data Mahasiswa dan *Good Learner* yang akan diikuti oleh mahasiwa tersebut. Mahasiswa hanya memilih Menu *Profile* dan akan ditampilkan data beserta *Good Learner*nya , jika terdapat *Good Learner*. Halaman ini dapat diakses dengan cara memilih menu **Halaman Utama 🡪 *Profile***.

Adapun tampilannya sebagai berikut:

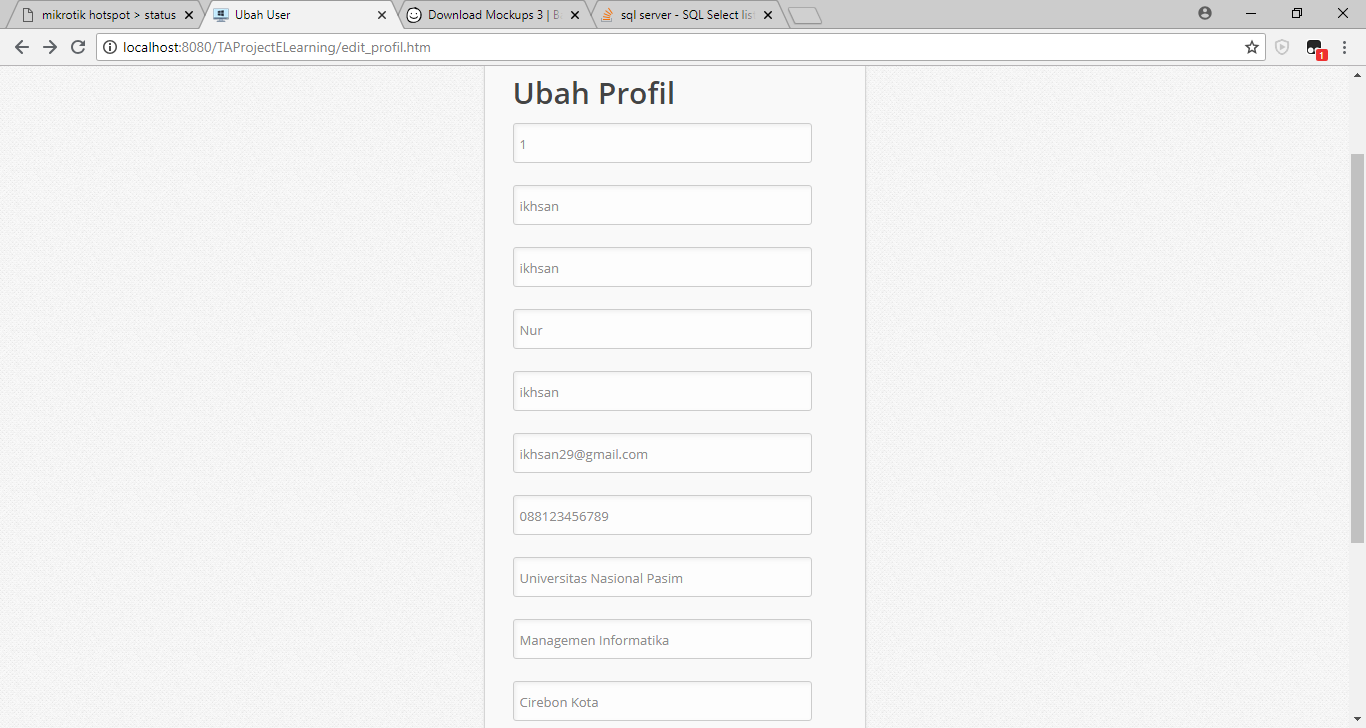
**Gambar 5.11** Halaman *Profile*

Berikut adalah *Good Learner* bagi mahasiswa tersebut.



**Gambar 5.12** Halaman *Profile* yang berisi *Good Learner*

Selanjutnya, ketika klik tombol *ubah profil* maka akan menampilkan halaman berikut:

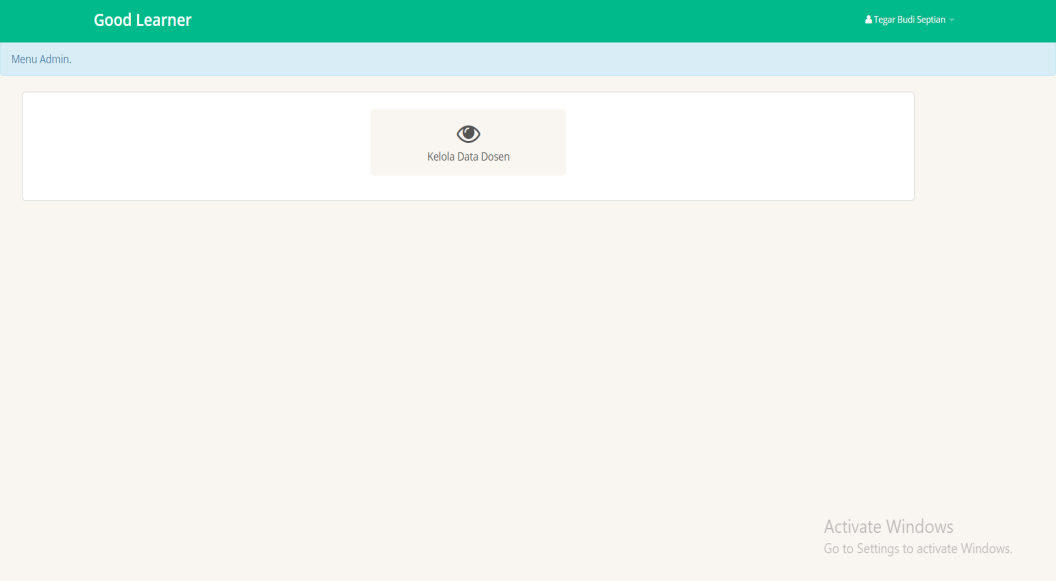


**Gambar 5.13** Halaman Ubah *Profile*

* + 1. **Implementasi Antarmuka (*Interface*) untuk *Admin***

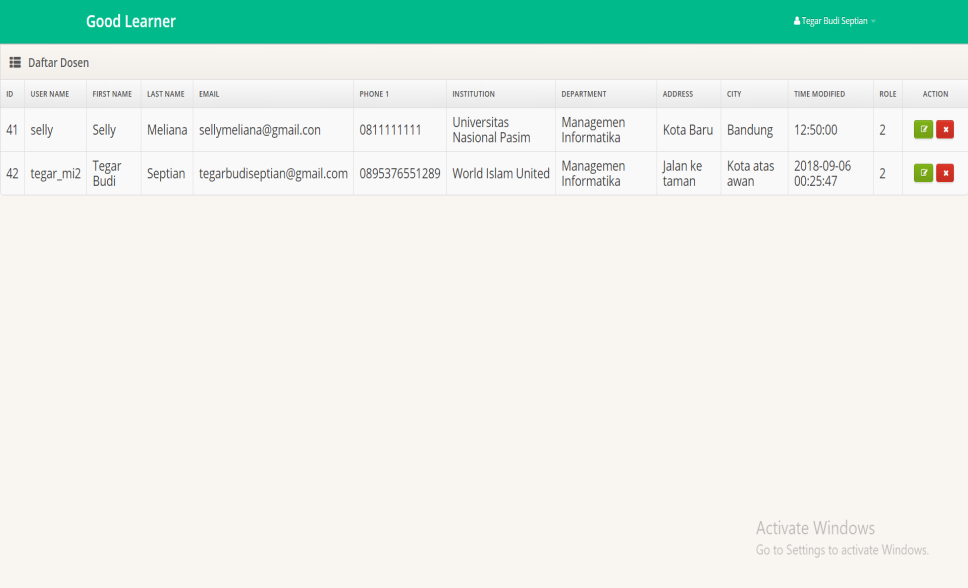
**5.3.3.1 Halaman Master Data**🡪**Kelola Data Dosen**

Halaman ini berfungsi untuk mengelola data Dosen*.* Dalam Tampilan ini *Admin* dapat melakukan tambah data Dosen, edit data Dosen dan hapus data Dosen. Halaman ini dapat diakses dengan cara memilih menu **Halaman Master Data 🡪 Kelola Data Dosen**. Adapun tampilannya sebagai berikut:

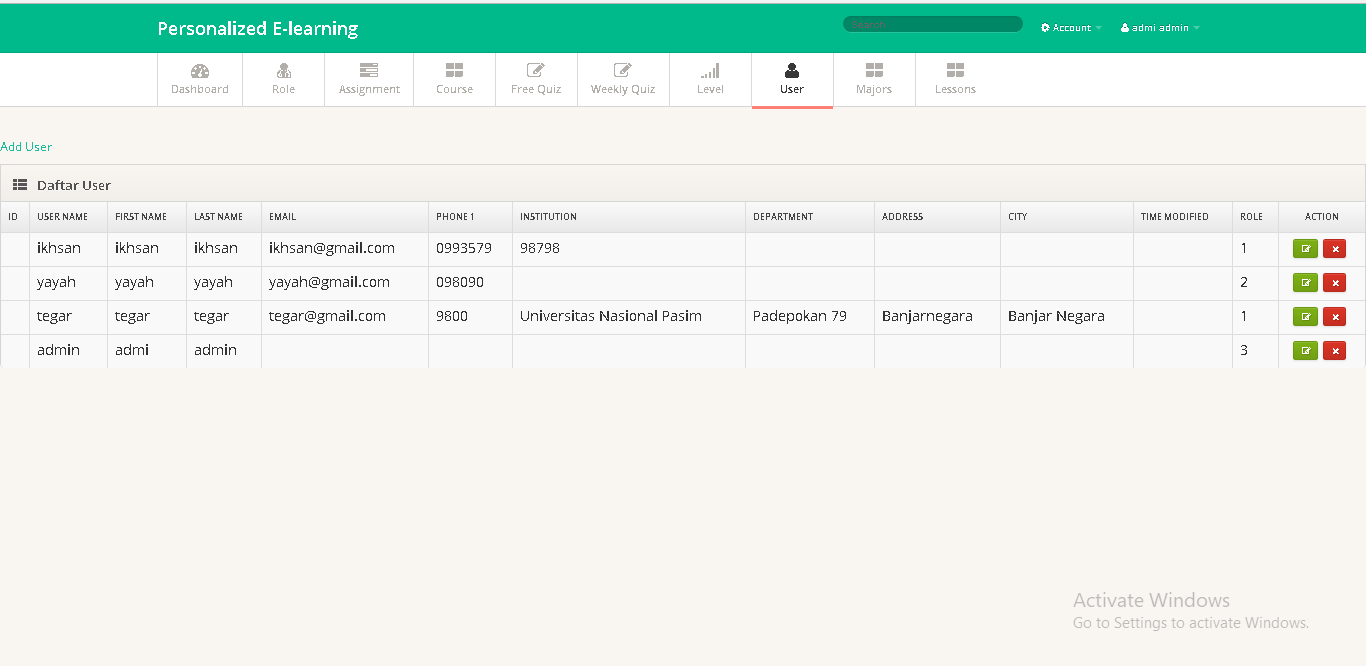
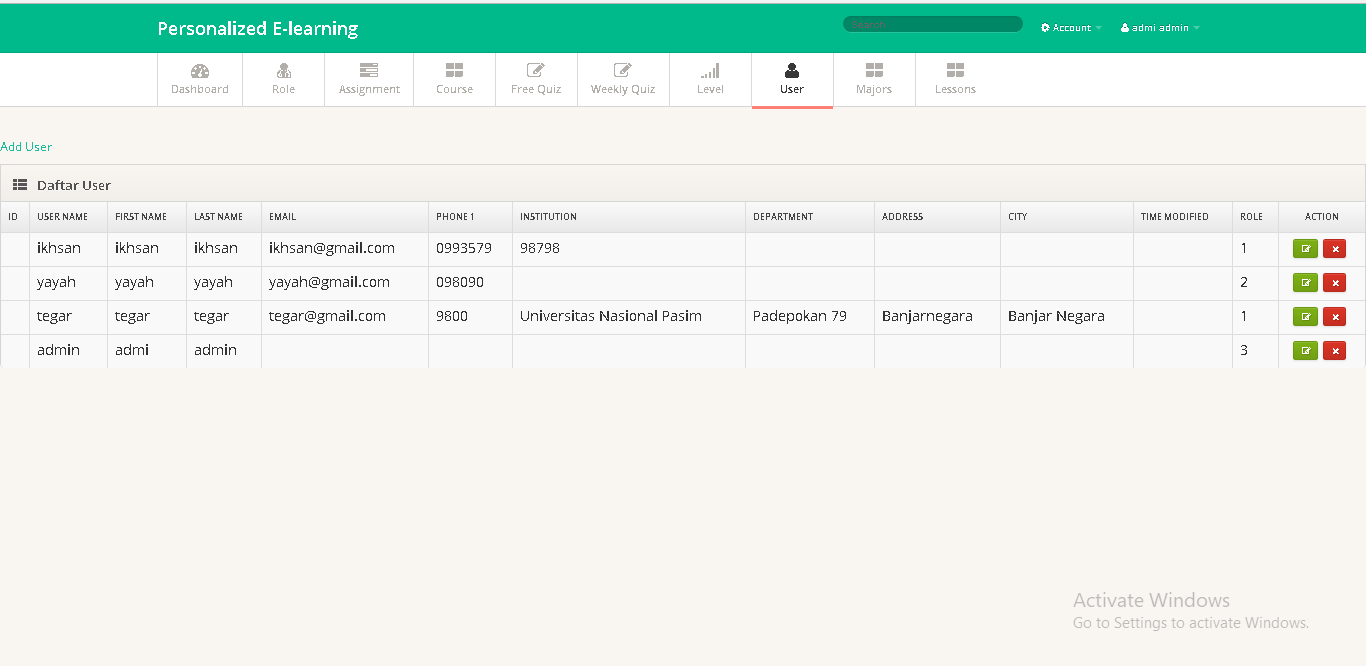
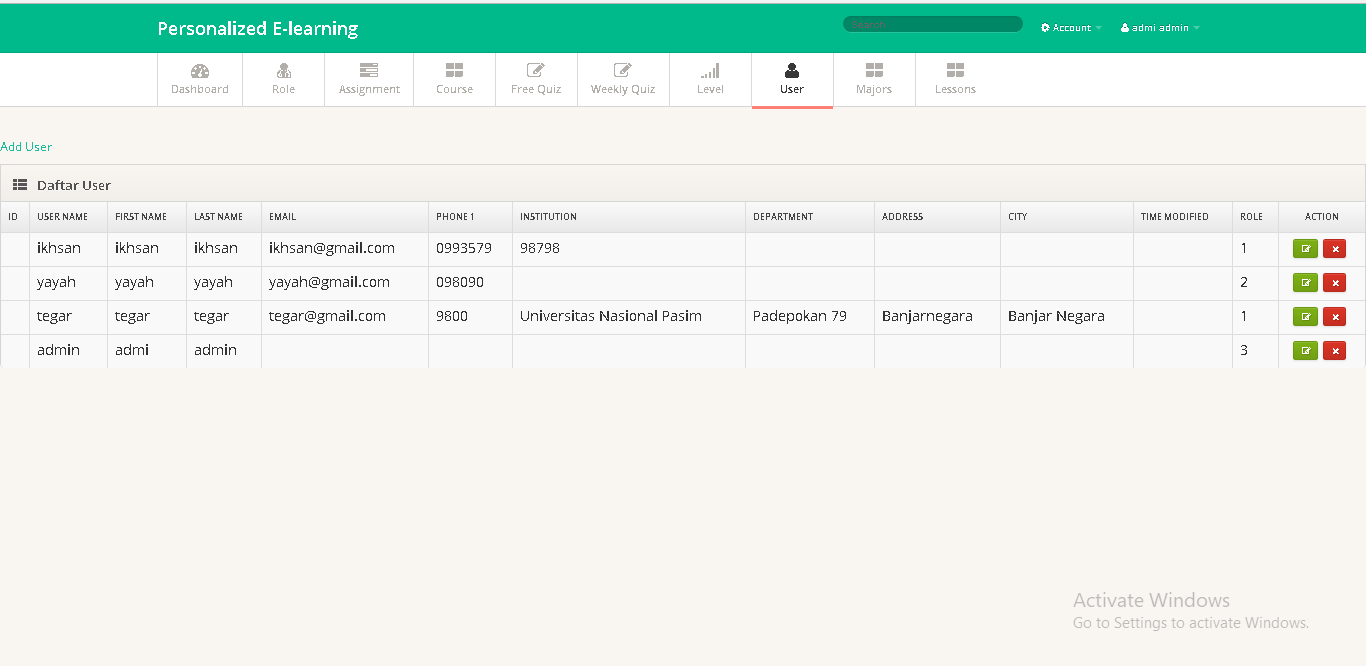
****

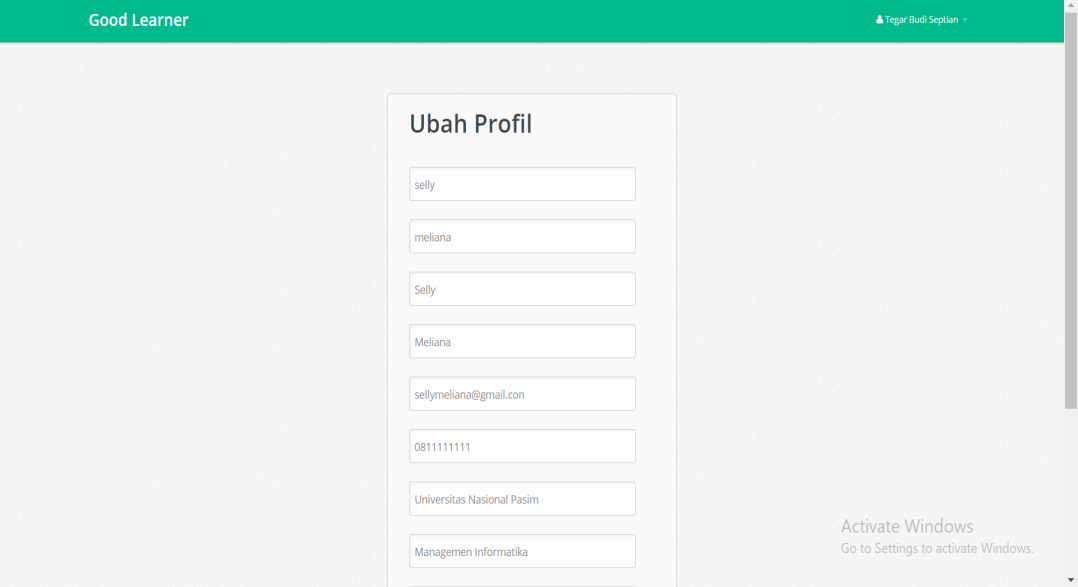
**Gambar 5.14** HalamanMenu *Admin*

Selanjutnya apabila *Admin* memilih Menu Kelola Data Dosen maka tampilan berikut yang muncul.



**Gambar 5.15** HalamanMenu *Admin:* Kelola Data Dosen

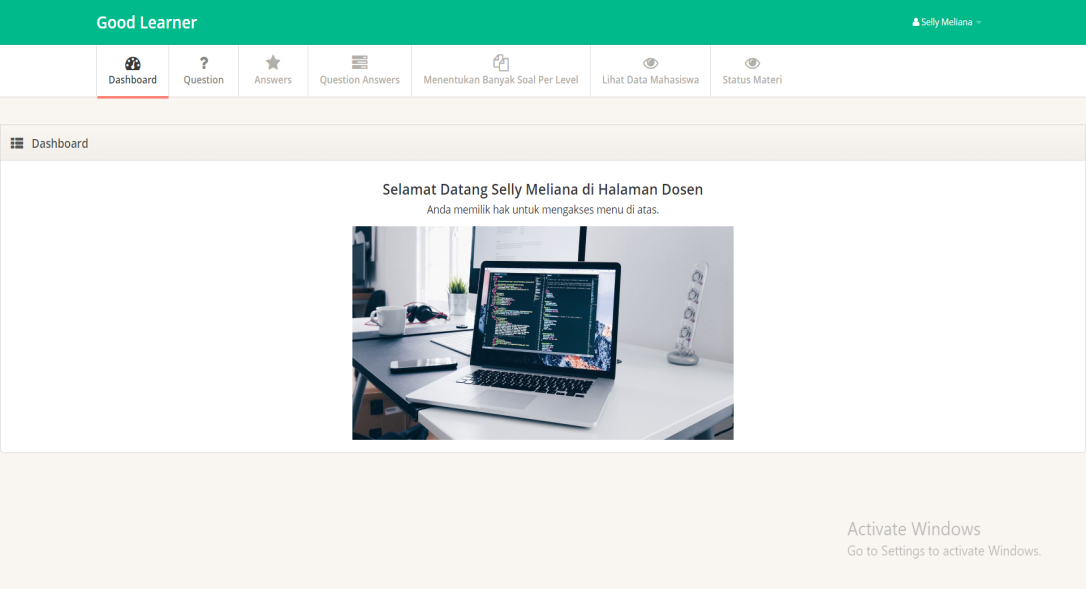
*Admin* dapat menghapus data dengan mengklik  dan mengedit Data Dosen dengan tombol . Ketika Admin mengklik tombol , maka tampilan berikut yang akan muncul.



**Gambar 5.16** HalamanMenu *Admin:* Ubah Data Dosen

* + 1. **Implementasi Antarmuka (*Interface*) untuk Dosen**

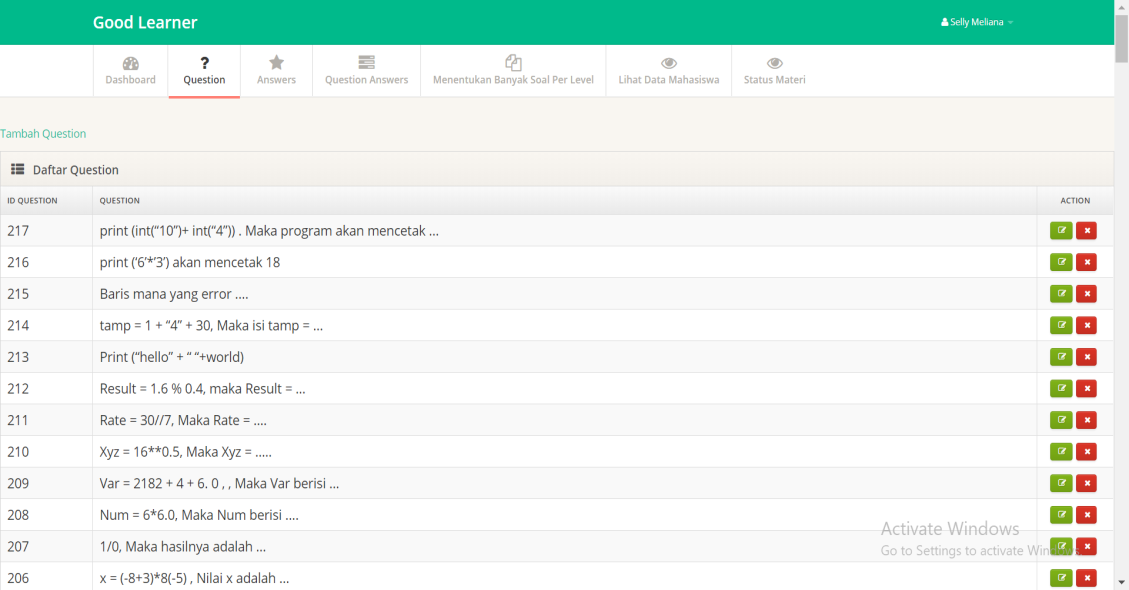
Halaman dosen adalah halaman yang akan di tampilkan ketika *user* yang di inputkan ketika *login* memiliki *role* sebagai dosen, tampilan halaman dosen sebagai berikut :



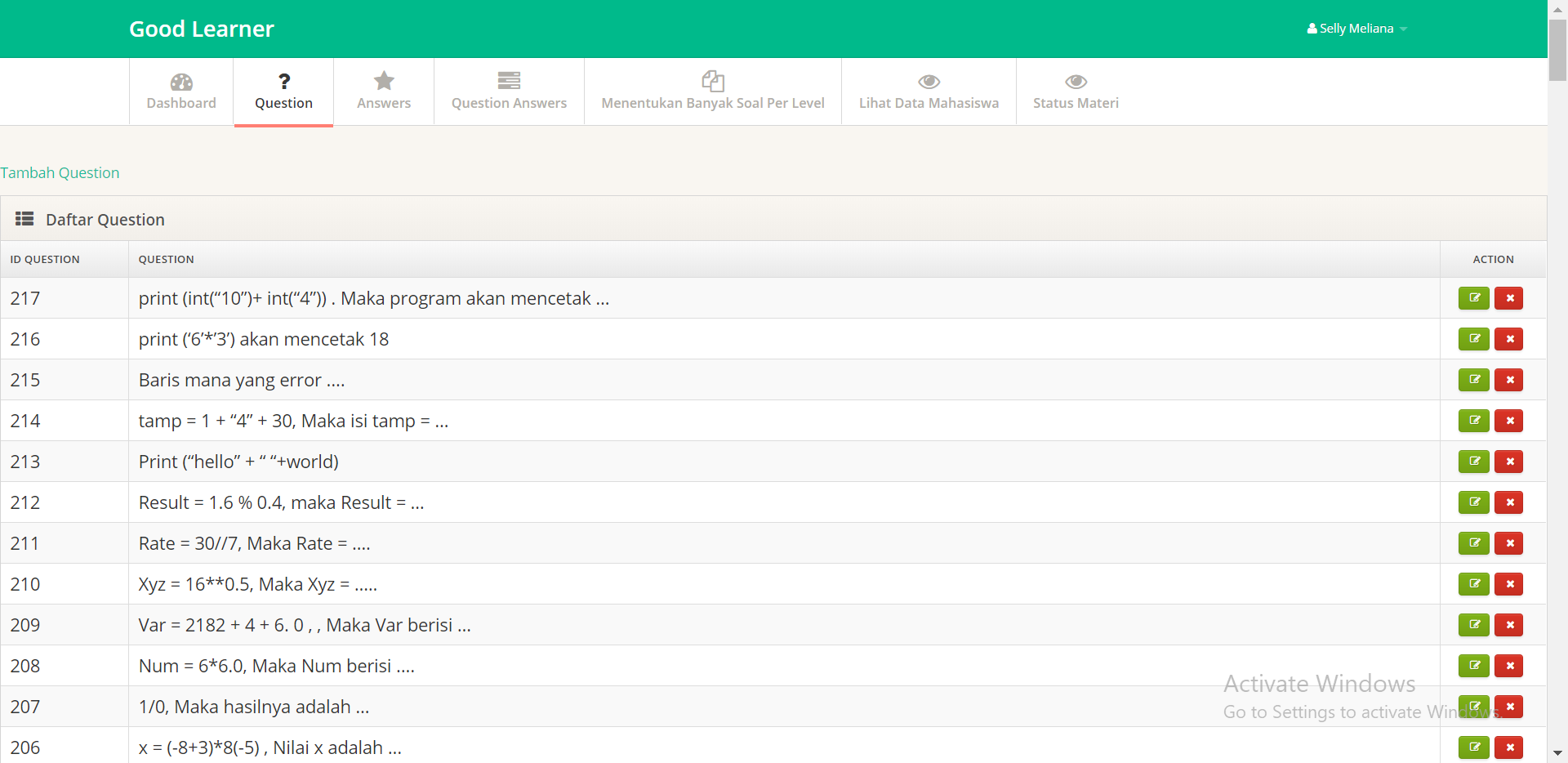
**Gambar 5.17** Halaman Dosen

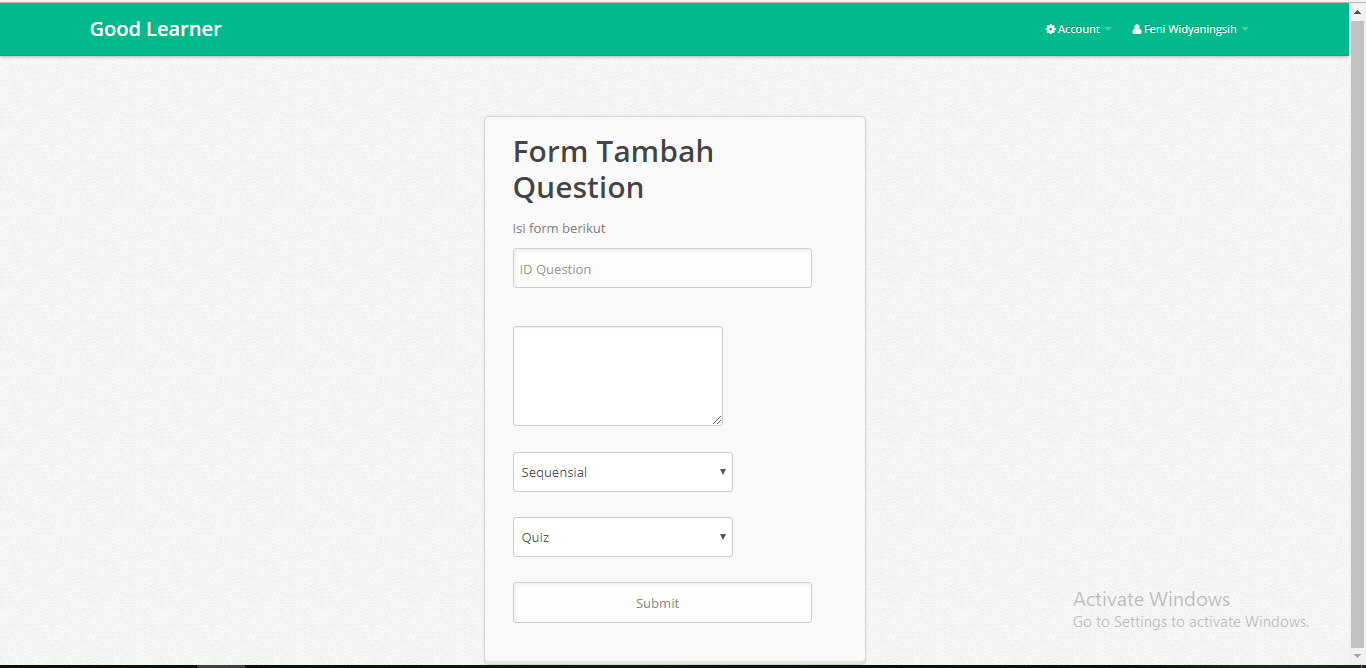
**5.3.4.1 Halaman Master Data**🡪**Question**

Halaman ini berfungsi untuk mengelola data Soal. Dalam Tampilan ini Dosendapat melakukan tambah data soal, edit soal dan hapus soal. Halaman ini dapat diakses dengan cara memilih menu **Halaman Master Data 🡪 Question**. Adapun tampilannya sebagai berikut:

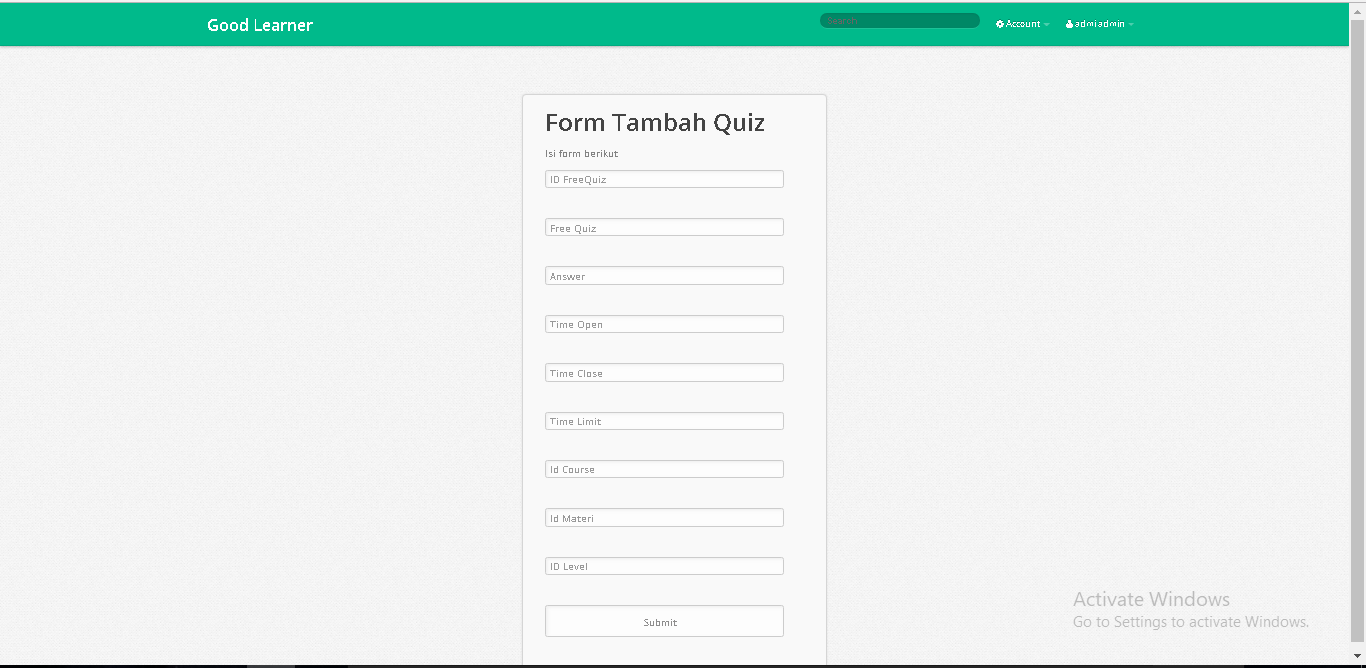
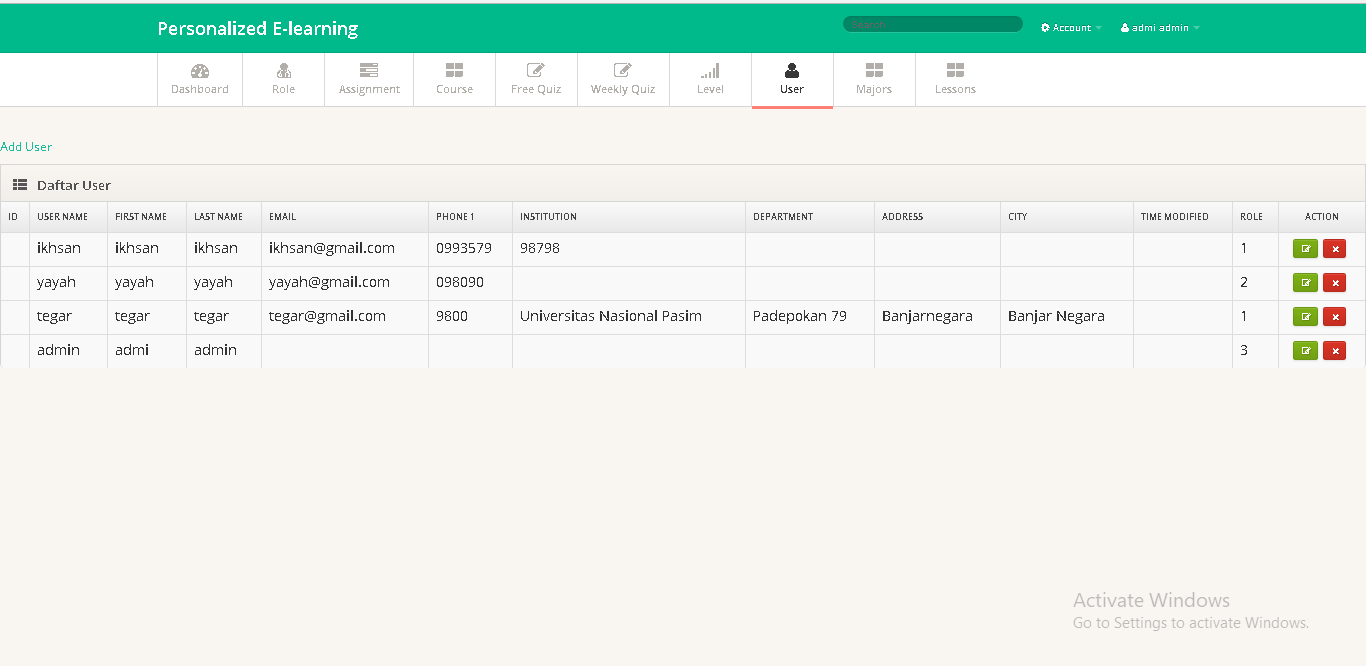
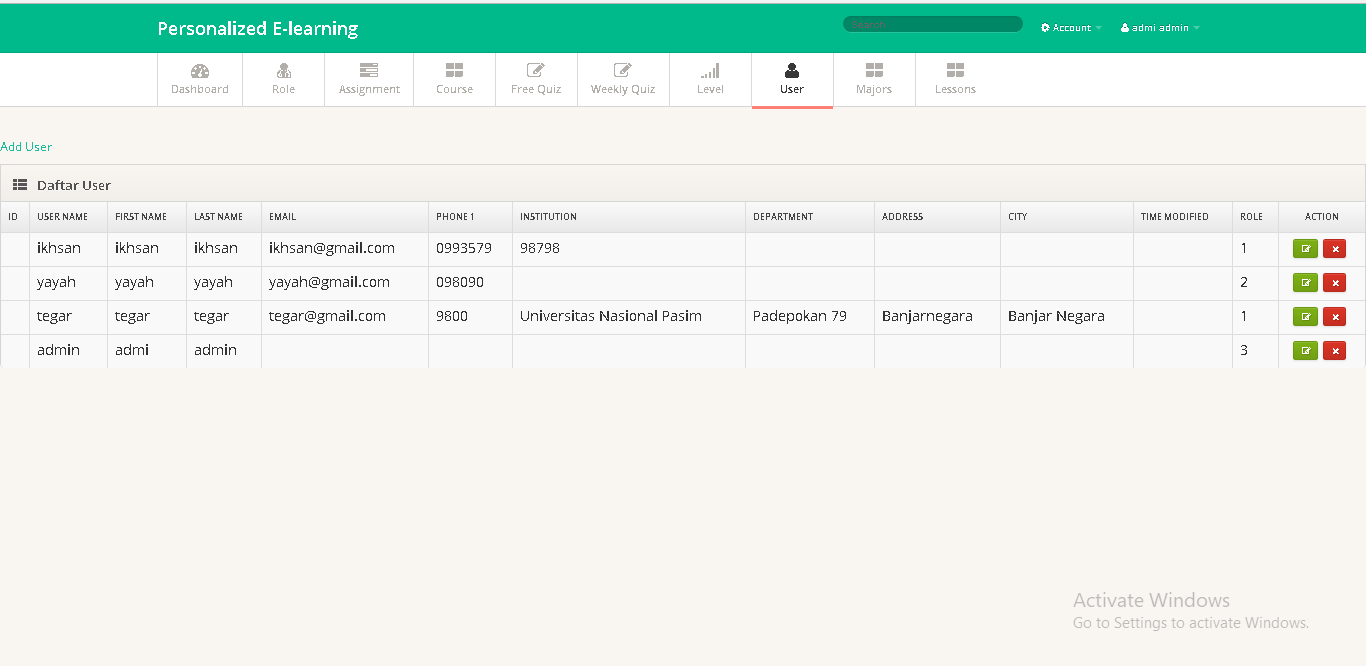


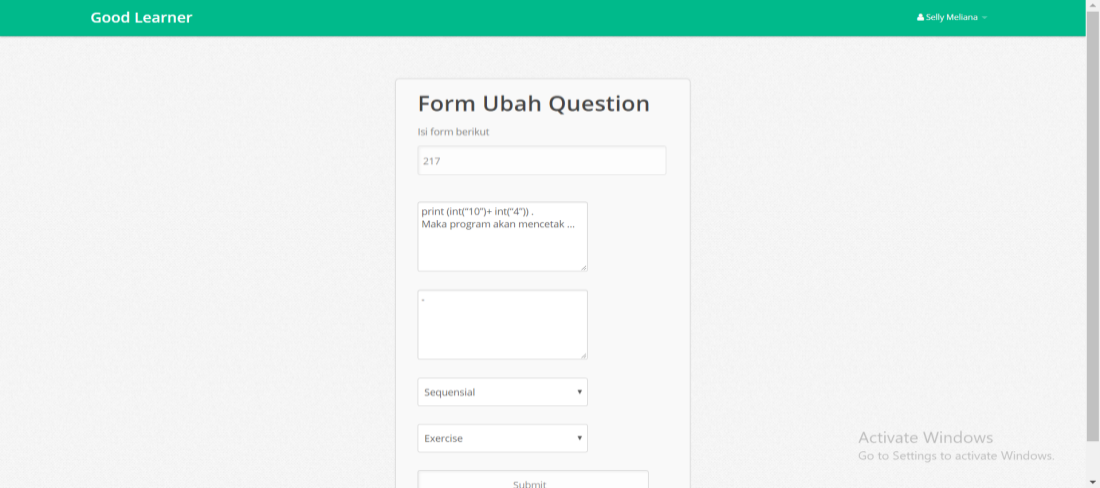
**Gambar 5.18** HalamanKelola Data *Question*

Dosen dapat menambahkan data Qusetion dengan mengklik link , maka akan muncul *form* tambah Question seperti gambar di bawah :



**Gambar 5.19** Halaman Dosen :Tambah Data Question

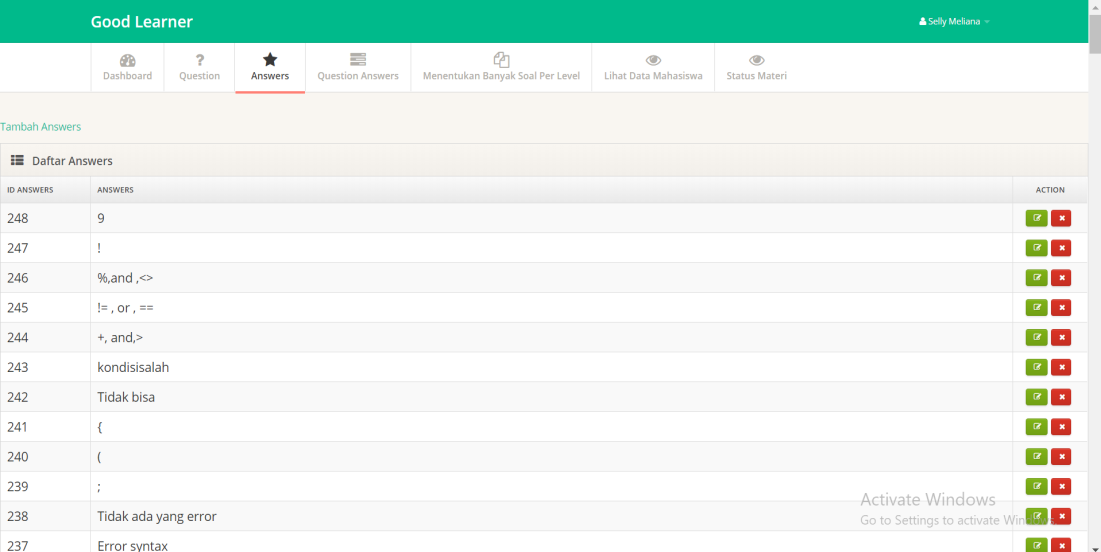
Jika data sudah terisi , maka Dosenmengklik tombol submit untuk menyimpan soal. Dosen dapat menghapus data Question dengan mengklik  .Dan untuk mengedit data Question maka klik tombol . Tampilan yang akan muncul seperti berikut.



**Gambar 5.20** Halaman Dosen :Ubah Data Question

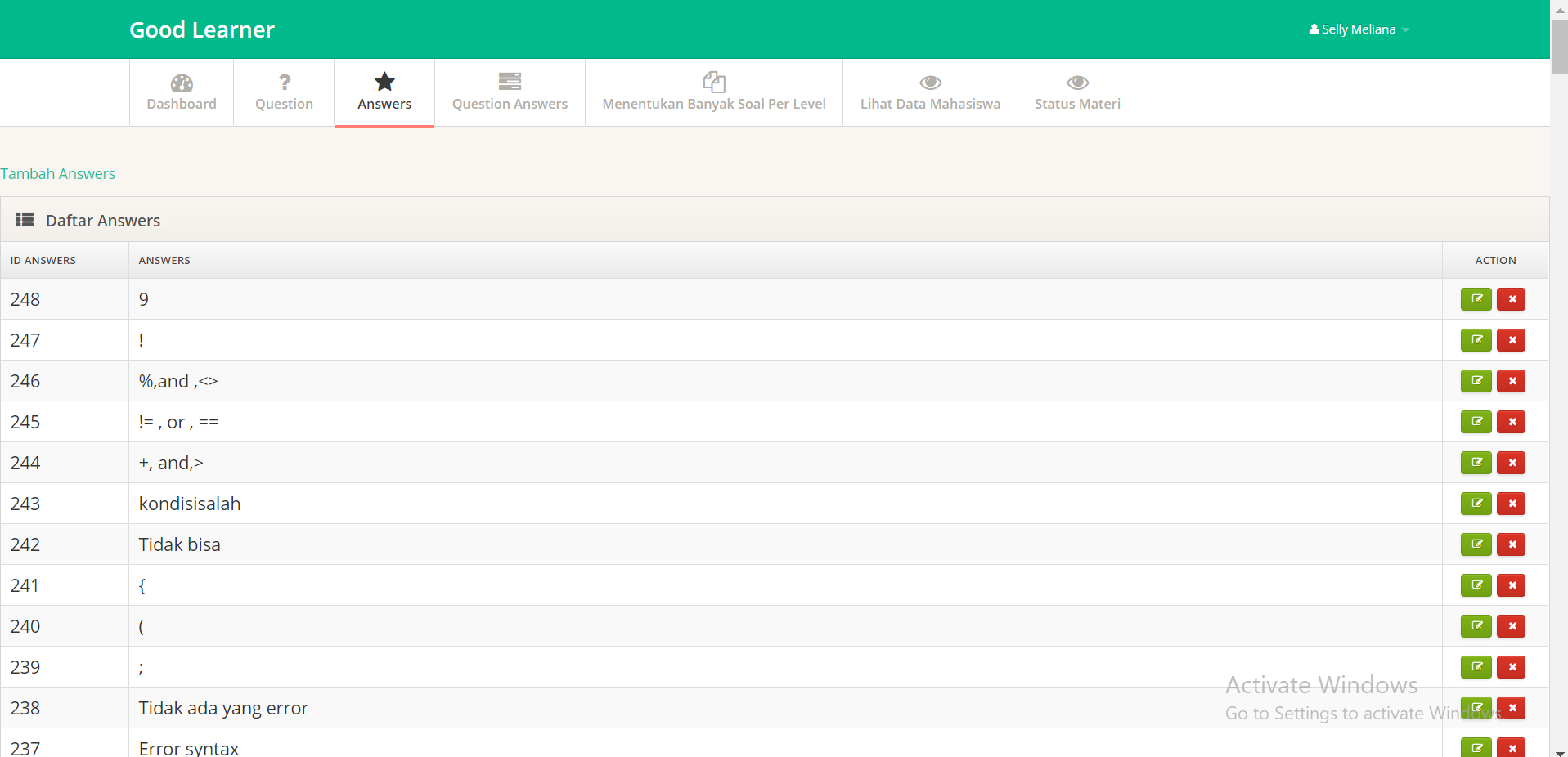
**5.3.4.2 Halaman Master Data**🡪 ***Answer***

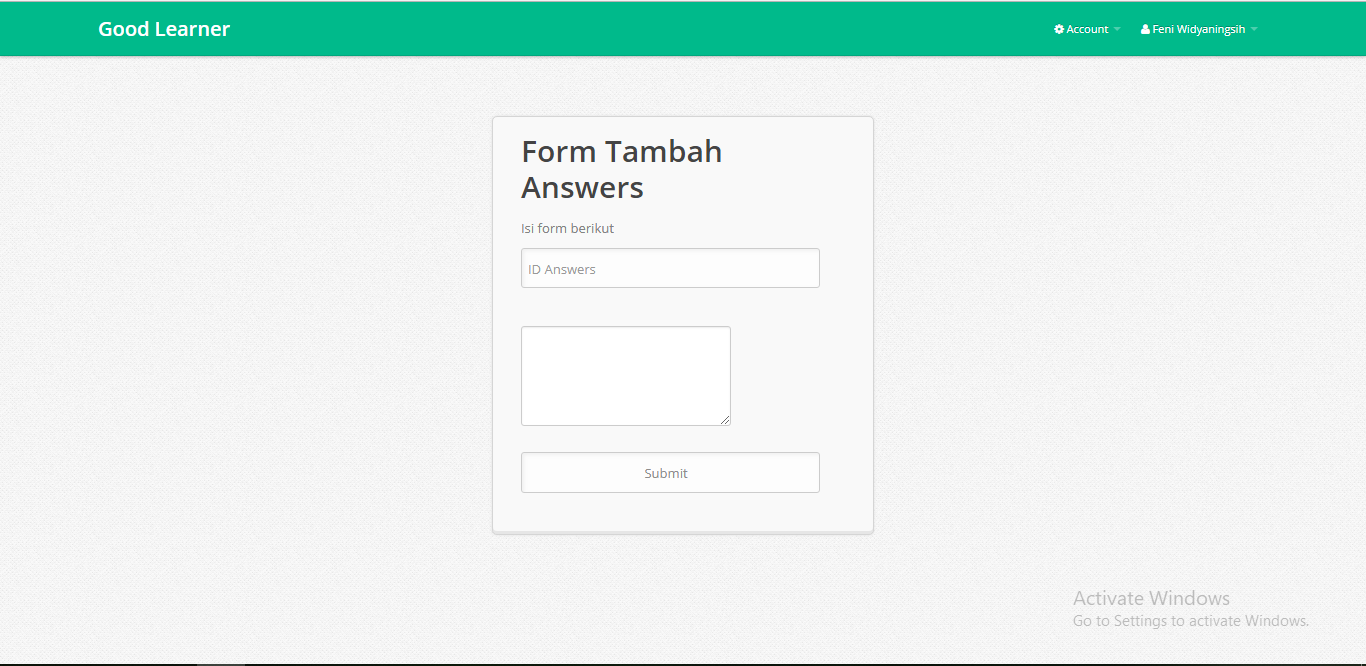
Halaman ini berfungsi untuk mengelola data Jawaban. Dalam Tampilan ini *Dosen* dapat melakukan tambah data jawaban, edit jawaban dan hapus jawaban. Halaman ini dapat diakses dengan cara memilih menu **Halaman Master Data 🡪 *Answer***. Adapun tampilannya sebagai berikut:

****

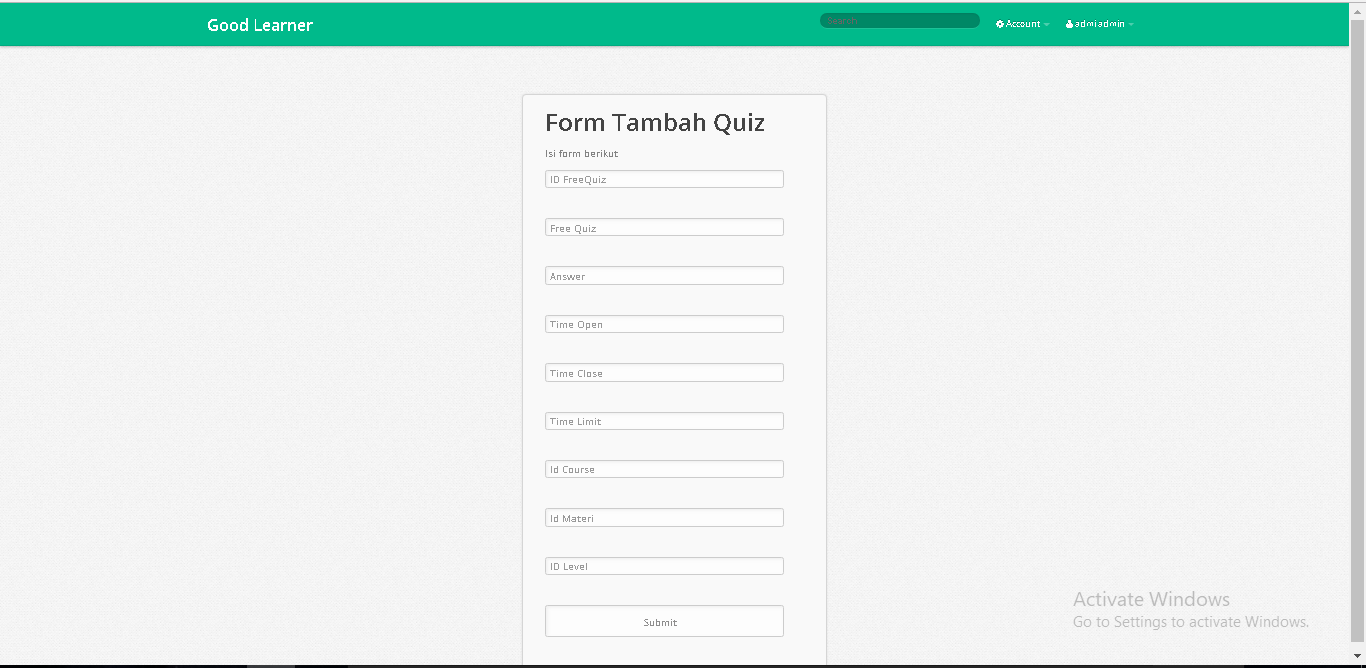
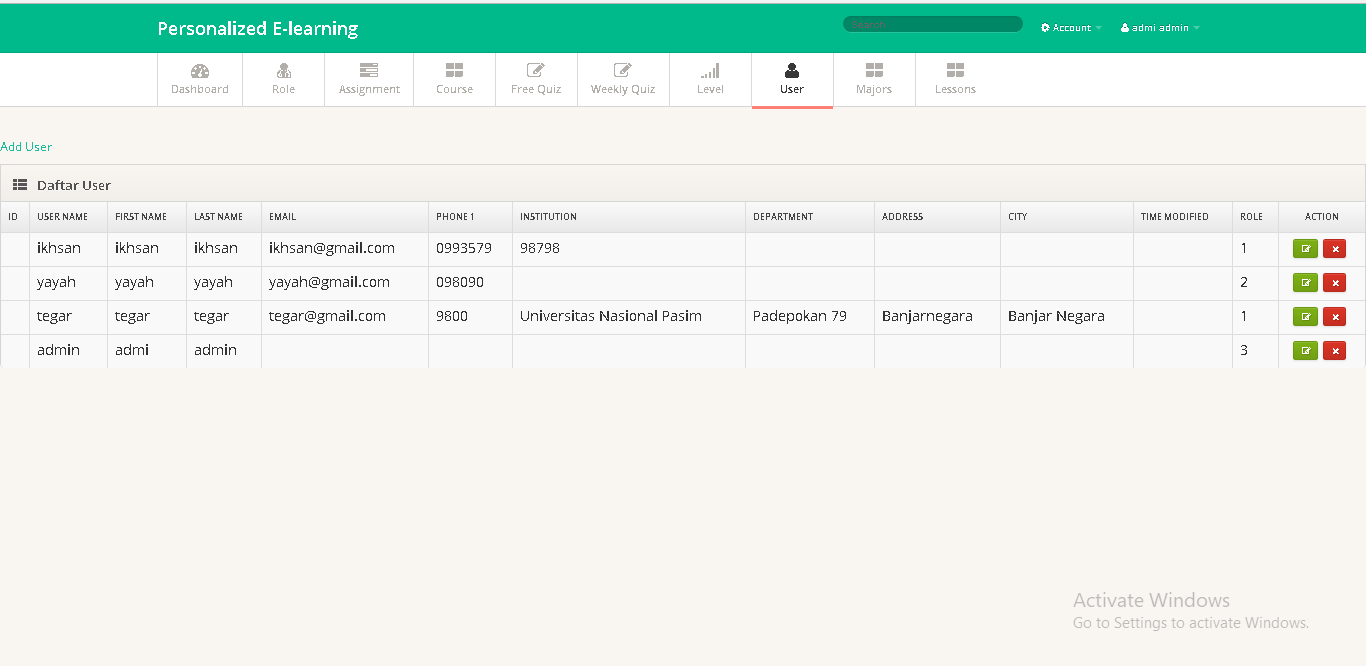
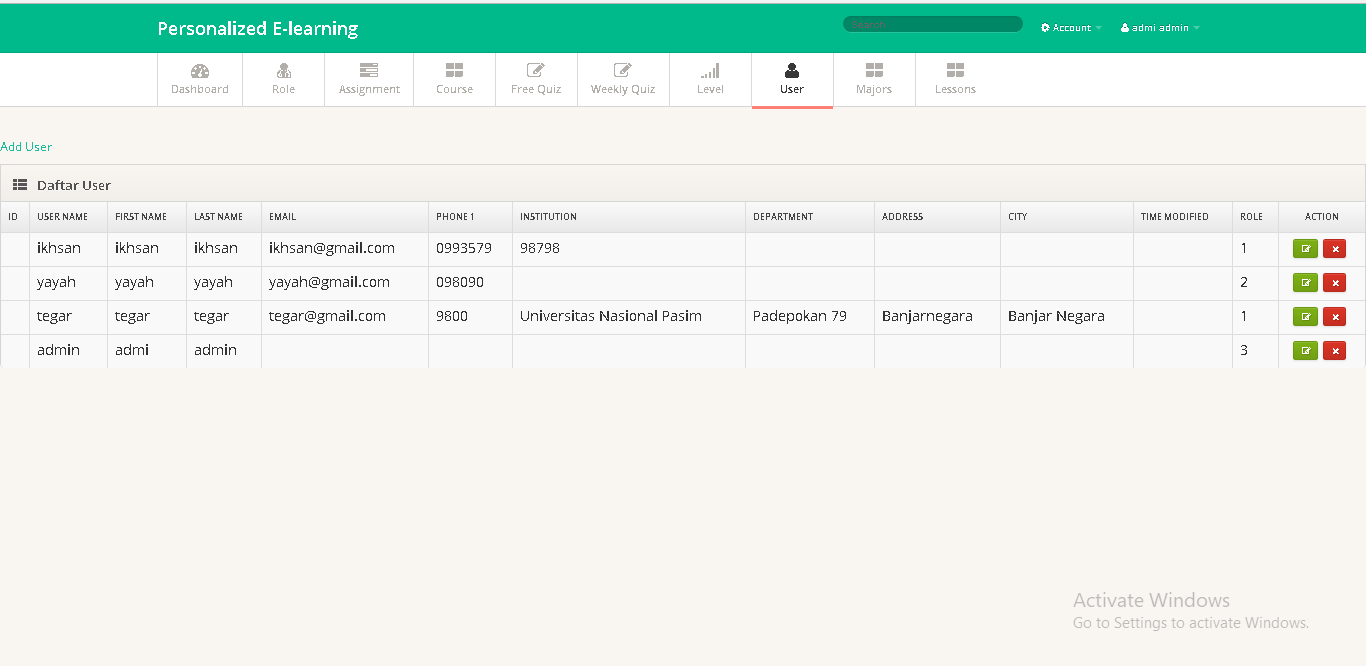
**Gambar 5.21** Halaman Dosen: Kelola Data *Answer*

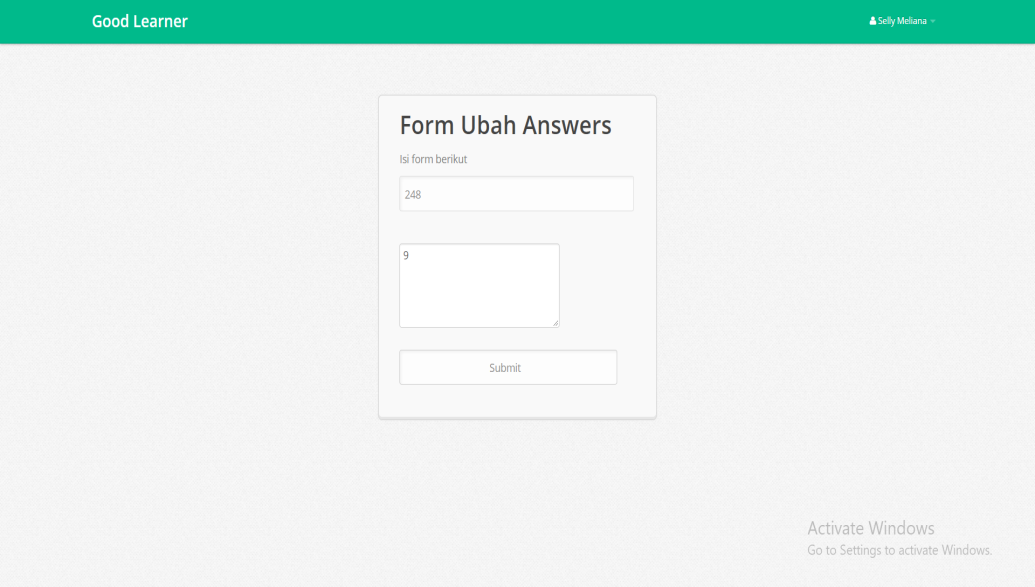
Dosen dapat menambahkan data *Question* dengan mengklik link

****, maka akan muncul *form* tambah Answer seperti gambar di bawah :



**Gambar 5.22** Halaman Dosen : Tambah Data *Answer*

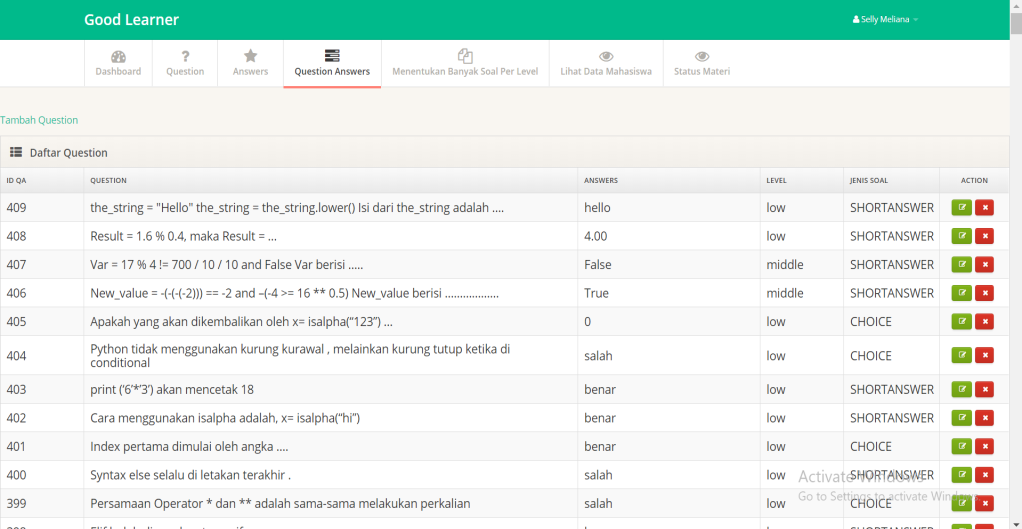
Jika data sudah terisi , maka *admin* mengklik tombol submit untuk menyimpan jawaban. Untuk menyimpan jawaban. *Admin* dapat menghapus data dengan mengklik  Dan untuk mengedit data Answer maka klik tombol . Tampilan yang akan muncul seperti berikut.



**Gambar 5.23** Halaman Dosen : Ubah Data *Answer*

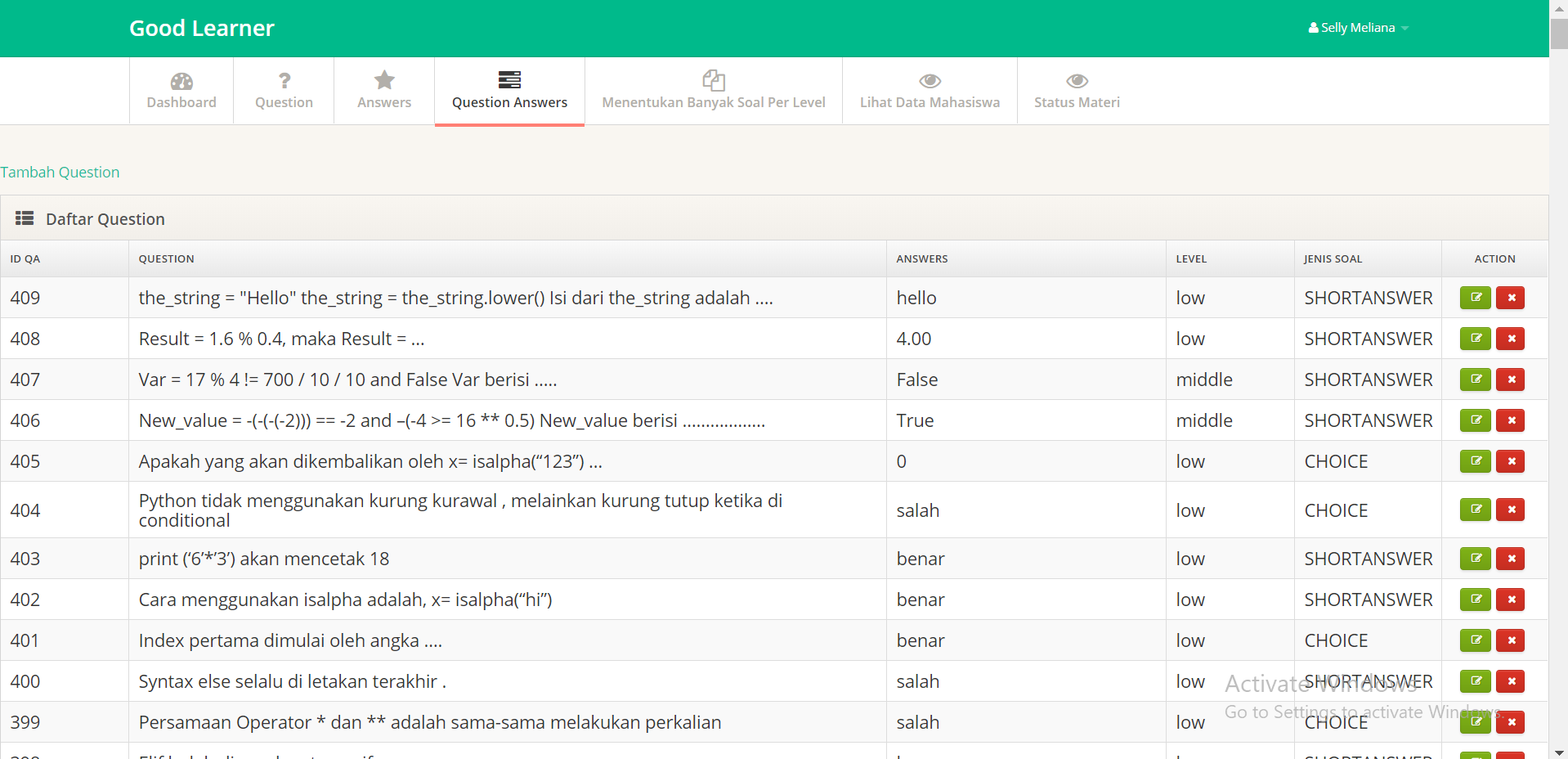
**5.3.4.3 Halaman Master Data**🡪 ***Question Answer***

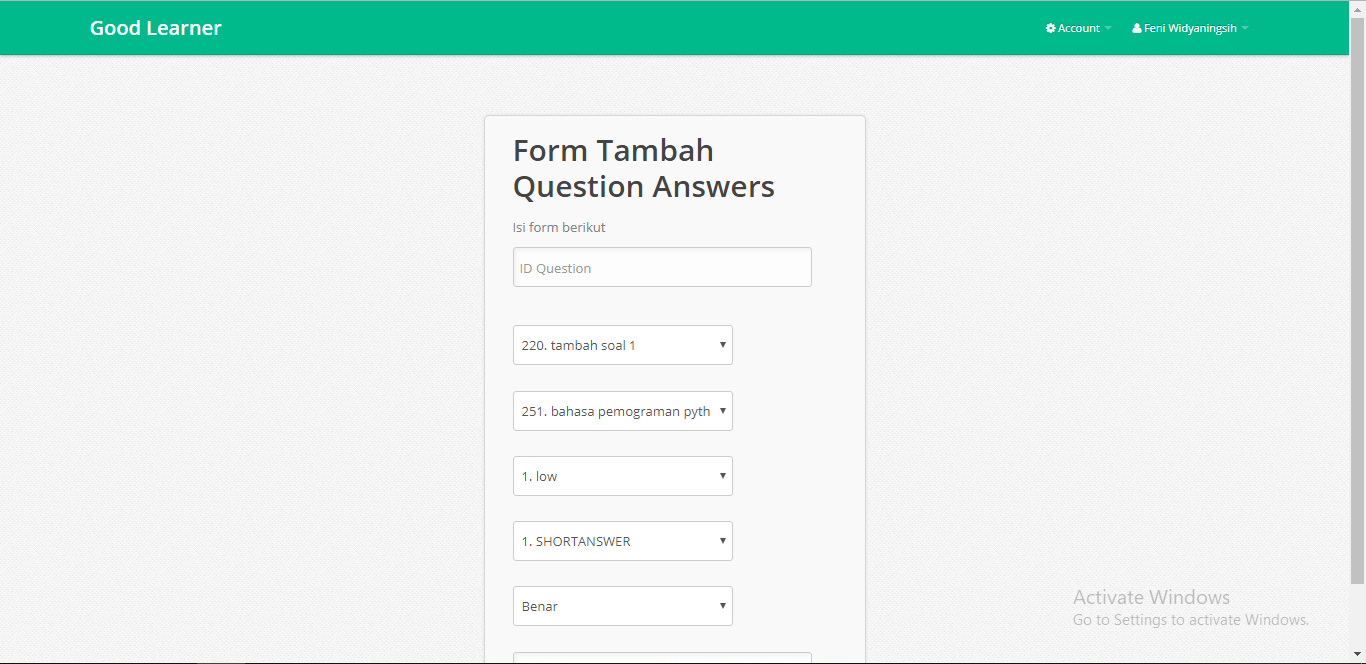
Halaman ini berfungsi untuk mengelola data Jawaban dan soal (Menggabungkan soal dan jawaban). Dalam Tampilan ini Dosen dapat melakukan tambah data jawaban dan soal, edit soal jawaban dan hapus soal jawaban. Halaman ini dapat diakses dengan cara memilih menu **Halaman Master Data 🡪 *Question Answer***. Adapun tampilannya sebagai berikut:

******

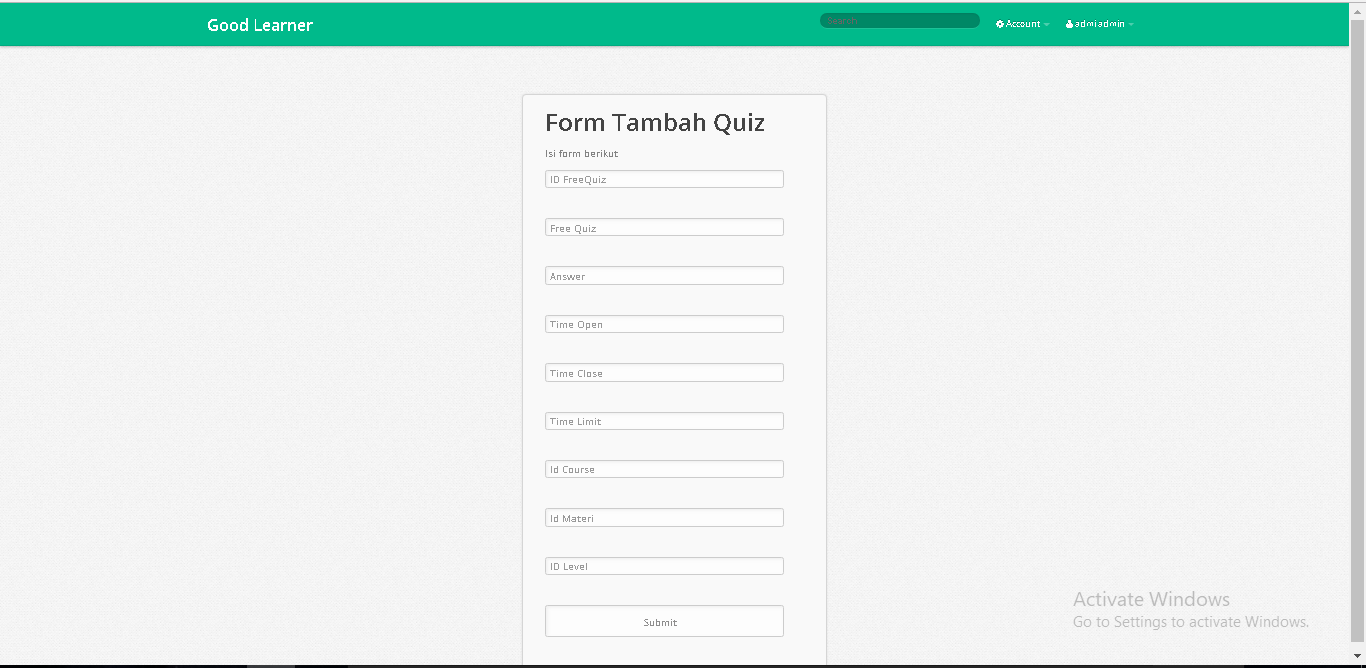
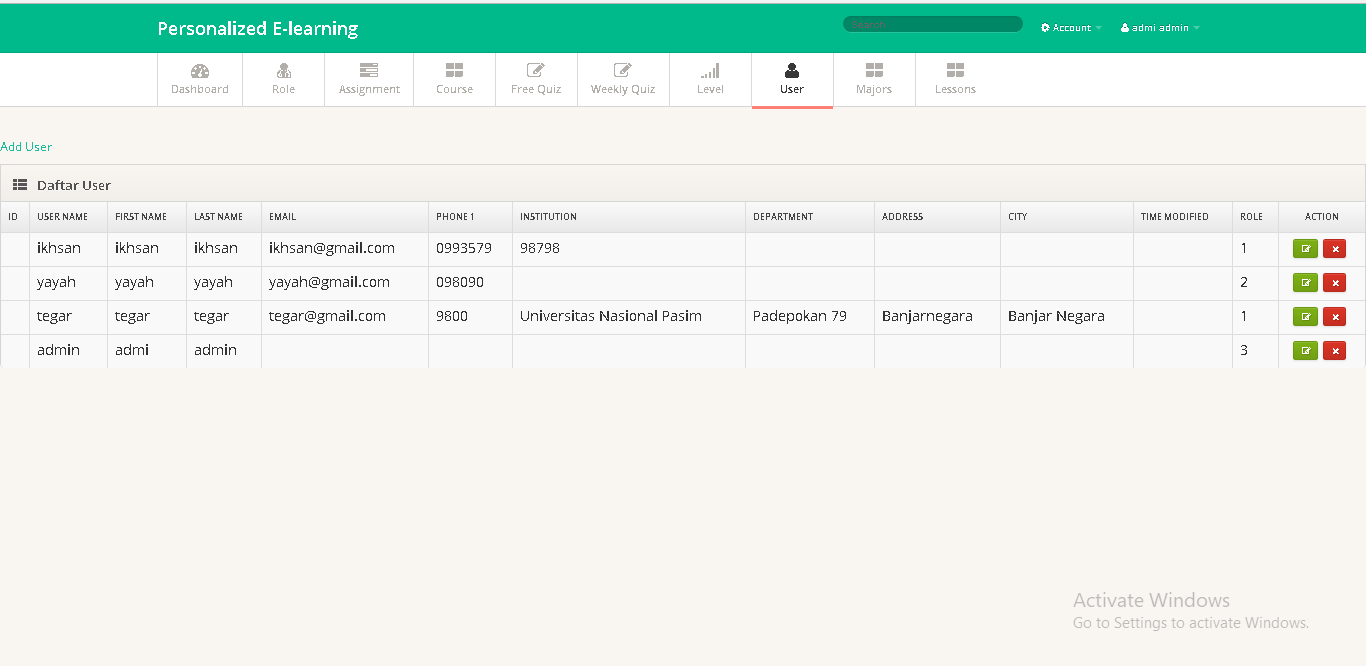
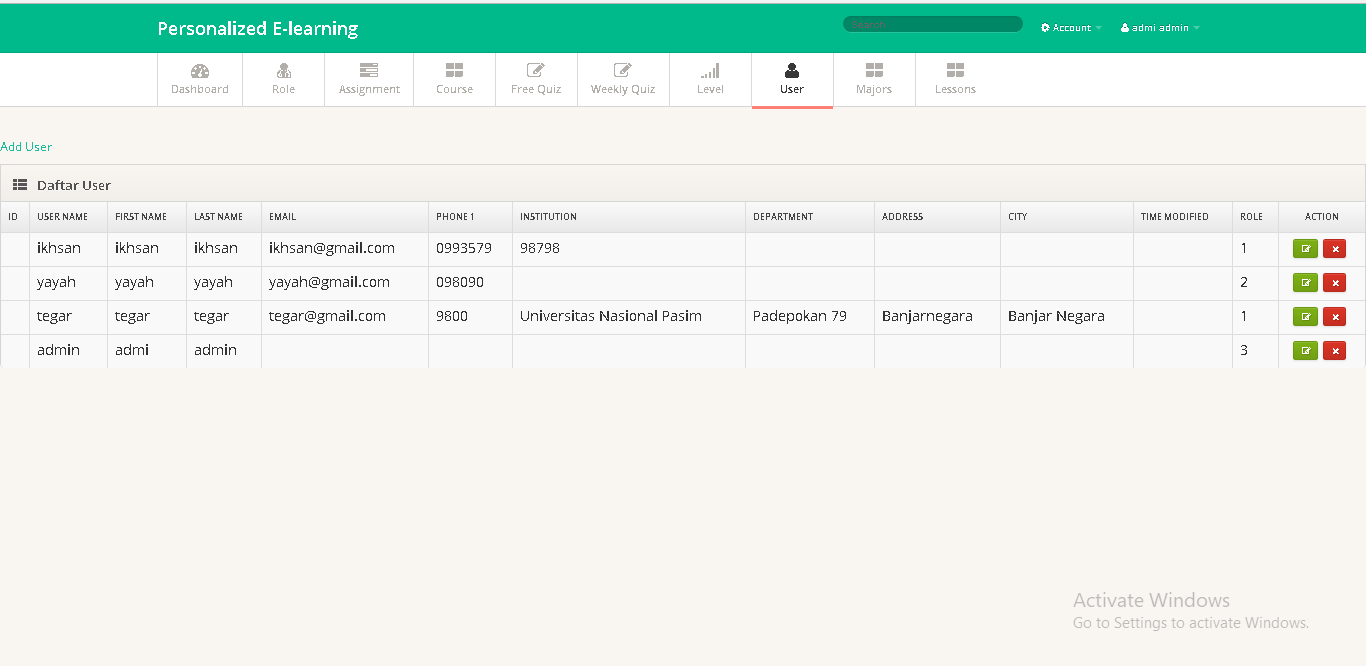
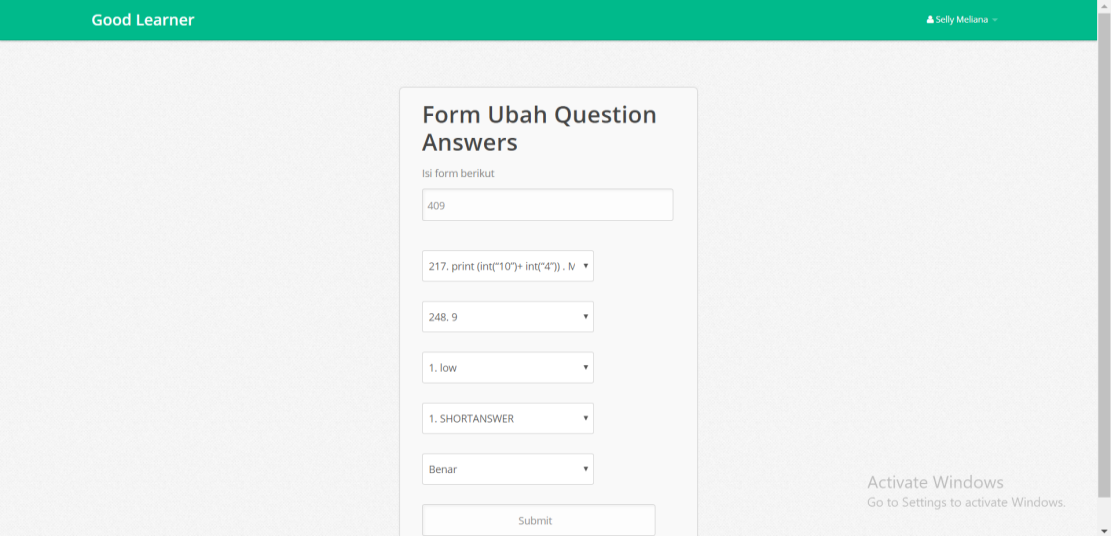
**Gambar 5.24** Halaman Dosen : Kelola Data *Question Answer*

*Dosen* dapat menambahkan soal jawaban dengan mengklik link

****** maka akan muncul *form* tambah Soal Jawaban seperti gambar di bawah :



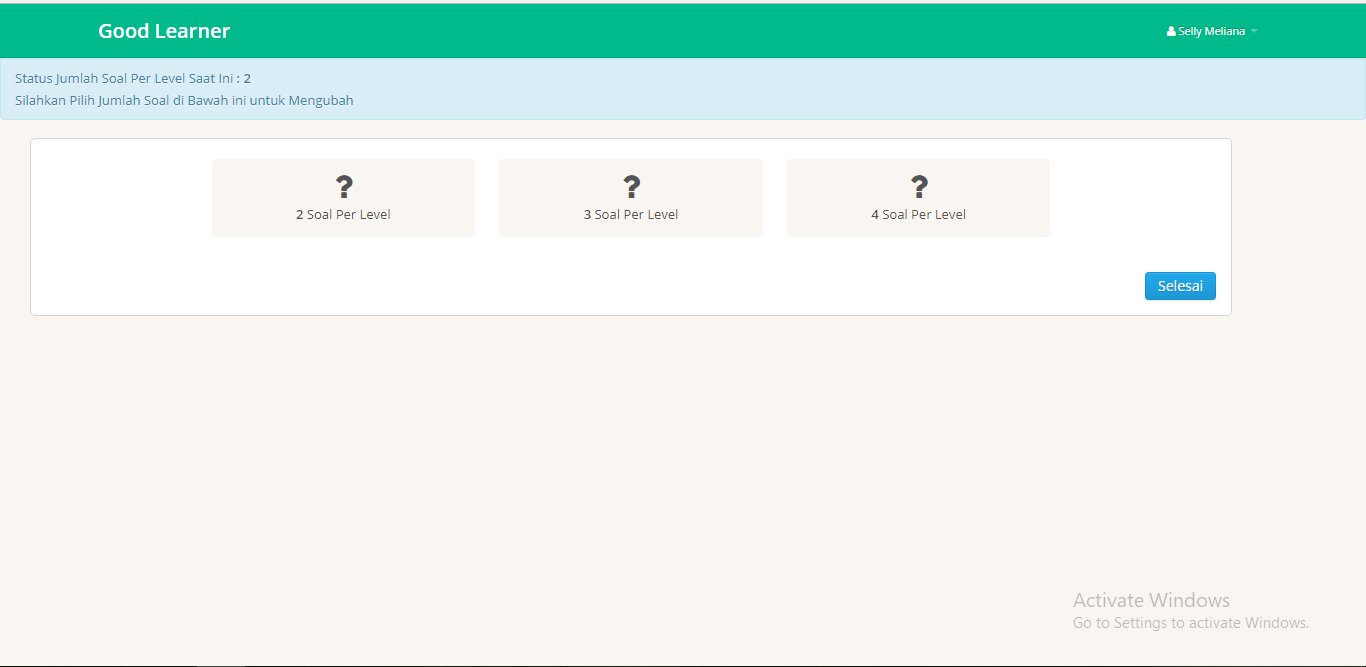
**Gambar 5.25** Halaman Dosen : Tambah *Question Answer*

Jika data sudah terisi , maka dosen mengklik tombol submit untuk menyimpan soal jawaban. Untuk menyimpan soal jawaban. Dosen dapat menghapus data dengan mengklik  dan apabila Dosen ingin mengedit data Question Answer, maka mengklik tombol . Dan akan muncul tampilan seperti berikut. 

**Gambar 5.26** Halaman Dosen : Ubah *Question Answer*

**5.3.4.4 Halaman Dosen**🡪**Menentukan jumlah soal *quiz* per level**

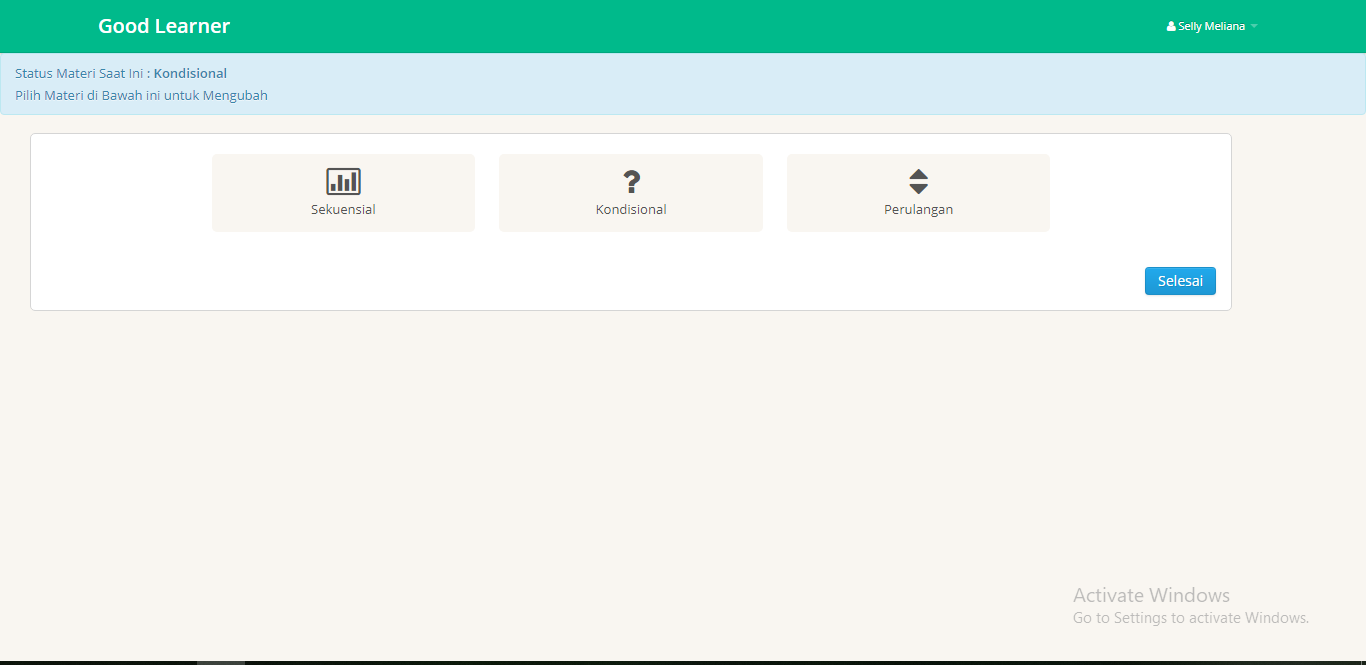
Dalam halaman ini dosen dapat menentukan banyak nya soal yang akan di berikan kepada mahasiswa. Tampilan dari halaman Menentukan jumlah soal *quiz* per level :



**Gambar 5.27** Halaman menentukan jumlah soal

**5.3.4.5 Halaman Dosen**🡪 ***Status materi***

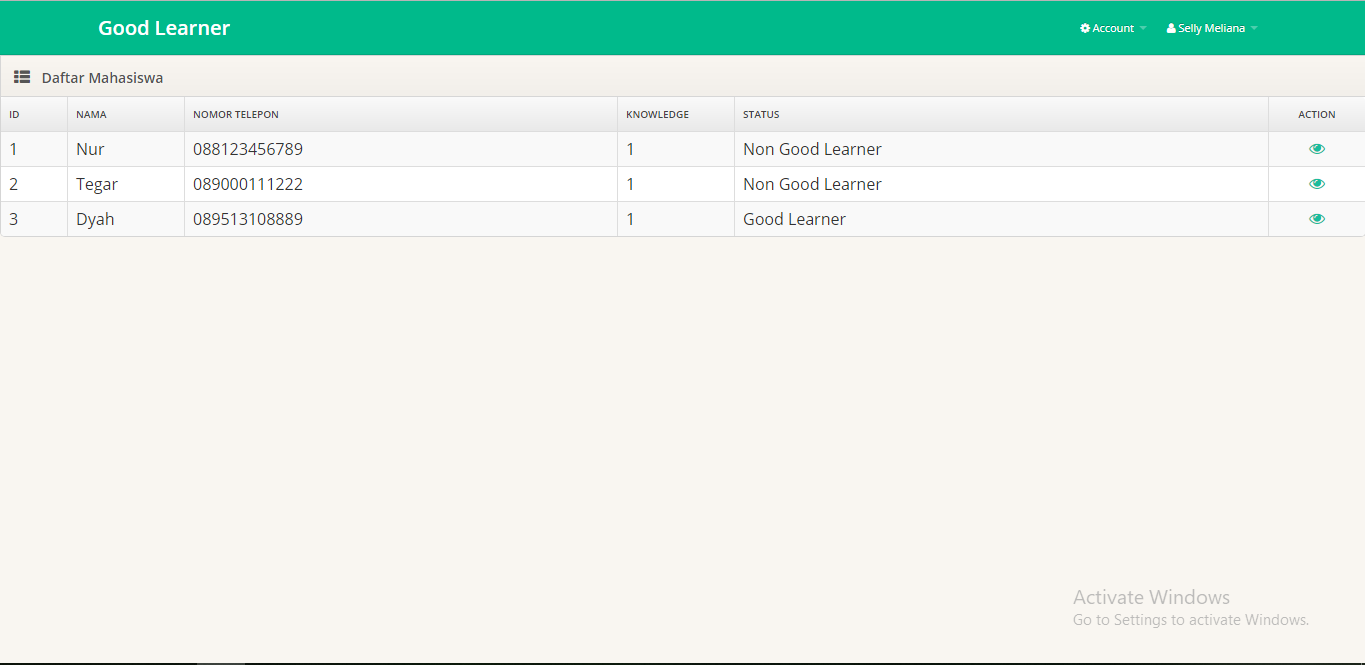
Di dalam halaman ini dosen dapat mengganti materi sesuai dengan keinginan dosen. Lebih jelasnya halaman ubah status materi sebagai berikut :



**Gambar 5.28** Halaman ubah status materi

**5.3.4.6 Halaman Dosen**🡪**Melihat mahasiswa**

Di dalam halaman ini dosen dapat melihat data mahasiswa dan melihat level dari setiap mahasiswa.



**Gambar 5.29** Halaman *view* mahasiswa

**5.3.4.7 Halaman Dosen**🡪**Lihat Data mahasiswa *– Detail* Mahasiswa**

Di dalam halaman ini dosen dapat melihat profil mahasiswa , halaman tersebut jika di tampilkan maka sebagai berikut:



**Gambar 5.30** Halaman *view* detail mahasiswa

**BAB VI**

**KESIMPULAN DAN SARAN**

1. **Kesimpulan**

Berdasarkan hasil penelitian yang telah dilakukan penulis, maka dapat disimpulkan ke dalam beberapa *point* sebagai berikut :

1. Aplikasi ini dapat mengetahui kemampuan mahasiswa dalam sub bab mata kuliah Bahasa Pemograman
2. Aplikasi menampilkan dapat menambah pengetahuan mahasiswa mengenai materi yang di kerjakan
3. Aplikasi ini digunakan diluar mata kuliah sehingga mahasiswa dapat belajar sesuai dengan apa yang dipelajari di mata kuliah tersebut
4. Mahasiswa dapat menemukan cara belajar dengan di tempatkannya model
5. Mahasiswa dapat menambah materi apabila Mahasiswa lebih banyak melakukan latihan
6. **Saran**

Aplikasi *E-Learning* yang telah dibuat ini bisa dikembangkan lebih lanjut. Berikut penulis memberikan beberapa saran yang dapat menjadi pertimbangan untuk pengembangan aplikasi, diantaranya:

1. Aplikasi *E-Learning* ini dapat dikembangkan dalam hal desain agar tampilan menjadi lebih menarik.
2. Aplikasi *E-Learning* ini dapat dikembangkan dengan menambah materi yang dibutuhkan mahasiswa dalam menjawab soal

157

1. Aplikasi parkir ini kedepannya dapat dikembangkan dengan cara menambah fitur untuk komunikasi mahasiswa dan dosen
2. Aplikasi ini akan dikembangkan dengan menambahkan fitur untuk Tempat menyimpan Tugas, *Quiz* dan materi dari Dosen

**DAFTAR PUSTAKA**

Deitel , 1992-2000. *Python How to Program ,Inggris* : *Deitel & Associates*

Amin, Muhammad Miftakhul, 2009, *Personalisasi Web*, Ilmukomputer.com

Pressman, R.S. 2002. Rekayasa Perangkat Lunak, Yogyakarta : Andi Offset

Effendi, E., dan Zhuang, H. (2005). *E-Learning*: Konsep dan aplikasi, Yogyakarta: Penerbit Andi.

Mason ,Robin., dan Rennie,Frank. (2009). *Elearning*. Penerbit Pustaka Baca,  
Yogyakarta

Setiyadi ,Alif Cahya dan Salim Mohammad Syam’un , (2013) , Pemerolehan Bahasa Kedua Menurut Stephen Krashen, Ponorogo : Pustaka Book *Publisher*

Edufiesta. 2008. Pengertian E-learning. Tersedia dalam : . Diakses pada tanggal 13 september 2015 Fathansyah, Ir. 1999. *Basis Data*. Bandung: Informatika Bandung.

Arbie, Manajemen Database dengan MySQL, Andi Yogyakarta, 2004.

Eko Budiarto , “E-learning” ,http://issuu.com/elitraharja/docs/e-learning