

LAPORAN UJIAN TENGAH SEMESTER BACK-END



WEBSITE UNIVERSITY MANAGEMENT DATABASE “UNDATABASE”

Hans Santoso – 535220129
Sabrina Phalosa Phai - 535220131
Lekrey Jerel Jacob Laipiopa – 535220134

**FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS TARUMANAGARA
TAHUN 2024**

DAFTAR ISI

	HALAMAN
HALAMAN SAMPUL	i
DAFTAR ISI	ii
BAB I PENDAHULUAN	1
BAB II METHODOLOGI	2
BAB III PROGRESS PROJECT	5
BAB IV KESIMPULAN	27
DAFTAR PUSTAKA	28

BAB 1

LATAR BELAKANG

UnDabase adalah sebuah inisiatif yang muncul dari kebutuhan yang semakin kompleks dalam mengelola administrasi mahasiswa di institusi pendidikan. Dalam era di mana teknologi informasi menjadi tulang punggung untuk efisiensi dan akurasi, sistem manual yang masih banyak digunakan rentan terhadap kesalahan dan lambat dalam pemrosesan data. Institusi pendidikan, dengan struktur organisasi yang kompleks dan jumlah mahasiswa yang besar, membutuhkan solusi yang dapat mengatasi tantangan tersebut dengan efisien.

Kebutuhan akan manajemen data mahasiswa yang efektif sangat krusial. Informasi seperti Nomor Induk Mahasiswa (NIM), nama, fakultas, dan informasi akademik lainnya perlu diatur dengan tepat dan mudah diakses. Sistem manual seringkali mengalami kendala dalam pelacakan data secara menyeluruh, terutama ketika terjadi perubahan atau pembaruan informasi. Hal ini dapat menghambat proses administrasi, mengakibatkan kesalahan dalam pengelolaan data, dan berdampak negatif pada pengalaman mahasiswa.

UnDabase hadir sebagai solusi yang efisien dan efektif untuk mengatasi masalah tersebut. Dengan memanfaatkan kemajuan teknologi informasi, UnDabase tidak hanya menyediakan platform yang dapat mengelola informasi mahasiswa secara akurat, tetapi juga memberikan layanan yang lebih baik kepada pengguna. Aksesibilitas yang mudah, kecepatan dalam pemrosesan data, dan tingkat keamanan yang tinggi menjadi fokus utama dalam pengembangan UnDabase

Dengan demikian, UnDabase bukan hanya sekadar solusi teknologi informasi biasa, tetapi merupakan sebuah inovasi yang dapat mengubah cara institusi pendidikan mengelola informasi mahasiswa, memberikan pengalaman pengguna yang lebih baik, dan mendukung pengambilan keputusan yang lebih tepat dan terinformasi.

BAB 2

METHODOLOGI

2.1 Penjelasan Methodology

Metodologi pengembangan yang digunakan dalam proyek UnDabase adalah Agile Scrum. Metodologi ini dipilih karena keunggulannya dalam menghadapi perubahan yang cepat, fleksibilitas, dan fokus pada hasil yang dapat disesuaikan dengan kebutuhan pengguna [1].

a. Pengertian Agile Scrum

Agile Scrum adalah pendekatan pengembangan perangkat lunak yang berbasis pada kerja kolaboratif tim, iterasi singkat, dan responsif terhadap perubahan [2]. Dalam konteks proyek UnDabase, Agile Scrum digunakan untuk memastikan kelancaran pengembangan sistem yang adaptif terhadap perubahan kebutuhan dan kondisi.

b. Struktur Tim

Tim pengembangan UnDabase terdiri dari beberapa peran kunci dalam kerangka Agile Scrum:

- Product Owner: Bertanggung jawab atas kebutuhan dan keinginan pengguna serta memprioritaskan backlog.
- Scrum Master: Mengelola proses Scrum, menghilangkan hambatan, dan memastikan tim berjalan sesuai standar Scrum.
- Tim Pengembangan: Anggota tim yang secara aktif terlibat dalam pengembangan, pengujian, dan pengiriman fitur-fitur UnDabase [3].

c. Sprint Planning

Setiap iterasi pengembangan, disebut Sprint, direncanakan dalam Sprint Planning Meeting. Product Owner mengidentifikasi dan menguraikan item backlog yang akan dikerjakan dalam sprint, sementara tim pengembangan mengestimasi kompleksitas dan mengkomitmenkan diri untuk menyelesaikan pekerjaan tersebut [4].

d. Daily Standup

Setiap hari, tim mengadakan pertemuan singkat yang disebut Daily Standup untuk membagikan update, mengidentifikasi hambatan, dan menyesuaikan rencana kerja agar sesuai dengan tujuan sprint.

e. Sprint Review dan Retrospective

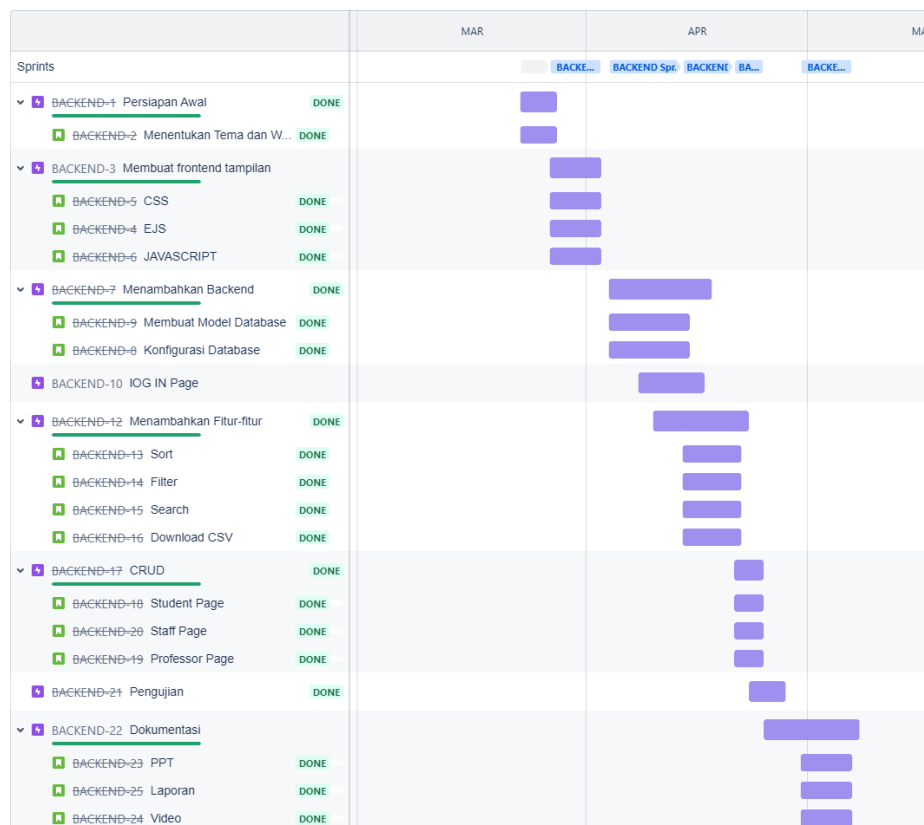
Setelah selesai sprint, tim melakukan Sprint Review untuk meninjau hasil yang telah dicapai dan mendapatkan umpan balik dari Product Owner dan pengguna. Selanjutnya, Sprint Retrospective digunakan untuk mengevaluasi proses pengembangan, mengidentifikasi pembelajaran, dan merencanakan perbaikan untuk sprint berikutnya.

f. Product Backlog dan Increment

Product Backlog terus diperbarui dengan kebutuhan baru dan perubahan prioritas, sedangkan setiap sprint menghasilkan Increment yang merupakan penambahan fungsional atau perbaikan yang dapat digunakan secara langsung oleh pengguna.

Metode Agile Scrum yang digunakan dalam pengembangan UnDatabase memberikan kerangka kerja yang terstruktur, adaptif, dan fokus pada pengiriman nilai ke pengguna secara berkala dan terukur. Dengan pendekatan ini, diharapkan UnDatabase dapat terus berkembang sesuai dengan kebutuhan dan ekspektasi pengguna.

2.2. Timeline Project



Gambar 1. Timeline Project

a. Sprint 1: Persiapan Awal (23 Maret 2024 - 27 Maret 2024)

Tujuan:

- Menentukan tema dan konsep website UnDabase.

b. Sprint 2: Membuat Frontend Penampilan (27 Maret 2024 - 2 April 2024)

Tujuan:

- Membuat CSS untuk tata letak dan tampilan visual.
- Mengembangkan halaman menggunakan EJS.
- Implementasi fitur interaktif dengan JavaScript.

c. Sprint 3: Menambahkan Backend (4 April 2024 - 17 April 2024)

Tujuan:

- Membuat model database.
- Konfigurasi koneksi database.

d. Sprint 4: Halaman Log In (8 April 2024 - 18 April 2024)

Tujuan:

- Mengembangkan halaman log masuk untuk pengguna.

e. Sprint 5: Membangun Fitur-Fitur (10 April 2024 - 22 April 2024)

Tujuan:

- Fitur sort, filter, search, dan download.

f. Sprint 6: CRUD Operations (21 April 2024 - 24 April 2024)

Tujuan:

- Implementasi operasi CRUD untuk pengelolaan data.

g. Sprint 7: Pengujian (23 April 2024 - 27 April 2024)

Tujuan:

- Melakukan pengujian menyeluruh.

h. Sprint 8: Dokumentasi (27 April 2024 - 6 Mei 2024)

Tujuan:

- Pembuatan presentasi (PPT).
- Penulisan laporan.
- Pembuatan video demonstrasi.

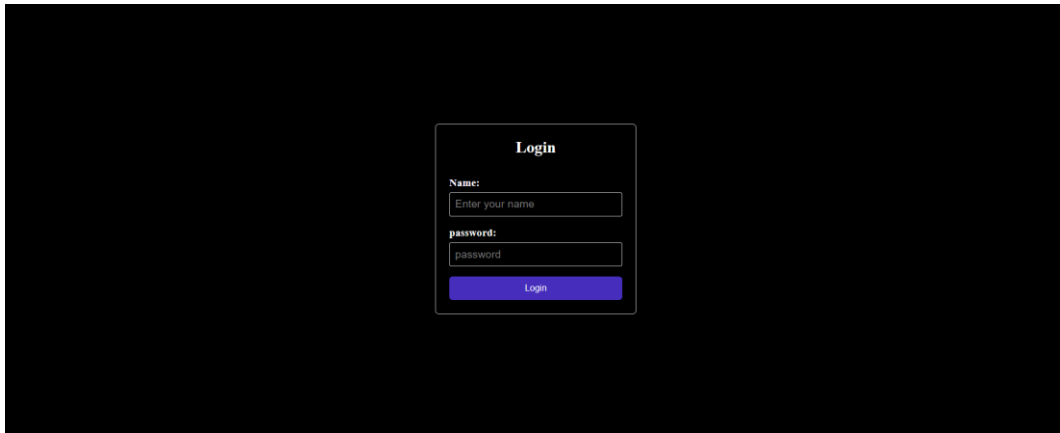
.BAB 3

PROGRESS PROJECT

3.1 Kondisi Terakhir Project Akhir

3.1.1 Log In Page

a. Tampilan :



Gambar 2. Tampilan Log In

Pada halaman login yang berada di (/login) berguna untuk menfilter agar hanya admin yang dapat memasuki web , user dapat memasukan Name dan Password untuk login.

b. Kode login.ejs :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>login</title>

  <link rel="stylesheet" href="log.css">
</head>
<body>

  <div class="form-container">
    <h2>Login</h2>
    <form action="/login" method="post">
      <div class="form-group">
        <label for="name">Name:</label>
        <input type="text" id="name" name="username" placeholder="Enter your name" required autocomplete="off">
      </div>
      <div class="form-group">
        <label for="password">password:</label>
        <input type="password" id="password" name="password" placeholder="password" required>
      </div>
      <button type="submit" class="submit-btn">Login</button>
    </form>
  </div>

</body>
</html>
```

Gambar 3. kode login.ejs

Ini merupakan file ejs yang memuat frontend untuk halaman login , hanya terdapat tag <form> yang berisi 2 <input> dengan <label> di masing masing inputannya , yang pertama untuk menginput Name dan yang kedua untuk Password , lalu di paling bawah terdapat <button> dengan type bervalue submit

c. Kode app.js

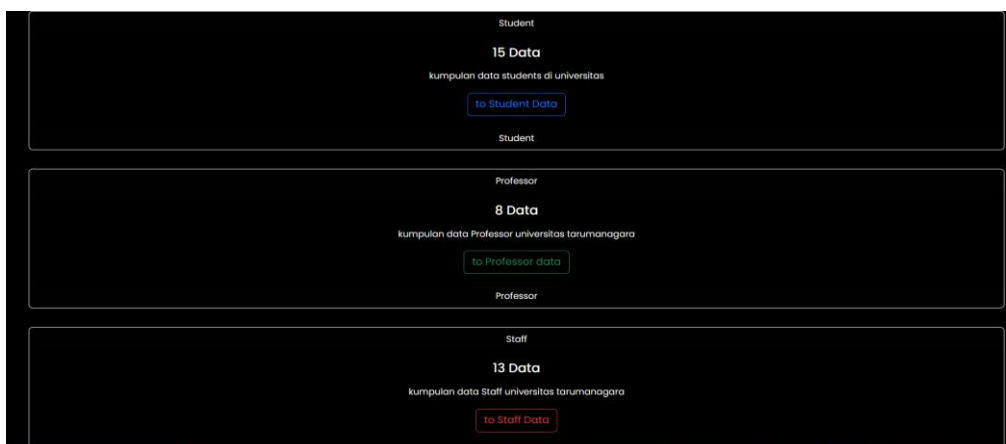
```
// Login user
app.post("/login", async (req, res) => {
  try {
    const check = await collection.findOne({ name: req.body.username });
    if (!check) {
      res.send("User name cannot found")
    }
    // Compare the hashed password from the database with the plaintext password
    const isPasswordMatch = await bcrypt.compare(req.body.password, check.password);
    if (!isPasswordMatch) {
      res.send("wrong Password");
    }
    else {
      res.redirect("/");
    }
  }
  catch {
    res.send("wrong Details");
  }
});
```

Gambar 4. Kode app.js

Ini adalah fungsi yang mengatur alur login pada halaman masuk. Proses dimulai dengan penanganan kesalahan (try and catch) yang disusul dengan pengambilan nama pengguna (username) dan kata sandi (password) yang tersimpan dalam basis data menggunakan metode findOne. Karena akses ini terbatas hanya untuk admin, kami telah memastikan bahwa hanya ada satu pasangan nama pengguna (username) dan kata sandi (password) yang valid. Untuk memverifikasi kata sandi yang dimasukkan oleh pengguna, kami menggunakan modul bcrypt dengan fungsi compare. Apabila pengguna salah memasukkan nama pengguna, mereka akan diarahkan ke halaman yang menampilkan pesan "User name cannot be found". Sedangkan jika kesalahan terjadi karena kata sandi yang salah, pengguna akan diarahkan ke halaman yang menampilkan pesan "Wrong password". Namun, jika seluruh proses berhasil diverifikasi, pengguna akan diarahkan secara langsung ke halaman utama ("/").

3.1.2 Dashboard

a. Tampilan :



Gambar 5. Tampilan Dashboard

Pada halaman dashboard (/Dashboard), berguna untuk navigasi ke berbagai halaman lainnya dan juga kesimpulan atau ringkasan dari web UnDaBase ini.

b. Kode Dashboard.ejs

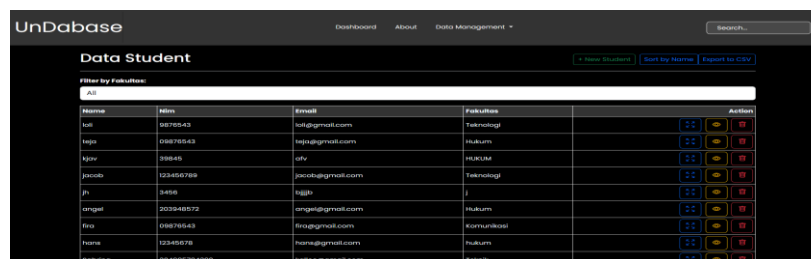
```
1 <div class="card text-center" style="background-color: black; border: 1px solid white; margin-top: 30px;" >
2 <div class="card-header" style="color: white;">Student</div>
3 <div class="card-body">
4 <h5 class="card-title" style="color: white;">15 Data</h5>
5 <p class="card-text" style="color: white;">
6 kumpulan data students di universitas
7 </p>
8 <a href="/" class="btn btn-outline-primary">to Student Data</a>
9 </div>
10 <div class="card-footer " style="color: white;" >Student</div>
11 </div>
12
13
14 <div class="card text-center" style="background-color: black; border: 1px solid white; margin-top: 30px;" >
15 <div class="card-header" style="color: white;">Professor</div>
16 <div class="card-body">
17 <h5 class="card-title" style="color: white;">8 Data</h5>
18 <p class="card-text" style="color: white;">
19 kumpulan data Professor universitas tarumanagara
20 </p>
21 <a href="/dosen" class="btn btn-outline-success">to Professor data</a>
22 </div>
23 <div class="card-footer " style="color: white;" >Professor</div>
24 </div>
25
26
27
28
29 <div class="card text-center" style="background-color: black; border: 1px solid white; margin-top: 30px;" >
30 <div class="card-header" style="color: white;">Staff</div>
31 <div class="card-body">
32 <h5 class="card-title" style="color: white;">13 Data</h5>
33 <p class="card-text" style="color: white;">
34 kumpulan data Staff universitas tarumanagara
35 </p>
36 <a href="/staff" class="btn btn-outline-danger">to Staff Data</a>
37 </div>
38 <div class="card-footer " style="color: white;" >Staff</div>
39 </div>
40 </div>
```

Gambar 6. Kode Dashboard.ejs

File Dashboard.ejs merupakan template yang digunakan untuk membuat tiga card yang berbeda, yang masing-masing merepresentasikan CRUD (Create, Read, Update, Delete) untuk data mahasiswa (students), profesor, dan staf. Setiap card terdiri dari struktur header, body, dan footer. Bagian header digunakan untuk menampilkan judul card, sedangkan bagian body berisi heading dengan tag `<h5>` dan deskripsi dengan tag `<p>`. Di bagian bawah body, terdapat tombol yang mengarahkan pengguna ke halaman yang ditentukan. Bagian footer menampilkan judul yang sama seperti di bagian header. Karena ada tiga jenis CRUD, struktur card ini diulang sebanyak tiga kali untuk menghasilkan tiga card yang berbeda, masing-masing dengan navigasi ke halaman yang sesuai untuk operasi CRUD terkait.

3.1.3 Home page

a. Tampilan :



ID	Name	Email	Password	Action
1	1901	1901@gmail.com	1901	
2	1902	1902@gmail.com	1902	
3	1903	1903@gmail.com	1903	
4	1904	1904@gmail.com	1904	
5	1905	1905@gmail.com	1905	
6	1906	1906@gmail.com	1906	
7	1907	1907@gmail.com	1907	
8	1908	1908@gmail.com	1908	
9	1909	1909@gmail.com	1909	
10	1910	1910@gmail.com	1910	

Gambar 7. Tampilan Home Page

Halaman utama (home page) adalah titik awal dari aplikasi yang menampilkan data yang sudah tersimpan dalam database. Halaman ini dilengkapi dengan fitur-fitur untuk mengelola data, antara lain sorting (pengurutan), export ke format CSV, dan filter. Pengguna dapat menggunakan tombol-tombol yang tersedia untuk melakukan navigasi ke halaman-halaman khusus yang memungkinkan proses CRUD (Create, Read, Update, Delete), berikut tampilannya :

b. Kode Index.ejs :

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Data Student</title>
7 <!-- Tambahkan CSS Bootstrap (contoh menggunakan CDN) -->
8 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
9 </head>
10 <body>
11
12
13
14 <div class="container">
15 <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">
16 <h1 class="h2">Data Student</h1>
17 <div class="btn-toolbar mb-2 mb-md-0">
18 <div class="btn-group me-2">
19 <a href="#" class="btn btn-sm btn-outline-success">New Student</a>
20 </div>
21 <!-- Tambahkan tombol untuk pengurutan -->
22 <button onclick="sortTable('Name')" class="btn btn-sm btn-outline-primary">Sort by Name</button>
23 <!-- Tambahkan tombol untuk ekspor data -->
24 <button onclick="exportToCSV('studentTable')" class="btn btn-sm btn-outline-primary">Export to CSV</button>
25 </div>
26 </div>
27
28 <!-- Filter fakultas -->
29 <div class="form-group">
30 <label for="fakultasFilter">Filter by Fakultas:</label>
31 <select class="form-control" id="fakultasFilter" onchange="filterByFakultas()">
32 <option value="">All</option>
33 <!-- Daftar fakultas disini -->
34 <option value="Teknologi">Teknologi</option>
35 <option value="Hukum">Hukum</option>
36 <option value="Teknik">Teknik</option>
37 <option value="Komunikasi">Komunikasi</option>
38 </select>
39 </div>
```

Gambar 8. Kode Index.ejs

Dalam halaman ini, pengguna disajikan dengan beberapa fitur yang memungkinkan mereka untuk berinteraksi dengan data mahasiswa dengan lebih efisien. Pertama, terdapat tombol "New Student" yang mengarahkan pengguna ke halaman tambah mahasiswa, memudahkan dalam proses penambahan data baru ke dalam sistem. Selain itu, terdapat tombol "Sort by Name" yang memungkinkan pengguna untuk mengurutkan data mahasiswa berdasarkan nama. Tombol "Export to CSV" memberikan pengguna kemampuan untuk mengunduh data mahasiswa dalam format CSV. Terakhir, form filter fakultas memungkinkan pengguna untuk menyaring data mahasiswa berdasarkan fakultas tertentu, sehingga mereka dapat dengan cepat menemukan data yang relevan.

```
1 <!-- Pesan -->
2 <{{ messages.forEach(element => { %
3 <div class="alert alert-success alert-dismissible fade show" role="alert">
4 <{{ element %
5 <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
6 <{{ % % %
7 <}} %
8
9 <!-- Tabel data student -->
10 <div class="table-responsive bg">
11 <table id="studentTable" class="table table-striped table-sm">
12 <thead>
13 <tr>
14 <th scope="col" style="background-color: #333; color: aliceblue; border: 1px white solid;">Name</th>
15 <th scope="col" style="background-color: #333; color: aliceblue; border: 1px white solid;">Nim</th>
16 <th scope="col" style="background-color: #333; color: aliceblue; border: 1px white solid;">Email</th>
17 <th scope="col" style="background-color: #333; color: aliceblue; border: 1px white solid;">Fakultas</th>
18 <th scope="col" class="text-end" style="background-color: #333; color: aliceblue; border: 1px white solid;">Action</th>
19 </tr>
20 </thead>
21 <tbody>
22 <!-- Data student akan dimasukkan di sini -->
23 <{{ customers.forEach(element => { %
24 <tr class="align-middle">
25 <td style="background-color: black; color: aliceblue; border: 1px white solid;"><{{ element.Name %</td>
26 <td style="background-color: black; color: aliceblue; border: 1px white solid;"><{{ element.Nim %</td>
27 <td style="background-color: black; color: aliceblue; border: 1px white solid;"><{{ element.Email %</td>
28 <td style="background-color: black; color: aliceblue; border: 1px white solid;"><{{ element.Fakultas %</td>
29 <td class="text-end" style="background-color: black; color: aliceblue; border: 1px white solid;">
30 <div class="d-flex flex-row-reverse justify-content-end gap-2">
31 <a href="#" class="btn btn-sm btn-outline-primary" type="button">Edit</a>
32 <a href="#" class="btn btn-sm btn-outline-warning" type="button">Delete</a>
33 <a href="#" class="btn btn-sm btn-outline-danger" type="button">Submit</a>
34 </div>
35 </td>
36 </tr>
37 </tbody>
38 <}} %
39 </table>
40 </div>
```

Gambar 9. Kode Index.ejs

Pertama, bagian untuk menampilkan pesan-pesan (messages) yang mungkin dihasilkan oleh sistem, seperti pesan sukses atau pesan kesalahan. Pesan-pesan tersebut diambil dari array messages dan ditampilkan satu per satu menggunakan loop forEach.

Kedua, terdapat tabel yang menampilkan data mahasiswa (students) dalam bentuk yang terstruktur.

- (`<tr>`) menunjukkan satu entri data mahasiswa dan setiap kolom tabel
- (`<td>`) menampilkan atribut-atribut dari setiap mahasiswa, seperti nama, NIM, email, dan fakultas. Data mahasiswa ini diambil dari array `customers` dan ditampilkan menggunakan loop `forEach`.

Pada setiap baris data mahasiswa, terdapat juga kolom "Action" yang berisi tautan untuk melihat detail Setiap tautan dan tombol ini akan mengarahkan pengguna ke halaman atau endpoint yang sesuai untuk menjalankan operasi CRUD yang diperlukan terhadap data mahasiswa



```
1      <% if (customers.length > 0) { %>
2      <!-- Pagination -->
3      <nav aria-label="Dashboard Pagination">
4        <ul class="pagination justify-content-center mt-5 c">
5          <% if (current == 1) { %>
6            <li class="page-item disabled"><a href="#" class="page-link custom-page">First</a></li>
7          <% } else { %>
8            <li class="page-item"><a href="/?page=1" class="page-link custom-page">First</a></li>
9          <% } %>
10         <% var i = (Number(current) > 5 ? Number(current) - 4 : 1) %>
11         <% if (i !== 1) { %>
12           <li class="page-item disabled"><a href="#" class="page-link custom-page">...</a></li>
13         <% } %>
14         <% for(; i <= (Number(current) + 4) && i <= pages; i++) { %>
15           <% if (i == current) { %>
16             <li class="page-item disabled"><a href="#" class="page-link custom-page"><%= i %></a></li>
17           <% } else { %>
18             <li class="page-item"><a href="/?page=<%= i %>" class="page-link custom-page"><%= i %></a></li>
19           <% } %>
20         <% if (i == Number(current) + 4 && i < pages) { %>
21           <li class="page-item disabled"><a href="#" class="page-link custom-page">...</a></li>
22         <% } %>
23         <% if (current == pages) { %>
24           <li class="page-item disabled"><a href="#" class="page-link custom-page">Last</a></li>
25         <% } else { %>
26           <li class="page-item"><a href="/?page=<%= pages %>" class="page-link custom-page">Last</a></li>
27         <% } %>
28       </ul>
29     </nav>
30   <% } %>
31 </div>
```

Gambar 10. Kode Index.ejs

Potongan kode tersebut adalah implementasi fitur pagination pada halaman Homepage menggunakan bahasa templating EJS. Fitur ini membagi data mahasiswa ke dalam beberapa halaman agar lebih mudah dinavigasi oleh pengguna.

1. Pengecekan Data : Dilakukan pengecekan apakah terdapat data mahasiswa yang akan ditampilkan menggunakan tag pembukaan ``<% if (customers.length > 0) { %>``.
2. Navigasi Pagination : Tombol navigasi pagination, seperti "First" dan "Last", ditampilkan di tengah halaman untuk memungkinkan pengguna melompat ke halaman pertama atau terakhir. Ini diimplementasikan dalam sebuah tag ``<nav>`` dengan atribut `aria-label="Dashboard Pagination"`.
3. Nomor Halaman : Nomor halaman yang tersedia ditampilkan dalam loop menggunakan tag pembukaan ``<% for(...) { %>``. Setiap nomor halaman diwakili oleh sebuah tag ```` yang berisi sebuah tautan dengan atribut `href` yang mengarahkan pengguna ke halaman yang sesuai.
4. Tanda Elipsis : Jika terdapat lebih dari lima halaman, tanda elipsis ditambahkan untuk menunjukkan adanya halaman lain di antara halaman yang ditampilkan. Ini diimplementasikan menggunakan tag pembukaan ``<% if(...) { %>``.

3.1.4 Fitur Sort



Gambar 11. Fitur Sort

Potongan kode JavaScript tersebut menggambarkan fungsi `sortTable()` yang menerapkan algoritma pengurutan bubble sort pada tabel:

1. Seleksi Elemen Tabel : Fungsi memilih tabel yang akan diurutkan dengan menggunakan metode `document.getElementById("studentTable")`.
2. Looping untuk Pengurutan : Dilakukan perulangan menggunakan loop `while` untuk mengecek dan melakukan pengurutan terhadap pasangan baris dalam tabel.
3. Perulangan Baris Tabel : Setiap baris tabel diakses melalui properti `rows` dari elemen tabel.
4. Pengecekan Kriteria Pengurutan : Nilai pada kolom yang ingin diurutkan diperoleh menggunakan metode `getElementsByTagName("TD")[0]`. Dalam kasus ini, nilai diambil dari kolom pertama (kolom nama).
5. Penentuan Perlunya Pertukaran : Dilakukan perbandingan nilai untuk menentukan apakah kedua baris perlu ditukar posisinya.
6. Pertukaran Posisi Baris : Jika kondisi pengurutan terpenuhi, kedua baris ditukar posisinya menggunakan metode `parentNode.insertBefore()`.
7. Update Tampilan : Setelah pertukaran posisi dilakukan, tabel diperbarui untuk mencerminkan urutan baru dari baris-barisnya.

3.1.5 Fitur Filter

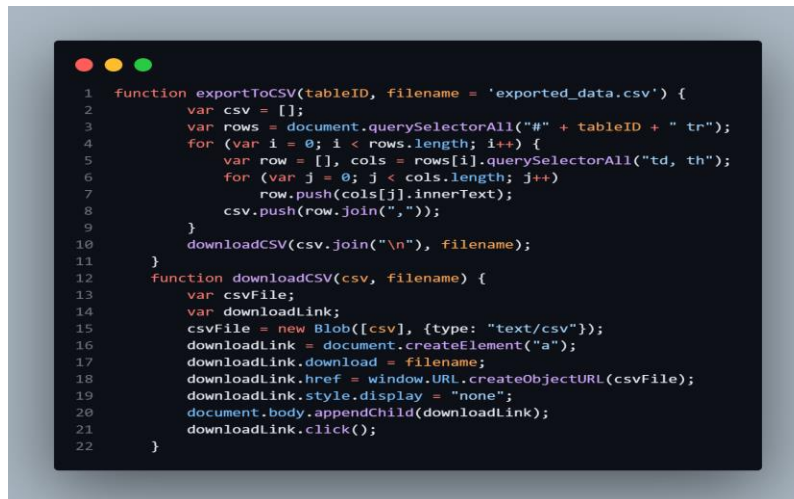


Gambar 12. Fitur Filter

Berikut adalah penjelasan lebih detail tentang cara kerja fungsi ini:

1. Seleksi Elemen : Fungsi ini pertama-tama mengambil elemen input yang berisi nilai filter berdasarkan fakultas yang dipilih oleh pengguna. Ini dilakukan dengan menggunakan `document.getElementById("fakultasFilter")`. Selain itu, fungsi juga mengambil referensi ke tabel yang akan difilter menggunakan `document.getElementById("studentTable")`.
2. Looping Melalui Baris Tabel : Dilakukan looping melalui setiap baris (`tr`) dalam tabel untuk memeriksa nilai pada kolom fakultas.
3. Pengecekan Nilai Kolom Fakultas : Untuk setiap baris, nilai pada kolom fakultas diperoleh dengan menggunakan `getElementsByTagName("td")[3]`, yang mengambil elemen pada indeks kolom keempat (indeks dimulai dari 0) dalam setiap baris. Nilai tersebut kemudian disimpan dalam variabel `txtValue`.
4. Penyesuaian Tampilan : Jika nilai dalam kolom fakultas cocok dengan nilai filter yang dipilih oleh pengguna atau jika filter kosong, maka baris tersebut tetap ditampilkan (`tr[i].style.display = ""`). Namun, jika nilai dalam kolom fakultas tidak cocok dengan filter yang dipilih, maka baris tersebut disembunyikan dari tampilan dengan mengatur `style.display`-nya menjadi `"none"`.

3.1.6. Fitur Export and download to csv



Gambar 13. Kode filter Export and download to csv

Berikut penjelasan lebih rinci tentang kedua fungsi tersebut:

1. Fungsi exportToCSV(tableID, filename) :

- Menerima dua parameter: `tableID` adalah ID dari tabel yang akan diekspor, dan `filename` adalah nama file CSV yang akan dihasilkan (dengan nilai default "exported_data.csv" jika tidak disediakan).
- Langkah pertama adalah inisialisasi variabel `csv` sebagai array kosong yang akan menyimpan data CSV.
- Selanjutnya, dilakukan iterasi melalui setiap baris tabel menggunakan loop `for`. Untuk setiap baris, nilai dari setiap sel (`cols`) diambil dan dimasukkan ke dalam array `row`.
- Array `row` yang berisi nilai dari setiap sel dalam baris tersebut kemudian digabungkan menjadi satu string dengan menggunakan tanda koma sebagai pemisah, dan hasilnya dimasukkan ke dalam array `csv`.
- Setelah semua baris diproses, data CSV yang sudah dihasilkan disampaikan ke fungsi `downloadCSV()` untuk diunduh.

2. Fungsi downloadCSV(csv, filename) :

- Menerima dua parameter: `csv` adalah data CSV yang akan diunduh, dan `filename` adalah nama file untuk file CSV yang dihasilkan.
- Pertama-tama, file CSV dibuat dalam bentuk objek `Blob` menggunakan data yang diberikan.
- Selanjutnya, sebuah elemen `` (link) dibuat untuk mengunduh file CSV.
- Nama file yang diunduh diatur sesuai dengan nilai parameter `filename`.
- Link unduh kemudian disimpan dalam atribut `href` dari elemen ``, menggunakan URL objek `Blob` yang telah dibuat sebelumnya.
- Untuk menghindari link unduh muncul secara visual, elemen `` tersebut disembunyikan dengan mengatur `display`-nya menjadi `"none"`.
- Setelah itu, elemen `` ditambahkan ke dalam DOM, dan link unduhnya diaktifkan dengan memanggil metode `click()` pada elemen tersebut.

3.1.7 Controllers

a. kode Studentcontrollers.js



```
1 exports.homepage = async (req, res) => {
2   const messages = await req.flash("info");
3   const locals = {
4     title: " Student Database",
5     description: "Student Management System",
6   };
7   let perPage = 12;
8   let page = req.query.page || 1;
9   try {
10    const customers = await Customer.aggregate([{$sort: { createdAt: -1 } }])
11      .skip(perPage * page - perPage)
12      .limit(perPage)
13      .exec();
14    const count = await Customer.countDocuments({});
15    res.render("index", {
16      locals,
17      customers,
18      current: page,
19      pages: Math.ceil(count / perPage),
20      messages,
21    });
22  } catch (error) {
23    console.log(error);
24  }
25 };
26
```

Gambar 14. Kode studentcontroller.js

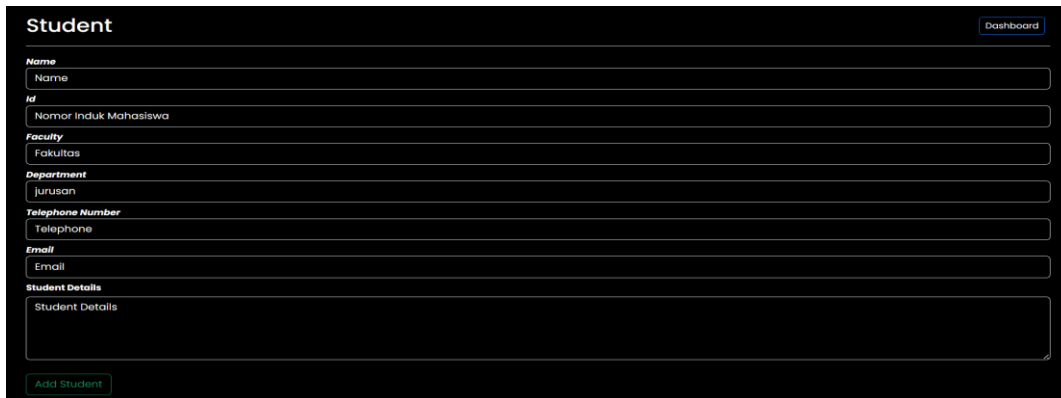
Fungsi ini bertujuan untuk menampilkan halaman utama (homepage) dari sistem tersebut. Berikut adalah penjelasan singkat tentang bagaimana fungsi ini bekerja:

- Saat fungsi ini dipanggil, pesan-pesan informasi (info messages) dari sesi flash (flash session) diambil menggunakan metode `req.flash("info")`.
- Variabel `perPage` digunakan untuk menentukan jumlah data yang akan ditampilkan per halaman.
- Nomor halaman saat ini (`page`) diambil dari query string jika tersedia, jika tidak maka akan default ke halaman pertama.
- Kemudian, data mahasiswa (`customers`) diambil dari database dengan menggunakan metode agregasi MongoDB. Data ini diurutkan berdasarkan waktu pembuatan (`createdAt`) secara menurun dan hanya mengambil data sesuai dengan halaman saat ini dan jumlah data per halaman yang ditentukan.
- Jumlah total data mahasiswa (`count`) dihitung untuk menghitung jumlah halaman yang diperlukan.
- Selanjutnya, halaman "index" dirender menggunakan template engine (dalam contoh ini, digunakan template engine pug) dengan menyertakan data lokal (`locals`), data mahasiswa (`customers`), nomor halaman saat ini (`current`), jumlah halaman total (`pages`), dan pesan-pesan informasi (`messages`).
- Jika terjadi kesalahan selama proses ini, kesalahan tersebut akan ditangkap dan dicetak ke konsol.

Dengan demikian, fungsi ini bertanggung jawab untuk menyiapkan dan merender halaman utama dengan data mahasiswa yang sesuai

3.1.8 Fitur Add

a. Tampilan fitur add :



Gambar 15. Fitur add

Halaman add di (“/add”) berguna untuk melakukan create data ke sistem dan , atau tempat user dapat memasukan student baru ke sistem , berikut tampilannya :

b. Kode add.ejs

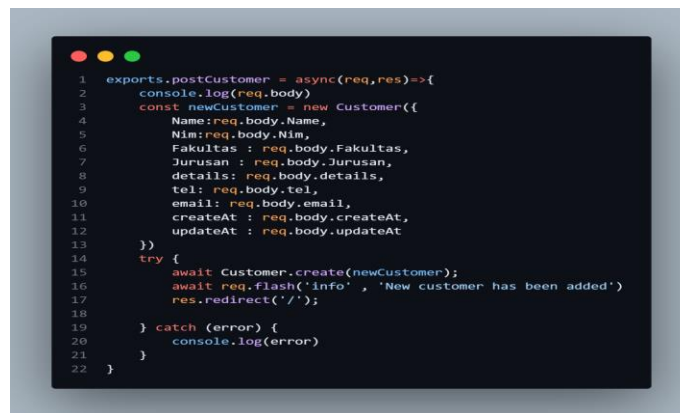
```
1 <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">
2   <h1 class="h2">Student</h1>
3   <div class="btn-toolbar mb-2 mb-md 0">
4     <div class="btn-group me-2">
5       <button class="btn btn-sm btn-outline-primary"><a href="/">Dashboard</a></button>
6     </div>
7   </div>
8 </div>
9 <form action="/add" method="POST">
10  <div class="col">
11    <label for="Name">Name</label>
12    <input type="text" class="form-control custom-input" id="Name" name="Name" value="" placeholder="Name" required>
13  </div>
14  <div class="col">
15    <label for="Nim">Id</label>
16    <input type="text" class="form-control custom-input" id="Nim" name="Nim" value="" placeholder="Nomor Induk Mahasiswa" required>
17  </div>
18  <div class="col">
19    <label for="Fakultas">Faculty</label>
20    <input type="text" class="form-control custom-input" id="Fakultas" name="Fakultas" value="" placeholder="Fakultas" required>
21  </div>
22  <div class="col">
23    <label for="Jurusan">Department</label>
24    <input type="text" class="form-control custom-input" id="Jurusan" name="Jurusan" value="" placeholder="Jurusan" required>
25  </div>
26  <div class="col">
27    <label for="tel">Telephone Number</label>
28    <input type="text" class="form-control custom-input" id="tel" name="tel" value="" placeholder="Telephone" required>
29  </div>
30  <div class="col">
31    <label for="email">Email</label>
32    <input type="text" class="form-control custom-input" id="email" name="email" value="" placeholder="Email" required>
33  </div>
34  <div class="form-group mb-4">
35    <label for="details">Student Details</label>
36    <textarea class="form-control custom-input" name="details" id="details" cols="30" rows="4" placeholder="Student Details"></textarea>
37  </div>
38  <div class="form-group mb-4">
39    <button type="submit" class="btn btn-outline-success">Add Student</button>
40  </div>
41 </form>
42
```

Gambar 16. Kode add.ejs

Formulir di atas digunakan untuk menambahkan data mahasiswa ke dalam sistem. Formulir ini memuat beberapa < input > fields yang harus diisi oleh pengguna sebelum data dikirimkan. Setiap input field dilengkapi dengan label yang sesuai untuk memudahkan pengguna memahami maksud dari setiap field. < Input > fields tersebut mencakup

- Nama Mahasiswa,
- Nomor Induk Mahasiswa (NIM),
- Fakultas, Jurusan
- Nomor Telepon
- Alamat Email
- Detail Mahasiswa

Pengguna diminta untuk mengisi semua informasi yang diperlukan sebelum menekan tombol "Add Student" untuk mengirimkan data formulir. Tombol tersebut bertujuan untuk mengirimkan data ke server menggunakan metode `POST` agar dapat diproses oleh backend.



```
1 exports.postCustomer = async(req,res)=>{
2   console.log(req.body)
3   const newCustomer = new Customer({
4     Name:req.body.Name,
5     Nim:req.body.Nim,
6     Fakultas : req.body.Fakultas,
7     Jurusan : req.body.Jurusan,
8     details: req.body.details,
9     tel: req.body.tel,
10    email: req.body.email,
11    createAt : req.body.createAt,
12    updateAt : req.body.updateAt
13  })
14  try {
15    await Customer.create(newCustomer);
16    await req.flash('info' , 'New customer has been added')
17    res.redirect('/');
18  }
19  catch (error) {
20    console.log(error)
21  }
22 }
```

Gambar 17. Studentcontrollers.js

Di Kode ini hanya berguna untuk membuat model database lalu memasukan semua nilai yang di input ke file add.ejs , setelah itu di create modelnya dibagian try handler dan jika berhasil akan di redirect ke halaman ("/") jika gagal akan menprint error



```
1 exports.addCustomer = async(req,res)=>{
2   const locals = {
3     title: 'add New Student',
4     description : "Management System"
5   }
6   res.render('customer/add' , locals);
7 }
```

Gambar 18. add.ejs

Kode ini hanya untuk merender tampilan yang berada di file ("add.ejs") , dan juga ada variable locals yang berguna untuk membuat title dan description

3.1.9 Fitur View

a. Tampilan :

Halaman "view" bertujuan untuk menampilkan informasi lengkap tentang seorang mahasiswa yang dipilih dari database atau informasi lain yang tidak ditampilkan dalam bentuk tabel. Berikut adalah tampilan nya :

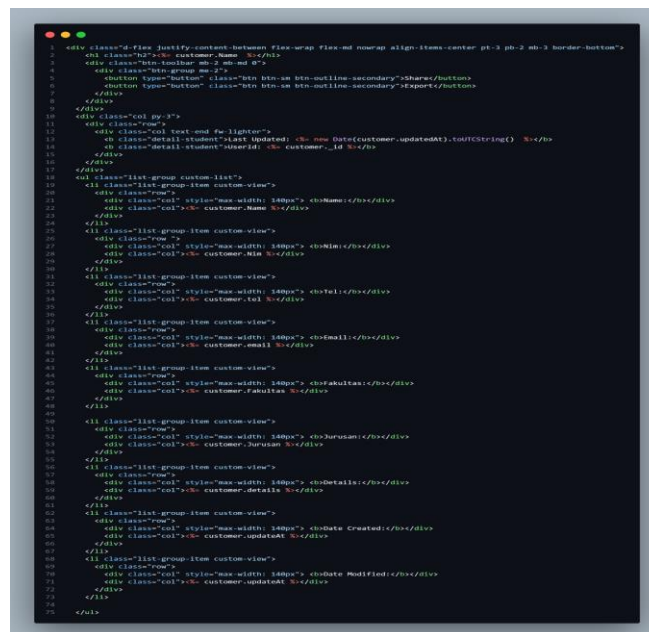


The screenshot shows a web application interface for a student profile. At the top, the name 'jacob' is displayed. Below it, a table lists various details: NIM (123456789), Tel (0123456789012345678901), Email (jacob@gmail.com), Fakultas (Teknologi), Jurusan (IT), Details (afad), Date Created (Tue Apr 16 2024 07:38:19 GMT+0700 (Western Indonesia Time)), and Date Modified (Tue Apr 16 2024 07:38:19 GMT+0700 (Western Indonesia Time)).

Name:	jacob
Nim:	123456789
Tel:	0123456789012345678901
Email:	jacob@gmail.com
Fakultas:	Teknologi
Jurusan:	IT
Details:	afad
Date Created:	Tue Apr 16 2024 07:38:19 GMT+0700 (Western Indonesia Time)
Date Modified:	Tue Apr 16 2024 07:38:19 GMT+0700 (Western Indonesia Time)

Gambar 19. Tampilan Fitur View

b. Kode view.ejs :



```
1 <!-- Define justify-content-between flex-wrap flex-md nowrap align-items-center pt-3 pb-2 mb-3 border-bottom -->
2 <div class="d-flex justify-content-between flex-wrap flex-md nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">
3   <div class="h2"><%= customer.Name %></div>
4   <div class="btn btn-outline-secondary">Share</div>
5   <div class="btn btn-outline-secondary">Export</div>
6   <div class="btn btn-outline-secondary">Update</div>
7   <div class="btn btn-outline-secondary">Delete</div>
8 </div>
9
10 <div class="col py-3">
11   <div class="row">
12     <div class="col text-end fw-light">
13       <div class="detail-student">Last Updated: <%= new Date(customer.updatedAt).toLocaleString() %></div>
14       <div class="detail-student">User ID: <%= customer._id %></div>
15     </div>
16   </div>
17
18   <div class="list-group custom-list">
19     <div class="list-group-item custom-view">
20       <div class="row">
21         <div class="col" style="max-width: 180px;"><%= customer.Name %></div>
22       </div>
23     </div>
24     <div class="list-group-item custom-view">
25       <div class="row">
26         <div class="col" style="max-width: 180px;"><%= customer.Nim %></div>
27         <div class="col" style="max-width: 180px;"><%= customer.Tel %></div>
28       </div>
29     </div>
30     <div class="list-group-item custom-view">
31       <div class="row">
32         <div class="col" style="max-width: 180px;"><%= customer.Email %></div>
33         <div class="col" style="max-width: 180px;"><%= customer.Fakultas %></div>
34       </div>
35     </div>
36     <div class="list-group-item custom-view">
37       <div class="row">
38         <div class="col" style="max-width: 180px;"><%= customer.Jurusan %></div>
39         <div class="col" style="max-width: 180px;"><%= customer.Details %></div>
40       </div>
41     </div>
42     <div class="list-group-item custom-view">
43       <div class="row">
44         <div class="col" style="max-width: 180px;"><%= customer.Date Created %></div>
45         <div class="col" style="max-width: 180px;"><%= customer.Date Modified %></div>
46       </div>
47     </div>
48   </div>
49 </div>
```

Gambar 20. Kode view.ejs

terdapat sebuah bagian header yang terdiri dari judul berupa nama pelanggan yang diperoleh dari variabel `customer.Name`. Selanjutnya, terdapat bagian informasi detail pelanggan. Informasi ini terdiri dari beberapa item yang disajikan dalam bentuk daftar. Setiap item dalam daftar ini menampilkan pasangan label dan nilai yang merepresentasikan atribut-atribut dari objek pelanggan.

Dua item terakhir menampilkan tanggal pembuatan (Date Created) dan tanggal terakhir modifikasi (Date Modified) dari data pelanggan. Selain itu, pada bagian "Last Updated" yang terletak di atas

Di page edit ini berguna untuk merubah semua data student yang sudah ada , lalu admin juga dapat menghapus data mahasiswa yang dipilih lewat tampilan edit , tampilannya sebagai berikut :

b. Kode edit.ejs :

```
1 <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">
2   <div class="h2"> Editing: <%= customer.Name %>/div>
3   <div class="btn btn-outline-primary" data-bbox="488 141 508 151">
4     <div class="btn-group" data-bbox="488 151 508 161">
5       <div class="btn btn-outline-primary" data-bbox="488 161 508 171">
6         <div class="btn btn-outline-primary" data-bbox="488 171 508 181">
7           <div class="btn btn-outline-primary" data-bbox="488 181 508 191">
8             <div class="btn btn-outline-primary" data-bbox="488 191 508 201">
9               <div class="btn btn-outline-primary" data-bbox="488 201 508 211">
10                <div class="btn btn-outline-primary" data-bbox="488 211 508 221">
11                <div class="btn btn-outline-primary" data-bbox="488 221 508 231">
12                <div class="btn btn-outline-primary" data-bbox="488 231 508 241">
13                <div class="btn btn-outline-primary" data-bbox="488 241 508 251">
14                <div class="btn btn-outline-primary" data-bbox="488 251 508 261">
15                <div class="btn btn-outline-primary" data-bbox="488 261 508 271">
16                <div class="btn btn-outline-primary" data-bbox="488 271 508 281">
17                <div class="btn btn-outline-primary" data-bbox="488 281 508 291">
18                <div class="btn btn-outline-primary" data-bbox="488 291 508 301">
19                <div class="btn btn-outline-primary" data-bbox="488 301 508 311">
20                <div class="btn btn-outline-primary" data-bbox="488 311 508 321">
21                <div class="btn btn-outline-primary" data-bbox="488 321 508 331">
22                <div class="btn btn-outline-primary" data-bbox="488 331 508 341">
23                <div class="btn btn-outline-primary" data-bbox="488 341 508 351">
24                <div class="btn btn-outline-primary" data-bbox="488 351 508 361">
25                <div class="btn btn-outline-primary" data-bbox="488 361 508 371">
26                <div class="btn btn-outline-primary" data-bbox="488 371 508 381">
27                <div class="btn btn-outline-primary" data-bbox="488 381 508 391">
28                <div class="btn btn-outline-primary" data-bbox="488 391 508 401">
29                <div class="btn btn-outline-primary" data-bbox="488 401 508 411">
30                <div class="btn btn-outline-primary" data-bbox="488 411 508 421">
31                <div class="btn btn-outline-primary" data-bbox="488 421 508 431">
32                <div class="btn btn-outline-primary" data-bbox="488 431 508 441">
33                <div class="btn btn-outline-primary" data-bbox="488 441 508 451">
34                <div class="btn btn-outline-primary" data-bbox="488 451 508 461">
35                <div class="btn btn-outline-primary" data-bbox="488 461 508 471">
36                <div class="btn btn-outline-primary" data-bbox="488 471 508 481">
37                <div class="btn btn-outline-primary" data-bbox="488 481 508 491">
38                <div class="btn btn-outline-primary" data-bbox="488 491 508 501">
39                <div class="btn btn-outline-primary" data-bbox="488 501 508 511">
40                <div class="btn btn-outline-primary" data-bbox="488 511 508 521">
41                <div class="btn btn-outline-primary" data-bbox="488 521 508 531">
42                <div class="btn btn-outline-primary" data-bbox="488 531 508 541">
43                <div class="btn btn-outline-primary" data-bbox="488 541 508 551">
44                <div class="btn btn-outline-primary" data-bbox="488 551 508 561">
45                <div class="btn btn-outline-primary" data-bbox="488 561 508 571">
46                <div class="btn btn-outline-primary" data-bbox="488 571 508 581">
47                <div class="btn btn-outline-primary" data-bbox="488 581 508 591">
48                <div class="btn btn-outline-primary" data-bbox="488 591 508 601">
49                <div class="btn btn-outline-primary" data-bbox="488 601 508 611">
50                <div class="btn btn-outline-primary" data-bbox="488 611 508 621">
51                <div class="btn btn-outline-primary" data-bbox="488 621 508 631">
52                <div class="btn btn-outline-primary" data-bbox="488 631 508 641">
53                <div class="btn btn-outline-primary" data-bbox="488 641 508 651">
54                <div class="btn btn-outline-primary" data-bbox="488 651 508 661">
55                <div class="btn btn-outline-primary" data-bbox="488 661 508 671">
56                <div class="btn btn-outline-primary" data-bbox="488 671 508 681">
57                <div class="btn btn-outline-primary" data-bbox="488 681 508 691">
58                <div class="btn btn-outline-primary" data-bbox="488 691 508 701">
59                <div class="btn btn-outline-primary" data-bbox="488 701 508 711">
60                <div class="btn btn-outline-primary" data-bbox="488 711 508 721">
61                <div class="btn btn-outline-primary" data-bbox="488 721 508 731">
62                <div class="btn btn-outline-primary" data-bbox="488 731 508 741">
63                <div class="btn btn-outline-primary" data-bbox="488 741 508 751">
64                <div class="btn btn-outline-primary" data-bbox="488 751 508 761">
65                <div class="btn btn-outline-primary" data-bbox="488 761 508 771">
66                <div class="btn btn-outline-primary" data-bbox="488 771 508 781">
67                <div class="btn btn-outline-primary" data-bbox="488 781 508 791">
68                <div class="btn btn-outline-primary" data-bbox="488 791 508 801">
69                <div class="btn btn-outline-primary" data-bbox="488 801 508 811">
70                <div class="btn btn-outline-primary" data-bbox="488 811 508 821">
71                <div class="btn btn-outline-primary" data-bbox="488 821 508 831">
72                <div class="btn btn-outline-primary" data-bbox="488 831 508 841">
73                <div class="btn btn-outline-primary" data-bbox="488 841 508 851">
74                <div class="btn btn-outline-primary" data-bbox="488 851 508 861">
75                <div class="btn btn-outline-primary" data-bbox="488 861 508 871">
76                <div class="btn btn-outline-primary" data-bbox="488 871 508 881">
77                <div class="btn btn-outline-primary" data-bbox="488 881 508 891">
78                <div class="btn btn-outline-primary" data-bbox="488 891 508 901">
79                <div class="btn btn-outline-primary" data-bbox="488 901 508 911">
80                <div class="btn btn-outline-primary" data-bbox="488 911 508 921">
81                <div class="btn btn-outline-primary" data-bbox="488 921 508 931">
82                <div class="btn btn-outline-primary" data-bbox="488 931 508 941">
83                <div class="btn btn-outline-primary" data-bbox="488 941 508 951">
84                <div class="btn btn-outline-primary" data-bbox="488 951 508 961">
85                <div class="btn btn-outline-primary" data-bbox="488 961 508 971">
86                <div class="btn btn-outline-primary" data-bbox="488 971 508 981">
87                <div class="btn btn-outline-primary" data-bbox="488 981 508 991">
88                <div class="btn btn-outline-primary" data-bbox="488 991 508 1000">
89                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
90                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
91                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
92                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
93                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
94                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
95                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
96                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
97                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
98                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
99                <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
100               <div class="btn btn-outline-primary" data-bbox="488 1000 508 1000">
101             </div>
102           </div>
103         </div>
104       </div>
105     </div>
106   </div>
107 </div>
```

Gambar 23. Kode edit.ejs

terdapat header yang menampilkan judul "Editing:" diikuti dengan nama pelanggan yang sedang diedit. Bagian metadata menampilkan informasi terkini tentang pelanggan, seperti tanggal terakhir kali data diperbarui dan ID pengguna. formulir pengeditan yang terdiri dari input fields untuk mengubah berbagai atribut pelanggan seperti nama, nomor identitas, nomor telepon, alamat email, fakultas, jurusan, dan detail mahasiswa. Setiap input field sudah diisi dengan nilai yang sesuai berdasarkan data pelanggan yang sedang diedit. Terdapat tombol "Update Student" untuk menyimpan perubahan yang dibuat, serta tombol "Delete Student" yang akan membuka dialog konfirmasi untuk menghapus entri pelanggan.

```
1 <div class="modal fade" tabindex="-1" role="dialog" id="deleteModal" style="border: 2px solid white;">
2   <div class="modal-dialog" role="document">
3     <div class="modal-content">
4       <div class="modal-header" style="background-color: black;">
5         <div class="modal-title" style="color: white;">You are about to remove a customer record.</div>
6         <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
7       </div>
8       <div class="modal-body" style="background-color: black;">
9         <p style="color: white;">
10           This will remove the Student record of <b class="fw-bold"><%= customer.Name %> </b><br/>
11           Are you sure?
12         </p>
13       </div>
14       <div class="modal-footer" style="background-color: black;">
15         <button type="button" class="btn btn-outline-secondary" data-bs-dismiss="modal">Close</button>
16         <form action="/edit/<%= customer._id %>" method="DELETE" method="POST" class="position-relative">
17           <button type="submit" class="btn btn-outline-danger">Yes, Remove Student</button>
18         </form>
19       </div>
20     </div>
21   </div>
22 </div>
```

Gambar 24. Kode edit.ejs

- Tag Modal : Kode dimulai dengan tag div yang menunjukkan sebuah modal. Modal ini memiliki atribut `id="deleteModal"` untuk mengidentifikasi modal tersebut.
- Judul dan Tombol Close : Bagian modal header menampilkan judul dan tombol close (data-bs-dismiss="modal").

- Isi Pesan Konfirmasi : Di dalam modal body, terdapat pesan konfirmasi yang menunjukkan nama pelanggan yang akan dihapus, diambil dari variabel `customer.Name`.
- Tombol Konfirmasi : Modal footer berisi dua tombol. Tombol pertama adalah "Close", yang mengizinkan pengguna untuk menutup modal tanpa menghapus entri pelanggan. Tombol kedua adalah "Yes, Remove Student", yang akan mengirimkan permintaan penghapusan melalui formulir dengan metode DELETE.

c. Kode Studentcontrollers.js



```

1  exports.editPost = async(req,res)=>{
2
3      try {
4          await Customer.findByIdAndUpdate(req.params.id , {
5              Name:req.body.Name,
6              Nim:req.body.Nim,
7              tel: req.body.tel,
8              email: req.body.email,
9              Fakultas : req.body.Fakultas,
10             Jurusan : req.body.Jurusan,
11             details: req.body.details,
12
13             updatedAt : Date.now()
14         });
15
16         await res.redirect(`/edit/${req.params.id}`);
17     } catch (error) {
18         console.log(error)
19     }
20 }
21
22
23

```

Gambar 25. Kode Studentcontroller.js

- Async Function : Fungsi ini didefinisikan sebagai sebuah async function, yang memungkinkan penggunaan operasi-operasi asynchronous di dalamnya tanpa memblokir eksekusi kode selanjutnya.
- Pembaruan Data Pelanggan : Fungsi ini menggunakan metode `findByIdAndUpdate()` dari model `Customer` untuk mencari dan memperbarui data pelanggan berdasarkan ID yang diberikan dalam permintaan (`req.params.id`). Data yang diperbarui mencakup nama, nim, nomor telepon, alamat email, fakultas, jurusan, dan detail pelanggan. Setelah pembaruan selesai, atribut `updatedAt` juga diperbarui dengan menggunakan `Date.now()`.

d. Studentcontrollers.js



```

1  exports.edit = async(req,res)=>{
2
3      try{
4          const customer = await Customer.findOne({_id:req.params.id})
5          const locals = {
6              title: 'edits',
7              description : "Management System"
8          };
9          res.render('customer/edit', {
10             locals,
11             customer
12         })
13     }catch(error){
14         console.log(error)
15     }
16 }

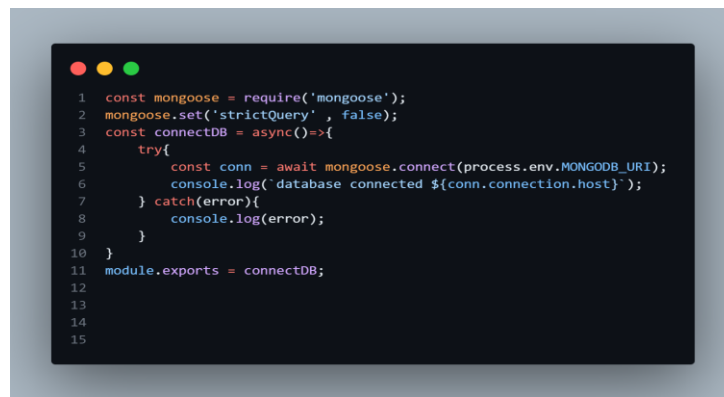
```

Gambar 26. Kode Studentcontroller.js

- pencarian Data Pelanggan : Di dalam blok try, fungsi ini mencari data pelanggan menggunakan metode `findOne()` dari model `Customer`. Pencarian ini dilakukan berdasarkan ID yang diterima dari parameter `req.params.id`. Jika data pelanggan ditemukan, informasi pelanggan tersebut akan dipassing ke dalam template rendering.
- Rendering View : Setelah data pelanggan berhasil ditemukan, objek pelanggan bersama dengan objek lokal `locals` digunakan untuk merender halaman view. Ini dilakukan menggunakan template engine yang digunakan oleh aplikasi web (seperti pug atau ejs). Halaman view yang dihasilkan adalah halaman pengeditan yang memuat data pelanggan yang akan diubah

3.1.11 Database

a. Connection :

A screenshot of a code editor with a dark background and light-colored text. The code is written in JavaScript and shows the setup for connecting to a MongoDB database using Mongoose. It includes comments in Indonesian. The code is as follows:

```
1  const mongoose = require('mongoose');
2  mongoose.set('strictQuery', false);
3  const connectDB = async()=>{
4    try{
5      const conn = await mongoose.connect(process.env.MONGODB_URI);
6      console.log(`database connected ${conn.connection.host}`);
7    } catch(error){
8      console.log(error);
9    }
10 }
11 module.exports = connectDB;
12
13
14
15
```

Gambar 27. Kode Connection

dilakukan pengaturan dengan menonaktifkan mode strict query pada Mongoose melalui `mongoose.set('strictQuery', false);`. Ini memungkinkan untuk melakukan query tanpa perlu menggunakan operator `$` pada field. Fungsi `connectDB` merupakan async function yang berisi proses koneksi dengan basis data MongoDB. Proses ini dilakukan secara asynchronous menggunakan metode `mongoose.connect()`. Setelah koneksi berhasil, informasi host basis data akan dicetak ke konsol untuk tujuan pemantauan. Untuk menangani kemungkinan kesalahan selama proses koneksi, terdapat blok `try-catch` yang menangkap dan menangani kesalahan yang mungkin terjadi. Kesalahan yang terjadi akan dicetak ke konsol untuk membantu pemecahan masalah.

3.1.12 Model

a. Studentsmodel (model/students.js)



```
1 const mongoose = require('mongoose');
2
3 const Schema = mongoose.Schema;
4
5 const CustomerSchema = new Schema({
6   Name: {
7     type: String,
8     required: true
9   },
10   Nim: {
11     type: String,
12     required: true
13   },
14   Fakultas: {
15     type: String,
16     required: true
17   },
18   Jurusan: {
19     type: String,
20     required: true
21   },
22   tel: {
23     type: String,
24     required: true
25   },
26   details: {
27     type: String,
28     required: true
29   },
30   email: {
31     type: String,
32     required: true
33   },
34   createdAt: {
35     type: Date,
36     default: Date.now()
37   },
38   updatedAt: {
39     type: Date,
40     default: Date.now()
41   },
42 });
43 module.exports = mongoose.model('Customer', CustomerSchema);
44
45
46
47
48
49
```

Gambar 28. Kode model

- Definisi Skema Pelanggan : Objek `CustomerSchema` merupakan definisi dari struktur data mahasiswa. Skema ini menentukan atribut-atribut yang diperlukan untuk setiap entitas pelanggan, seperti nama, nim, fakultas, jurusan, nomor telepon, detail, dan alamat email. Setiap atribut memiliki tipe data yang sesuai dan beberapa mungkin memiliki konfigurasi tambahan, seperti `required: true` yang menandakan atribut tersebut wajib diisi.
- Default Value untuk Tanggal : Terdapat dua atribut tambahan dalam skema, yaitu `createdAt` dan `updatedAt`, yang bertipe data `Date`. Atribut ini akan menyimpan tanggal dan waktu ketika entitas pelanggan dibuat dan terakhir diupdate. Kedua atribut ini memiliki nilai default yang diset ke `Date.now()`, yang berarti nilai defaultnya adalah waktu saat objek model dibuat.

b. loginmodel (model/login.js)



```
1 const loginSchema = new mongoose.Schema({
2   name: {
3     type: String,
4     required: true
5   },
6   password: {
7     type: String,
8     required: true
9   }
10 })
```

Gambar 29. Kode loginmodel

- Pembuatan Skema : Objek `logInSchema` dibuat menggunakan konstruktor `mongoose.Schema()`. Ini adalah skema yang digunakan untuk mendefinisikan struktur data untuk model data masuk.
- Atribut-atribut Skema : Skema ini memiliki dua atribut, yaitu `name` dan `password`. Kedua atribut ini memiliki tipe data `String` dan ditandai sebagai wajib diisi (`required: true`). `name` digunakan untuk menyimpan nama pengguna yang akan digunakan untuk login, sedangkan `password` digunakan untuk menyimpan kata sandi yang sesuai.

c. Model untuk curd lainnya :

- model/Professor.js :

```

1  const mongoose = require('mongoose');
2  const Schema = mongoose.Schema;
3
4  const DosenSchema = new Schema({
5    Name:{
6      type:String,
7      required : true
8    },
9    ID:{
10     type:String,
11     required : true
12   },
13   Fakultas:{
14     type:String,
15     required : true
16   },
17   Course:{
18     type:String,
19     required : true
20   },
21   tel:{
22     type:String,
23     required : true
24   },
25   email:{
26     type:String,
27     required : true
28   },
29   last_educ: {
30     type: String,
31     required: true
32   },
33   createdAt:{
34     type:Date,
35     default : Date.now()
36   },
37   updatedAt:{
38     type:Date,
39     default : Date.now()
40   },
41 });
42
43 module.exports = mongoose.model('Dosen' ,DosenSchema);

```

Gambar 30. Kode Professor.js

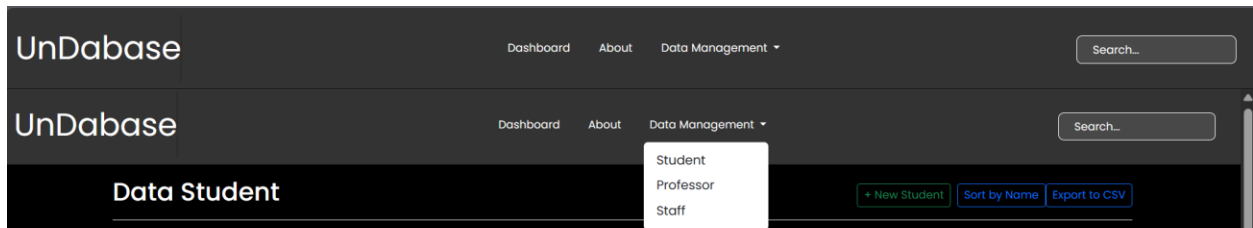
- model/staff.js :

```
1  const mongoose = require('mongoose');
2
3  const Schema = mongoose.Schema;
4
5  const StaffSchema = new Schema({
6    first_name: {
7      type: String,
8      required: true
9    },
10   last_name: {
11     type: String,
12     required: true
13   },
14   id: {
15     type: String,
16     required: true
17   },
18   umur: {
19     type: String,
20     required: true
21   },
22   alamat: {
23     type: String,
24     required: true
25   },
26   tempat_tanggal_lahir: {
27     type: String,
28     required: true
29   },
30   email: {
31     type: String,
32     required: true
33   },
34   tel: {
35     type: String,
36     required: true
37   },
38   job: {
39     type: String,
40     required: true
41   },
42   createdAt: {
43     type: Date,
44     default: Date.now()
45   },
46   updatedAt: {
47     type: Date,
48     default: Date.now()
49   },
50 });
51
52 module.exports = mongoose.model('Staff', StaffSchema);
53
54
55
56
57
58
```

Gambar 31. Kode staff.js

3.1.3 HEADER

a. Tampilan :



Gambar 32. Tampilan Header

Ini merupakan tampilan untuk menampilkan nav bar yang mempermudah navigasi admin dalam memmanage data , inilah tampilannya :

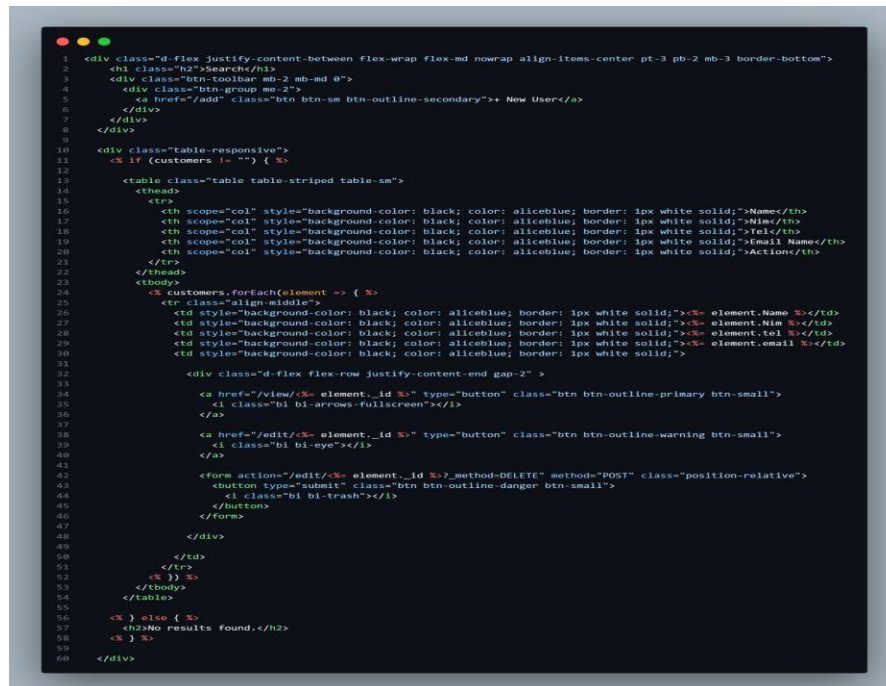
b. Kode header.js :



Gambar 33. Kode header.js

- Link Logo : Tag `` digunakan untuk membuat tautan ke halaman Dashboard dengan teks "UnDabase". Kelas `navbar-brand` menunjukkan bahwa ini adalah bagian dari merek (brand) navbar.
- Dropdown Menu : Salah satu menu memiliki dropdown menu yang menampilkan opsi untuk mengelola data, seperti mahasiswa, professor, dan staff. Dropdown menu ini diimplementasikan menggunakan kelas `dropdown` dan `dropdown-menu` dari Bootstrap.
- Form Pencarian : Terdapat juga sebuah form pencarian (`<form>`) yang memungkinkan pengguna untuk melakukan pencarian di dalam aplikasi. Ketika pengguna memasukkan kata kunci dan menekan enter, permintaan pencarian akan dikirim ke URL yang ditentukan pada atribut `action` form

c. search.ejs



Gambar 34. Kode search.ejs

- Tabel Hasil Pencarian : Tabel yang ditampilkan menggunakan kelas-kelas Bootstrap untuk membuatnya responsif dan bergaya. Data pelanggan yang sesuai dengan hasil pencarian ditampilkan dalam bentuk baris dan kolom. Setiap baris menampilkan informasi seperti nama, nim, nomor telepon, dan alamat email pelanggan.
- Aksi untuk Setiap Pelanggan : Setiap baris dalam tabel memiliki kolom aksi yang berisi tautan untuk melihat detail pelanggan (`view`), mengedit informasi pelanggan (`edit`), dan menghapus pelanggan (`delete`). Tautan-tautan ini dibuat dengan menggunakan tag `` atau formulir `` yang mengarahkan ke rute yang sesuai pada aplikasi.
- Pesan Jika Tidak Ada Hasil : Jika tidak ada hasil yang ditemukan dalam pencarian, maka sebuah pesan yang menyatakan "No results found." akan ditampilkan.

d. studentscontroller.js



```

1
2 exports.searchCustomers = async (req, res) => {
3   const locals = {
4     title: "Search Customer Data",
5     description: "Free NodeJs User Management System",
6   };
7
8   try {
9     let searchTerm = req.body.searchTerm;
10    const searchNoSpecialChar = searchTerm.replace(/[^a-zA-Z0-9 ]/g, "");
11
12    const customers = await Customer.find({
13      $or: [
14        { Name: { $regex: new RegExp(searchNoSpecialChar, "i") } },
15        { email: { $regex: new RegExp(searchNoSpecialChar, "i") } },
16      ],
17    });
18
19    res.render("search", {
20      customers,
21      locals,
22    });
23  } catch (error) {
24    console.log(error);
25  }
26 }
27

```

Gambar 35. kode studentcontroller.js

- Async Function : Seperti yang terlihat dari deklarasi `async`, fungsi ini adalah sebuah async function. Ini memungkinkan penggunaan operasi-operasi asynchronous seperti pemanggilan database tanpa menghentikan eksekusi kode selanjutnya.
- Pemrosesan Pencarian : Pertama, fungsi mengambil kata kunci pencarian dari `req.body.searchTerm`. Kemudian, kata kunci tersebut dihilangkan karakter khususnya menggunakan ekspresi reguler untuk menghindari masalah pencarian. Setelah itu, sebuah query pencarian dibuat menggunakan `\$or` untuk mencari entitas pelanggan berdasarkan nama atau alamat email yang cocok dengan kata kunci pencarian. Hasilnya disimpan dalam variabel `customers`.
- Rendering Tampilan : Setelah mendapatkan hasil pencarian, fungsi ini merender tampilan pencarian dengan menggunakan template engine yang digunakan oleh aplikasi. Data pelanggan yang sesuai dengan hasil pencarian disertakan dalam proses rendering.

3.2 Pembagian Kerja Anggota Kelompok

1. Hans Santoso - Scrum Master

Hans Santoso memegang peran vital sebagai Scrum Master dalam tim. Sebagai Scrum Master, Hans bertanggung jawab dalam fase persiapan awal proyek UnDabase, di mana dia terlibat langsung dalam menentukan topik dan tema yang akan diusung oleh UnDabase. Selain itu, Hans juga bertanggung jawab dalam membuat CSS untuk tampilan frontend, yang merupakan aspek penting dalam pengalaman pengguna. Tugasnya tidak berhenti di situ, Hans juga turut menambahkan fitur sort yang memudahkan pengguna dalam mengatur data. Selain kontribusi teknisnya, Hans juga berperan dalam pembuatan halaman staff page, serta memiliki tanggung jawab dalam penulisan laporan proyek dan pembuatan video demonstrasi untuk menggambarkan fitur-fitur UnDabase secara menyeluruh.

2. Lekrey Jerel Jacob Laipiopa - Tim Pengembangan

Jerel adalah anggota tim pengembangan yang memiliki peran besar dalam pembangunan frontend UnDabase. Dalam fase pengembangan frontend, Jerel aktif menggunakan EJS (Embedded JavaScript) dan JavaScript untuk menghasilkan tampilan yang menarik dan interaktif. Selain itu, Jerel juga bertanggung jawab dalam membuat model database, yang menjadi pondasi penting dalam pengelolaan data mahasiswa, staff, dan professor. Jerel turut menambahkan fitur filter dan download CSV, yang meningkatkan fungsionalitas UnDabase bagi pengguna. Tugasnya tidak berhenti di situ, Jerel juga memiliki peran dalam pembuatan halaman student page, serta bertanggung jawab dalam pengujian akhir untuk memastikan bahwa semua fitur berjalan dengan lancar dan sesuai dengan kebutuhan pengguna.

3. Sabrina Phalosa Phai - Product Owner

Sabrina berperan sebagai Product Owner dalam tim pengembangan UnDabase. Sebagai Product Owner, Sabrina memiliki tanggung jawab dalam konfigurasi database, yang merupakan landasan penting dalam pengelolaan data secara efisien. Selain itu, Sabrina juga terlibat dalam pengembangan fitur search, yang memungkinkan pengguna untuk menemukan informasi dengan cepat dan akurat. Sabrina juga bertanggung jawab dalam pembuatan halaman professor page, yang menyediakan informasi terkait dosen dan materi ajar. Selain kontribusi teknisnya, Sabrina juga memiliki peran dalam pembuatan presentasi PowerPoint (PPT) yang menjelaskan secara komprehensif tentang UnDabase, yang akan digunakan untuk mempresentasikan proyek kepada pihak terkait.

Dengan pembagian tugas yang jelas dan kontribusi yang berfokus, tim pengembangan UnDabase dapat bekerja secara efisien untuk mencapai tujuan pengembangan sistem manajemen mahasiswa yang adaptif dan responsif.

BAB 4

KESIMPULAN

Pengembangan UnDabase sebagai sistem manajemen mahasiswa melalui pendekatan Agile Scrum telah menghasilkan solusi yang adaptif dan efisien. Proyek ini mencapai berbagai tonggak penting yang mencerminkan keberhasilan dalam menghadapi tantangan teknis dan manajerial.

Dalam fase awal proyek, persiapan dan penentuan tema UnDabase dilakukan dengan cermat untuk memastikan kesesuaian dengan kebutuhan pengguna. Pengembangan frontend dengan CSS, EJS, dan JavaScript menghasilkan antarmuka yang menarik dan responsif, sedangkan penambahan fitur-fitur seperti sort, filter, dan search meningkatkan fungsionalitas sistem.

Aspek backend juga diperhatikan dengan baik melalui pembuatan model database dan konfigurasi yang sesuai, memastikan integritas dan keamanan data. Implementasi CRUD operations pada halaman-halaman pengelolaan data menunjukkan komitmen dalam menyediakan pengalaman pengguna yang intuitif dan efisien.

Pengujian yang dilakukan secara menyeluruh memastikan kualitas dan keandalan UnDabase sebelum peluncuran, sementara dokumentasi yang komprehensif seperti laporan dan video demonstrasi memberikan panduan yang jelas bagi pengguna.

Dengan keseluruhan pencapaian ini, UnDabase tidak hanya memenuhi kebutuhan operasional institusi pendidikan, tetapi juga memberikan nilai tambah dalam pengelolaan informasi mahasiswa. Diharapkan sistem ini dapat meningkatkan efisiensi administrasi institusi pendidikan dan memberikan pengalaman pengguna yang lebih baik.

DAFTAR PUSTAKA

1. Martin, R.C. dan Martin, M., Agile Principles, Patterns, and Practice in C#, Upper Saddle River: Pearson Education, 2006.
2. Fauzi, I. (2023). Manajemen Proyek Agile: Sistem Informasi untuk Lingkungan Dinamis. Teknologi Terkini. URL: <http://teknologiterkini.org/index.php/terkini/article/view/488/448>
3. Harris, Chandler. Agile Scrum Artifacts. URL: <https://www.atlassian.com/agile/scrum/artifacts>
4. Huda, A. N. Teknik Penentuan Prioritas Product Backlog pada Metode Scrum Menggunakan Pendekatan Program Dinamis. Bandung. Institut Teknologi Bandung.