

Laporan Praktikum
Mata Kuliah Pemrograman Web dan
Pembelajaran Berorientasi Objek



UJIAN TENGAH SEMESTER
"CRUD - SESSION"

Dosen Pengampu :
Willdan Aprizal Arifin, S.Pd., M.Kom.

Disusun Oleh :
Silvia Isti Lestary
2311883

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

CRUD (Create Read Update Delete) adalah suatu akronim yang diartikan untuk membentuk perintah basis data. Siklus CRUD dirancang sebagai empat metode fungsi dasar untuk meningkatkan penyimpanan persisten — dengan database. Prinsip-prinsip siklus CRUD didefinisikan sebagai CREATE, READ / RETRIEVE, UPDATE, dan DELETE. Dalam pengembangan perangkat lunak modern, CRUD telah melampaui asal-usulnya sebagai fungsi dasar dari sebuah basis data dan sekarang memetakan dirinya untuk prinsip-prinsip desain untuk aplikasi dinamis seperti protokol HTTP, DDS, dan SQL. (Adrian Halomoan & Putra Kharisma, 2021)

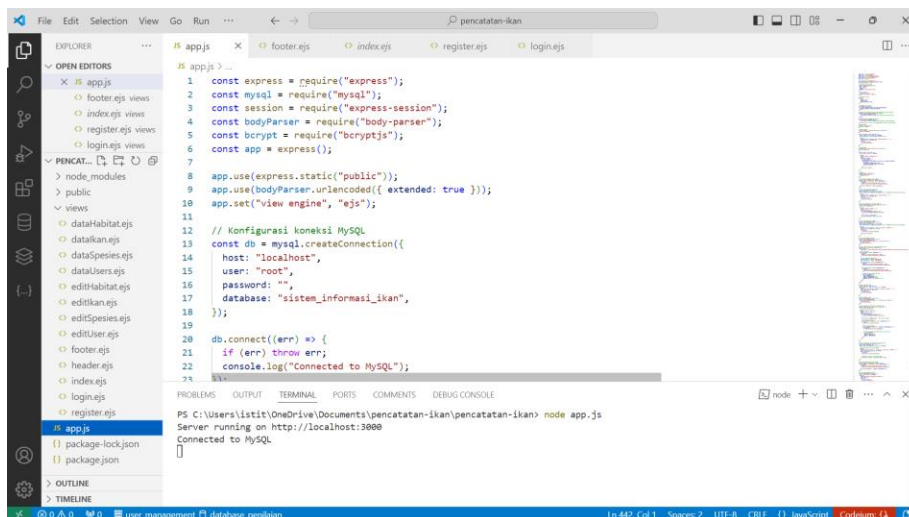
Session adalah mekanisme yang digunakan dalam aplikasi web untuk menyimpan informasi pengguna secara sementara saat mereka berinteraksi dengan aplikasi tersebut. Dalam konteks Node.js, session digunakan untuk melacak dan mengingat status pengguna di antara permintaan HTTP (request) yang berbeda.

II. ALAT DAN BAHAN

1. Laptop
2. Browser atau terminal (Node.js untuk menjalankan JavaScript di luar browser)
3. Visual Studio Code
4. Node.js

III. PENJELASAN

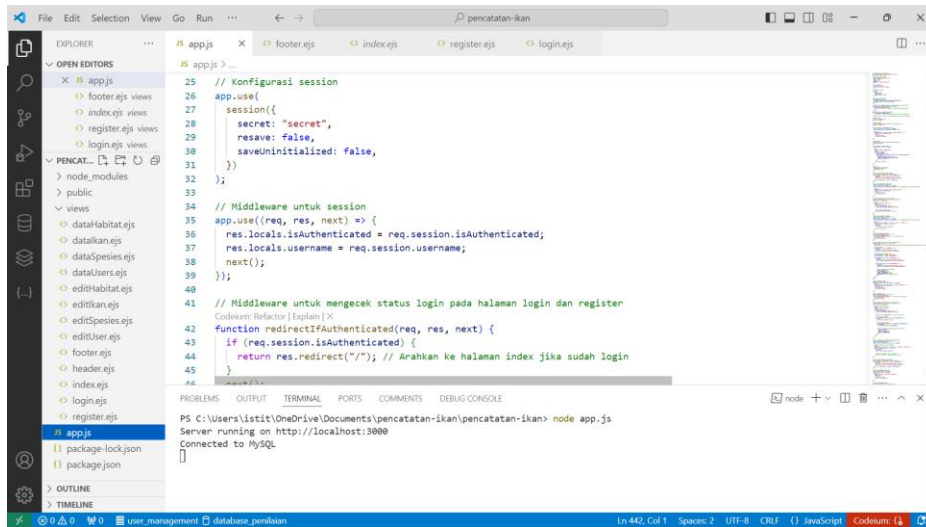
1. App.js



```
1 const express = require("express");
2 const mysql = require("mysql");
3 const session = require("express-session");
4 const bodyParser = require("body-parser");
5 const bcrypt = require("bcryptjs");
6 const app = express();
7
8 app.use(express.static("public"));
9 app.use(bodyParser.urlencoded({ extended: true }));
10 app.set("view engine", "ejs");
11
12 // Konfigurasi koneksi MySQL
13 const db = mysql.createConnection({
14   host: "localhost",
15   user: "root",
16   password: "",
17   database: "sistem_informasi_ikan",
18 });
19
20 db.connect((err) => {
21   if (err) throw err;
22   console.log("Connected to MySQL");
23 });
```

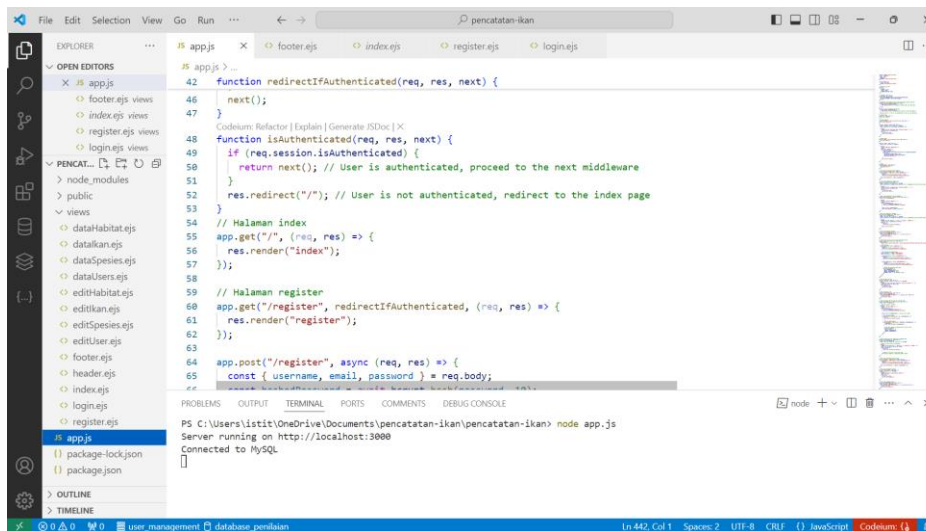
Terminal Output:

```
PS C:\Users\liti\OneDrive\Documents\pencatatan-ikan\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL
```



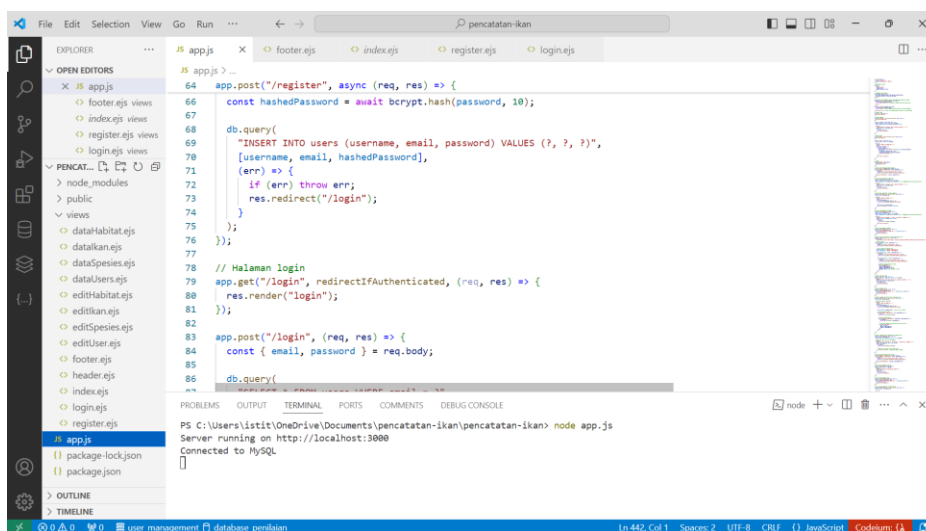
```
25 // Konfigurasi session
26 app.use(
27   session({
28     secret: "secret",
29     resave: false,
30     saveUninitialized: false,
31   })
32 );
33
34 // Middleware untuk session
35 app.use((req, res, next) => {
36   res.locals.isAuthenticated = req.session.isAuthenticated;
37   res.locals.username = req.session.username;
38   next();
39 });
40
41 // Middleware untuk mengecek status login pada halaman login dan register
42 function redirectIfAuthenticated(req, res, next) {
43   if (req.session.isAuthenticated) {
44     return res.redirect("/"); // Arahkan ke halaman index jika sudah login
45   }
46 }
47
48 // Halaman index
49 app.get("/", (req, res) => {
50   res.render("index");
51 });
52
53 // Halaman register
54 app.get("/register", redirectIfAuthenticated, (req, res) => {
55   res.render("register");
56 });
57
58 // Halaman login
59 app.get("/login", redirectIfAuthenticated, (req, res) => {
60   res.render("login");
61 });
```

PS C:\Users\istiti\OneDrive\Documents\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL



```
42 function redirectIfAuthenticated(req, res, next) {
43   next();
44 }
45
46 function isAuthenticated(req, res, next) {
47   if (req.session.isAuthenticated) {
48     return next(); // User is authenticated, proceed to the next middleware
49   }
50   res.redirect("/"); // User is not authenticated, redirect to the index page
51 }
52
53 // Halaman index
54 app.get("/", (req, res) => {
55   res.render("index");
56 });
57
58 // Halaman register
59 app.get("/register", redirectIfAuthenticated, (req, res) => {
60   res.render("register");
61 });
62
63 // Halaman login
64 app.get("/login", redirectIfAuthenticated, (req, res) => {
65   res.render("login");
66 });
67
68 // Endpoint untuk register
69 app.post("/register", async (req, res) => {
70   const { username, email, password } = req.body;
71   const hashedPassword = await bcrypt.hash(password, 10);
72   db.query(
73     "INSERT INTO users (username, email, password) VALUES (?, ?, ?)",
74     [username, email, hashedPassword],
75     (err) => {
76       if (err) throw err;
77       res.redirect("/login");
78     }
79   );
80 }
81
82 // Halaman login
83 app.get("/login", redirectIfAuthenticated, (req, res) => {
84   res.render("login");
85 });
86
87 // Endpoint untuk login
88 app.post("/login", (req, res) => {
89   const { email, password } = req.body;
90   db.query(
91     "SELECT * FROM users WHERE email = ?",
92     [email],
93     (err, results) => {
94       if (err) throw err;
95       if (results.length === 0) {
96         res.status(401).send("Email tidak ditemukan");
97       } else {
98         const user = results[0];
99         bcrypt.compare(password, user.password, (err, isMatch) => {
100           if (err) throw err;
101           if (isMatch) {
102             req.session.isAuthenticated = true;
103             req.session.username = user.username;
104             res.redirect("/");
105           } else {
106             res.status(401).send("Password salah");
107           }
108         });
109       }
110     }
111   );
112 }
```

PS C:\Users\istiti\OneDrive\Documents\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL



```
64 app.post("/register", async (req, res) => {
65   const { username, email, password } = req.body;
66   const hashedPassword = await bcrypt.hash(password, 10);
67   db.query(
68     "INSERT INTO users (username, email, password) VALUES (?, ?, ?)",
69     [username, email, hashedPassword],
70     (err) => {
71       if (err) throw err;
72       res.redirect("/login");
73     }
74   );
75 }
76
77 // Halaman login
78 app.get("/login", redirectIfAuthenticated, (req, res) => {
79   res.render("login");
80 });
81
82 // Endpoint untuk login
83 app.post("/login", (req, res) => {
84   const { email, password } = req.body;
85   db.query(
86     "SELECT * FROM users WHERE email = ?",
87     [email],
88     (err, results) => {
89       if (err) throw err;
90       if (results.length === 0) {
91         res.status(401).send("Email tidak ditemukan");
92       } else {
93         const user = results[0];
94         bcrypt.compare(password, user.password, (err, isMatch) => {
95           if (err) throw err;
96           if (isMatch) {
97             req.session.isAuthenticated = true;
98             req.session.username = user.username;
99             res.redirect("/");
100           } else {
101             res.status(401).send("Password salah");
102           }
103         });
104       }
105     }
106   );
107 }
```

PS C:\Users\istiti\OneDrive\Documents\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL

```
83 app.post("/login", (req, res) => {
84   db.query(
85     "SELECT * FROM users WHERE email = ?",
86     [email],
87     async (err, results) => {
88       if (err) throw err;
89       if (results.length > 0) {
90         const isMatch = await bcrypt.compare(password, results[0].password);
91         if (isMatch) {
92           req.session.isAuthenticated = true;
93           req.session.username = results[0].username;
94           return res.redirect("/");
95         }
96       }
97       res.redirect("/login");
98     }
99   );
100 }
101 // Logout
102 app.get("/logout", (req, res) => {
103   req.session.destroy(() => {
104     res.redirect("/");
105   });
106 }
107
```

PROBLEMS OUTPUT TERMINAL PORTS COMMENTS DEBUG CONSOLE

PS C:\Users\Istif\OneDrive\Documents\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL

```
108 // Logout
109 app.get("/logout", (req, res) => {
110   req.session.destroy(() => {
111     res.redirect("/");
112   });
113 }
114 // Route untuk menampilkan daftar pengguna
115 app.get("/users", isAuthenticated, (req, res) => {
116   db.query("SELECT * FROM users", (error, userResults) => {
117     if (error) {
118       console.error("Error fetching user data:", error);
119       return res.status(500).send("Error fetching user data");
120     }
121     res.render("dataUsers", { users: userResults });
122   });
123 }
124 // Route untuk menambahkan pengguna baru
125 app.post("/users/add", (req, res) => {
126   const { username, email, password } = req.body;
127   const hashedPassword = bcrypt.hashSync(password, 10); // Menggunakan bcrypt untuk hash password
128 }
129
```

PROBLEMS OUTPUT TERMINAL PORTS COMMENTS DEBUG CONSOLE

PS C:\Users\Istif\OneDrive\Documents\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL

```
122 // Route untuk menambahkan pengguna baru
123 app.post("/users/add", (req, res) => {
124   const { username, email, password } = req.body;
125   const hashedPassword = bcrypt.hashSync(password, 10); // Menggunakan bcrypt untuk hash password
126 }
127 db.query(
128   "INSERT INTO users (username, email, password) VALUES (?, ?, ?)",
129   [username, email, hashedPassword],
130   (err) => {
131     if (err) throw err;
132     res.redirect("/users");
133   }
134 );
135 // Route untuk menampilkan form edit pengguna
136 app.get("/users/edit/:id", isAuthenticated, (req, res) => {
137   const userId = req.params.id;
138   db.query(
139     "SELECT * FROM users WHERE id = ?",
140     [userId],
141     (err, results) => {
142       if (err) throw err;
143       if (results.length > 0) {
144         // User found, render edit form
145       } else {
146         // User not found, redirect to /users
147         res.redirect("/users");
148       }
149     }
150   );
151 }
152
```

PROBLEMS OUTPUT TERMINAL PORTS COMMENTS DEBUG CONSOLE

PS C:\Users\Istif\OneDrive\Documents\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL

```
138 app.get("/users/edit/:id", isAuthenticated, (req, res) => {
139   db.query(
140     "SELECT * FROM users WHERE id = ?",
141     [req.params.id],
142     (error, userResults) => {
143       if (error || userResults.length === 0) {
144         return res.status(404).send("User not found");
145       }
146       const userItem = userResults[0];
147       res.render("editUser", { user: userItem });
148     }
149   );
150 }
151 // Route untuk memperbarui pengguna
152 app.post("/users/update/:id", (req, res) => {
153   const userId = req.params.id;
154   const { username, email, password } = req.body;
155   const hashedPassword = bcrypt.hashSync(password, 10); // Menggunakan bcrypt untuk hash password
156   db.query(
157     "UPDATE users SET username = ?, email = ?, password = ?, WHERE id = ?",
158     [username, email, hashedPassword, userId, password]
159   );
160   res.redirect("/users");
161 }
162 // Route untuk menghapus pengguna
163 app.post("/users/delete/:id", (req, res) => {
164   const userId = req.params.id;
165   db.query("DELETE FROM users WHERE id = ?", [userId], (err) => {
166     if (err) throw err;
167     res.redirect("/users");
168   });
169 }
170 }
171 // Route untuk menampilkan daftar ikan dan form tambah ikan
172 app.get("/ikan", isAuthenticated, (req, res) => {
173   const ikanQuery =
174     "SELECT ikan.ikan_id, species.nama_species, habitat.nama_habitat FROM ikan JOIN species ON ikan.species_id = species.id";
175   db.query(ikanQuery, (error, ikanResults) => {
176     if (error) {
177       console.error("Error fetching data:", error);
178       return res.status(500).send("Error fetching data");
179     }
180     // Fetch species and habitat data
181     const speciesQuery = "SELECT * FROM species";
182     const habitatQuery = "SELECT * FROM habitat";
183     db.query(speciesQuery, (error, speciesResults) => {
184       if (error) {
185         console.error("Error fetching species data:", error);
186         return res.status(500).send("Error fetching species data");
187       }
188     });
189     db.query(habitatQuery, (error, habitatResults) => {
190       if (error) {
191         console.error("Error fetching habitat data:", error);
192         return res.status(500).send("Error fetching habitat data");
193       }
194     });
195     res.render("ikan", { ikanResults, speciesResults, habitatResults });
196   });
197 }
198 }
199 }
200 }
201 }
202 }
203 }
```

```
156 app.post("/users/update/:id", (req, res) => {
157   db.query(
158     "UPDATE users SET username = ?, email = ?, password = ?, WHERE id = ?",
159     [username, email, hashedPassword, userId, password]
160   );
161   res.redirect("/users");
162 }
163 // Route untuk menghapus pengguna
164 app.post("/users/delete/:id", (req, res) => {
165   const userId = req.params.id;
166   db.query("DELETE FROM users WHERE id = ?", [userId], (err) => {
167     if (err) throw err;
168     res.redirect("/users");
169   });
170 }
171 // Route untuk menampilkan daftar ikan dan form tambah ikan
172 app.get("/ikan", isAuthenticated, (req, res) => {
173   const ikanQuery =
174     "SELECT ikan.ikan_id, species.nama_species, habitat.nama_habitat FROM ikan JOIN species ON ikan.species_id = species.id";
175   db.query(ikanQuery, (error, ikanResults) => {
176     if (error) {
177       console.error("Error fetching data:", error);
178       return res.status(500).send("Error fetching data");
179     }
180     // Fetch species and habitat data
181     const speciesQuery = "SELECT * FROM species";
182     const habitatQuery = "SELECT * FROM habitat";
183     db.query(speciesQuery, (error, speciesResults) => {
184       if (error) {
185         console.error("Error fetching species data:", error);
186         return res.status(500).send("Error fetching species data");
187       }
188     });
189     db.query(habitatQuery, (error, habitatResults) => {
190       if (error) {
191         console.error("Error fetching habitat data:", error);
192         return res.status(500).send("Error fetching habitat data");
193       }
194     });
195     res.render("ikan", { ikanResults, speciesResults, habitatResults });
196   });
197 }
198 }
199 }
200 }
201 }
202 }
203 }
```

```
182 // Route to display the list of fish and the add fish form
183 app.get("/ikan", isAuthenticated, (req, res) => {
184   const ikanQuery =
185     "SELECT ikan.ikan_id, species.nama_species, habitat.nama_habitat FROM ikan JOIN species ON ikan.species_id = species.id";
186   db.query(ikanQuery, (error, ikanResults) => {
187     if (error) {
188       console.error("Error fetching data:", error);
189       return res.status(500).send("Error fetching data");
190     }
191     // Fetch species and habitat data
192     const speciesQuery = "SELECT * FROM species";
193     const habitatQuery = "SELECT * FROM habitat";
194     db.query(speciesQuery, (error, speciesResults) => {
195       if (error) {
196         console.error("Error fetching species data:", error);
197         return res.status(500).send("Error fetching species data");
198       }
199     });
200     db.query(habitatQuery, (error, habitatResults) => {
201       if (error) {
202         console.error("Error fetching habitat data:", error);
203         return res.status(500).send("Error fetching habitat data");
204       }
205     });
206     res.render("ikan", { ikanResults, speciesResults, habitatResults });
207   });
208 }
209 }
210 }
211 }
212 }
213 }
```

```
183 app.get("/ikan", isAuthenticated, (req, res) => {
184   db.query(ikanQuery, (error, ikanResults) => {
185     db.query(speciesQuery, (error, speciesResults) => {
186       db.query(habitatQuery, (error, habitatResults) => {
187         if (error) {
188           console.error("Error fetching habitat data:", error);
189           return res.status(500).send("Error fetching habitat data");
190         }
191         // Render the view without edit mode
192         res.render("dataikan", {
193           ikan: ikanResults || [],
194           species: speciesResults || [],
195           habitat: habitatResults || [],
196           editNode: false, // Set editNode to false
197         });
198       });
199     });
200   });
201 });
202 // Route to add a new fish
203 app.post("/ikan/add", (req, res) => {
```

PS C:\Users\istiti\OneDrive\Documents\pencatatan-ikan\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL

```
222 // Route to add a new fish
223 app.post("/ikan/add", (req, res) => {
224   const { nama_ikan, species_id, habitat_id } = req.body;
225   db.query(
226     "INSERT INTO ikan (nama_ikan, species_id, habitat_id) VALUES (?, ?, ?)",
227     [nama_ikan, species_id, habitat_id],
228     (err, result) => {
229       if (err) throw err;
230       res.redirect("/ikan");
231     }
232   );
233 });
234 // Route to delete a fish
235 app.post("/ikan/delete/:id", (req, res) => {
236   const ikanId = req.params.id;
237   db.query("DELETE FROM ikan WHERE id = ?", [ikanId], (err, result) => {
238     if (err) throw err;
239     res.redirect("/ikan");
240   });
241 });
242 // Route to fetch the edit form for a specific ikan
243 app.get("/ikan/edit/:id", (req, res) => {
```

PS C:\Users\istiti\OneDrive\Documents\pencatatan-ikan\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL

```
244 // Route to display the edit form for a specific ikan
245 app.get("/ikan/edit/:id", isAuthenticated, (req, res) => {
246   const ikanId = req.params.id;
247   // Fetch the specific ikan data
248   db.query(
249     "SELECT * FROM ikan WHERE id = ?",
250     [ikanId],
251     (error, ikanResults) => {
252       if (error || ikanResults.length === 0) {
253         return res.status(404).send("Ikan not found");
254       }
255       const ikanItem = ikanResults[0]; // This will be an object
256       // Fetch all species data
257       db.query("SELECT * FROM species", (err, speciesResults) => {
258         if (err) {
259           return res.status(500).send(err);
260         }
261       });
262       // Fetch all habitat data
263       db.query("SELECT * FROM habitat", (err, habitatResults) => {
264         if (err) {
265           return res.status(500).send(err);
266         }
267       });
268     }
269   );
270   // Render the edit form
271   res.render("editikan", {
272     ikan: ikanItem,
273     species: speciesResults,
274     habitat: habitatResults,
275     editNode: true,
276   });
277 });
```

PS C:\Users\istiti\OneDrive\Documents\pencatatan-ikan\pencatatan-ikan> node app.js
Server running on http://localhost:3000
Connected to MySQL

This screenshot shows the VS Code editor with the `app.js` file open. The code defines a GET route for `/ikan/edit/:id` that checks if the user is authenticated. If not, it redirects to the login page. If authenticated, it fetches the ikan record and its associated species and habitat data from the database. The data is then passed to the `editikan` view. A comment indicates that the next step is to handle the form submission for updating the ikan.

```
245 app.get("/ikan/edit/:id", isAuthenticated, (req, res) => {
252   (error, ikanResults) => {
260     db.query("SELECT * FROM species", (err, speciesResults) => {
265       // Fetch all habitat data
266       db.query("SELECT * FROM habitat", (habitatErr, habitatResults) => {
267         if (habitatErr) {
268           return res.status(500).send(habitatErr);
269         }
270       }
271       // Pass ikan, species, and habitat data to the edit view
272       res.render("editikan", {
273         ikan: ikanItem,
274         species: speciesResults,
275         habitat: habitatResults,
276       });
277     });
278   });
279 }
280 );
281 });
282 // Route to handle the form submission for updating ikan
```

The Explorer sidebar on the left shows the project structure, including `node_modules`, `public`, `views`, and various `.ejs` files. The terminal at the bottom shows the command `node app.js` being executed, with output indicating the server is running on `http://localhost:3000` and connected to MySQL.

This screenshot shows the VS Code editor with the `app.js` file open, displaying the POST route for `/ikan/update/:id`. The route checks for authentication and extracts the `ikanId` from the request parameters. It then updates the `ikan` record in the database with the provided `nama_ikan`, `species_id`, and `habitat_id`. If the update is successful, it redirects the user to the `/ikan` page.

```
283 // Route to handle the form submission for updating ikan
284 app.post("/ikan/update/:id", (req, res) => {
285   const ikanId = req.params.id;
286   const { nama_ikan, species_id, habitat_id } = req.body;
287
288   // Update the ikan record in the database
289   db.query(
290     "UPDATE ikan SET nama_ikan = ?, species_id = ?, habitat_id = ? WHERE id = ?",
291     [nama_ikan, species_id, habitat_id, ikanId],
292     (error, results) => {
293       if (error) {
294         return res.status(500).send(error);
295       }
296       // Redirect to the ikan list or detail page
297       res.redirect("/ikan"); // Change this path as needed
298     }
299   );
300 }
301 );
```

The Explorer sidebar and terminal output are consistent with the previous screenshot, showing the project structure and the server running on `http://localhost:3000`.

This screenshot shows the VS Code editor with the `app.js` file open, displaying two additional routes. The first is a GET route for `/species` that fetches all species from the database and renders the `dataSpecies` view. The second is a POST route for `/species/add` that takes a new species record from the request body and inserts it into the database.

```
303 // Route untuk menampilkan daftar species
304 app.get("/species", isAuthenticated, (req, res) => {
305   db.query("SELECT * FROM species", (error, speciesResults) => {
306     if (error) {
307       console.error("Error fetching species data:", error);
308       return res.status(500).send("Error fetching species data");
309     }
310     res.render("dataSpecies", { species: speciesResults });
311   });
312 }
313 );
314 // Route untuk menambahkan species
315 app.post("/species/add", (req, res) => {
316   const { nama_species } = req.body;
```

The Explorer sidebar and terminal output are consistent with the previous screenshots, showing the project structure and the server running on `http://localhost:3000`.


```
File Edit Selection View Go Run ... pencatatan-ikan
EXPLORER
  JS app.js
  JS app.js views
  JS footer.ejs
  JS index.ejs
  JS register.ejs
  JS login.ejs
  JS public
  JS views
  JS dataHabitat.ejs
  JS dataikan.ejs
  JS dataSpecies.ejs
  JS dataUsers.ejs
  JS editHabitat.ejs
  JS editikan.ejs
  JS editSpecies.ejs
  JS editUser.ejs
  JS footer.ejs
  JS header.ejs
  JS index.ejs
  JS login.ejs
  JS register.ejs
  JS app.js
  JS package-lock.json
  JS package.json
  JS OUTLINE
  JS TIMELINE
  JS user_management
  JS database_penilaian

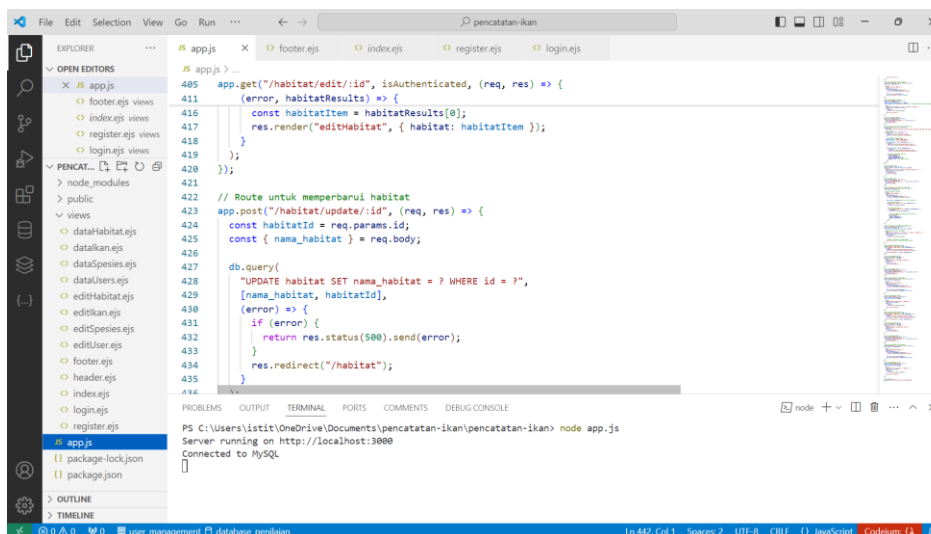
JS app.js
313 // Route untuk menambahkan spesies
314 app.post("/species/add", (req, res) => {
315   const { nama_spesies } = req.body;
316   db.query(
317     "INSERT INTO spesies (nama_spesies) VALUES (?)",
318     [nama_spesies],
319     (err) => {
320       if (err) throw err;
321       res.redirect("/species");
322     }
323   );
324 });
325
326 // Route untuk menghapus spesies
327 app.post("/species/delete/:id", (req, res) => {
328   const speciesId = req.params.id;
329   db.query("DELETE FROM spesies WHERE id = ?", [speciesId], (err) => {
330     if (err) throw err;
331     res.redirect("/species");
332   });
333 });
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

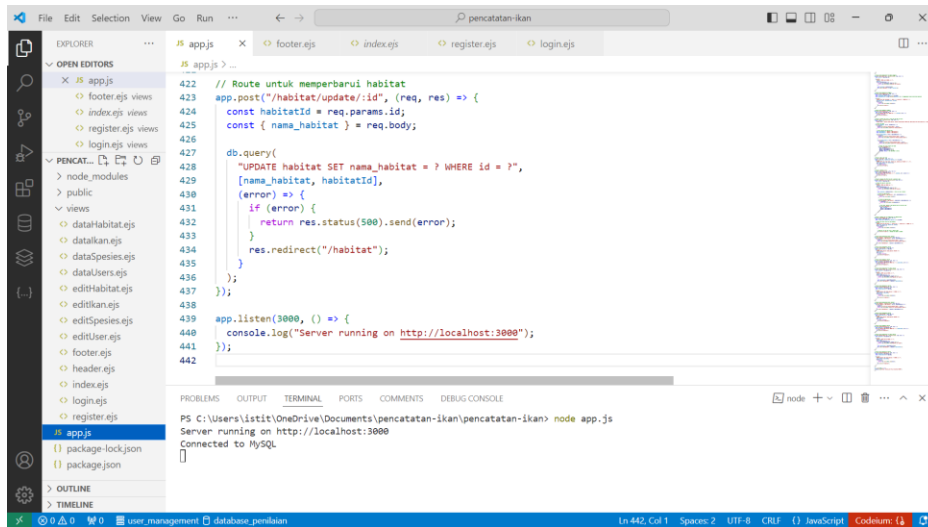
```
File Edit Selection View Go Run ... pencatatan-ikan
EXPLORER
  JS app.js
  JS app.js views
  JS footer.ejs
  JS index.ejs
  JS register.ejs
  JS login.ejs
  JS public
  JS views
  JS dataHabitat.ejs
  JS dataikan.ejs
  JS dataSpecies.ejs
  JS dataUsers.ejs
  JS editHabitat.ejs
  JS editikan.ejs
  JS editSpecies.ejs
  JS editUser.ejs
  JS footer.ejs
  JS header.ejs
  JS index.ejs
  JS login.ejs
  JS register.ejs
  JS app.js
  JS package-lock.json
  JS package.json
  JS OUTLINE
  JS TIMELINE
  JS user_management
  JS database_penilaian

JS app.js
336 // Route untuk menampilkan form edit spesies
337 app.get("/species/edit/:id", isAuthenticated, (req, res) => {
338   const speciesId = req.params.id;
339   db.query(
340     "SELECT * FROM spesies WHERE id = ?",
341     [speciesId],
342     (error, speciesResults) => {
343       if (error || speciesResults.length === 0) {
344         return res.status(404).send("Spesies not found");
345       }
346       const speciesItem = speciesResults[0];
347       res.render("editSpecies", { species: speciesItem });
348     }
349   );
350 });
351
352 // Route untuk memperbarui spesies
353 app.post("/species/update/:id", (req, res) => {
354   const speciesId = req.params.id;
355   const { nama_spesies } = req.body;
356   db.query(
357     "UPDATE spesies SET nama_spesies = ? WHERE id = ?",
358     [nama_spesies, speciesId],
359     (error) => {
360       if (error) {
361         return res.status(500).send(error);
362       }
363       res.redirect("/species");
364     }
365   );
366 });
367
368 // Route untuk menampilkan daftar habitat
369 app.get("/habitat", isAuthenticated, (req, res) => {
370   db.query("SELECT * FROM habitat", (error, habitatResults) => {
371     if (error) {
372       console.error("Error fetching habitat data:", error);
373     }
374   });
375 });
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

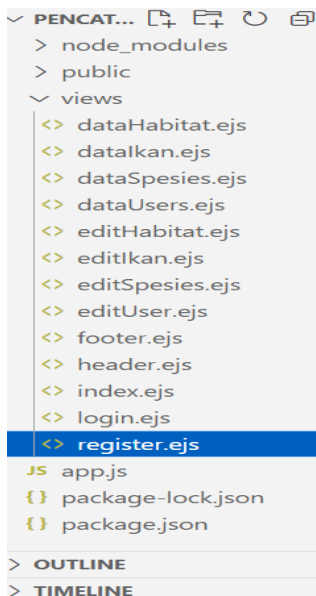
```
File Edit Selection View Go Run ... pencatatan-ikan
EXPLORER
  JS app.js
  JS app.js views
  JS footer.ejs
  JS index.ejs
  JS register.ejs
  JS login.ejs
  JS public
  JS views
  JS dataHabitat.ejs
  JS dataikan.ejs
  JS dataSpecies.ejs
  JS dataUsers.ejs
  JS editHabitat.ejs
  JS editikan.ejs
  JS editSpecies.ejs
  JS editUser.ejs
  JS footer.ejs
  JS header.ejs
  JS index.ejs
  JS login.ejs
  JS register.ejs
  JS app.js
  JS package-lock.json
  JS package.json
  JS OUTLINE
  JS TIMELINE
  JS user_management
  JS database_penilaian

JS app.js
354 // Route untuk memperbarui spesies
355 app.post("/species/update/:id", (req, res) => {
356   const speciesId = req.params.id;
357   const { nama_spesies } = req.body;
358   db.query(
359     "UPDATE spesies SET nama_spesies = ? WHERE id = ?",
360     [nama_spesies, speciesId],
361     (error) => {
362       if (error) {
363         return res.status(500).send(error);
364       }
365       res.redirect("/species");
366     }
367   );
368 });
369
370 // Route untuk menampilkan daftar habitat
371 app.get("/habitat", isAuthenticated, (req, res) => {
372   db.query("SELECT * FROM habitat", (error, habitatResults) => {
373     if (error) {
374       console.error("Error fetching habitat data:", error);
375     }
376   });
377 });
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

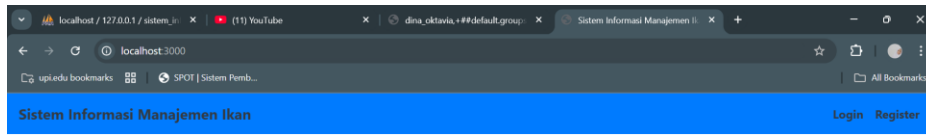


➤ MODUL PADA PENCATATAN IKAN



Pada modul view, terdapat file (data habitat, data ikan, data spesies, data users, edit habitat, edit ikan, edit spesies, edit user, footer, header, index, login, dan register).

TAMPILAN WEBSITE



Selamat Datang di Sistem Informasi Manajemen Ikan

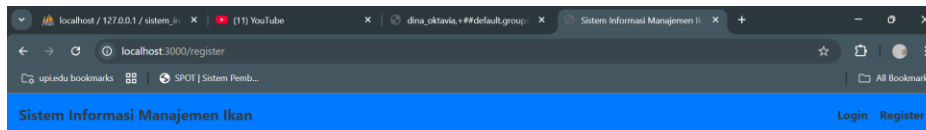
Sistem ini dirancang untuk membantu dalam mengelola data ikan, meliputi informasi spesies, lokasi, dan data penjualan. Aplikasi ini memberikan akses yang mudah bagi pengguna untuk memantau dan mengelola informasi terkait ikan secara efektif.

Testimoni Pengguna

"Sistem ini sangat membantu dalam pengelolaan data ikan saya!"

— Ahmad, Pemilik Toko Ikan

© 2024 Sistem Informasi Manajemen Ikan By : Silvia Isti Lestary



Register

Username:

Email:

Password:

Register

© 2024 Sistem Informasi Manajemen Ikan By : Silvia Isti Lestary



Login

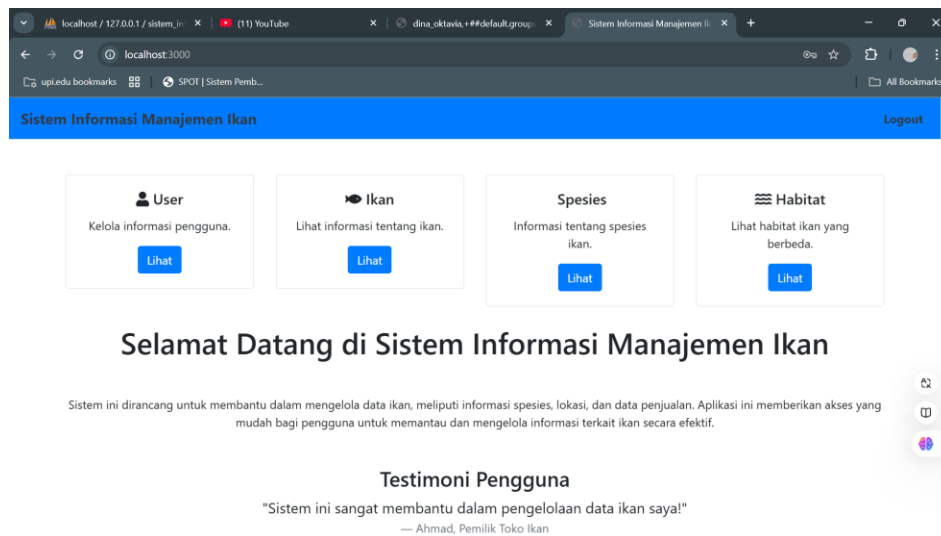
Email:

Password:

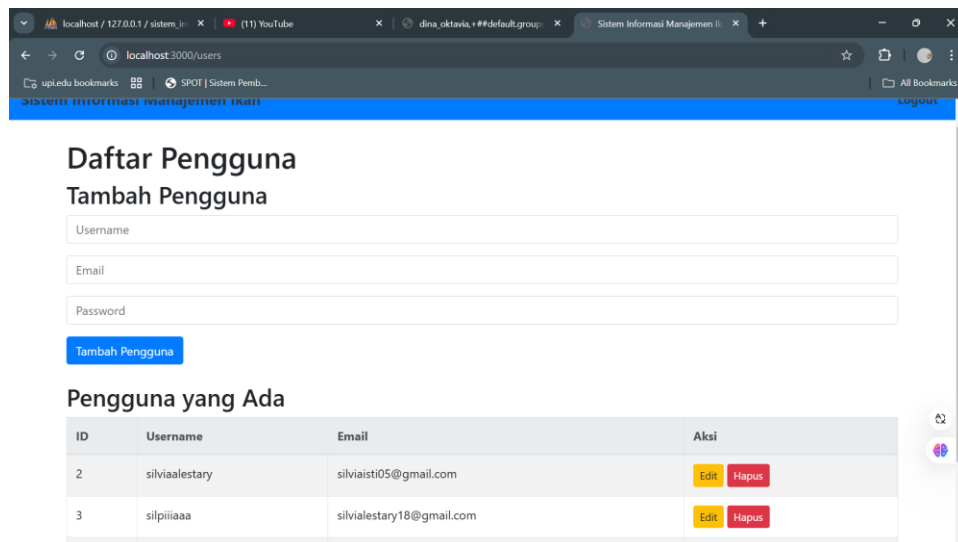
Login

© 2024 Sistem Informasi Manajemen Ikan By : Silvia Isti Lestary

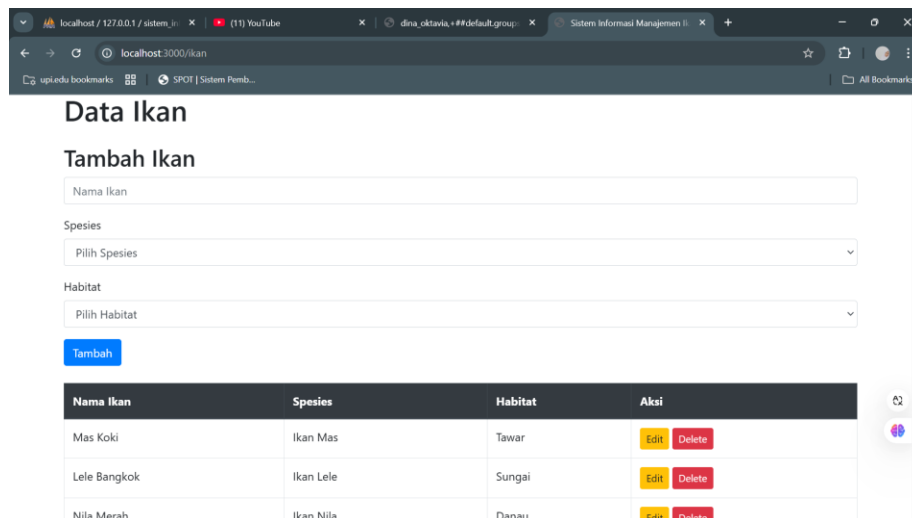
localhost:3000/login



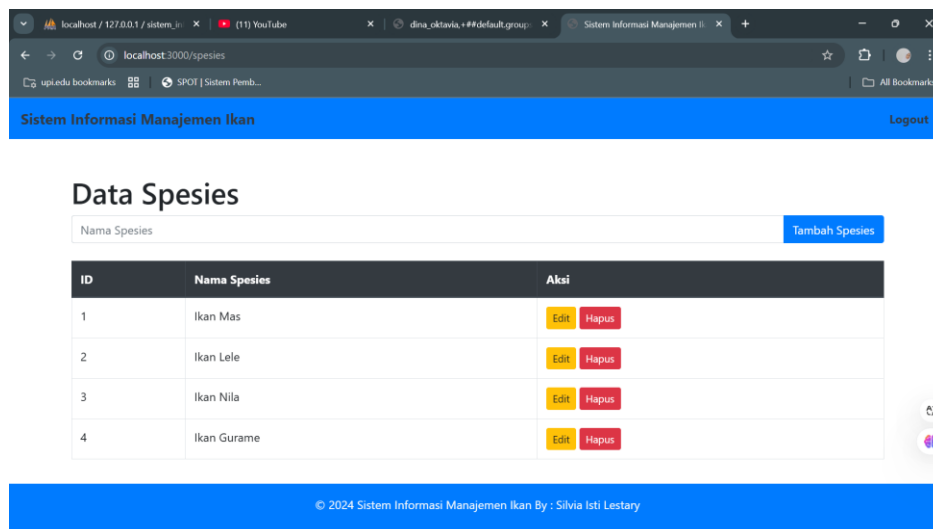
➤ Tampilan Halaman User



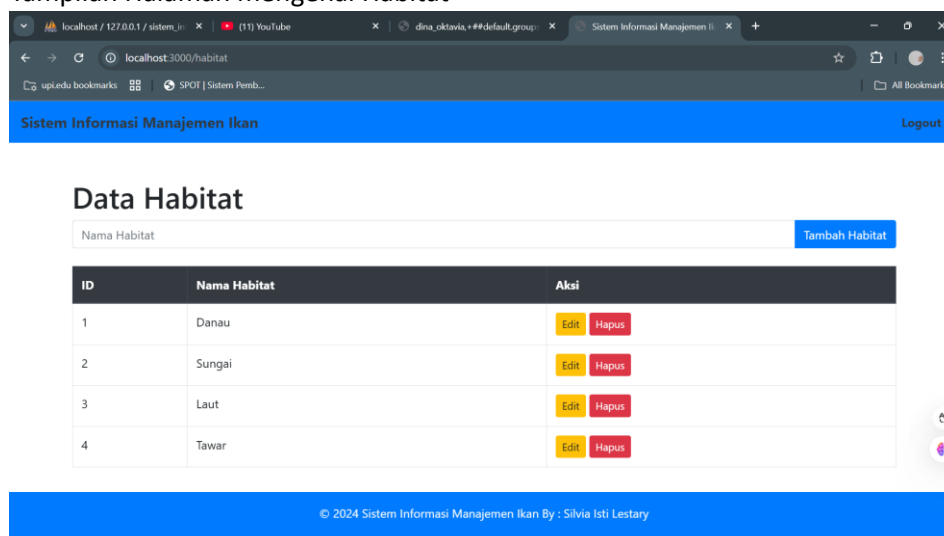
➤ Tampilan Halaman Tabel Ikan



➤ Tampilan Halaman mengenai Spesies



➤ Tampilan Halaman mengenai Habitat



➤ PANDUAN PENGGUNAAN

1. Buka folder “pencatatan ikan” di visual studio code
2. Pilih new terminal
3. Masukan “node app.js”
4. Klik <http://localhost:3000> kemudian, akan langsung masuk ke website tersebut
5. Klik register, buat akun terlebih dahulu “username, email, dan password”
6. Kemudian akan diarahkan ke halaman login, masukan lagi email dan password
7. Pada halaman selanjutnya, terdapat halaman dengan fitur fitur tertentu, pengguna dapat memilih form mana yang akan dituju
8. Pengguna dapat melakukan tambahan data, edit data, update data, maupun hapus data.
9. Setelah selesai memakai website tersebut, pengguna dapat logout dari web.
10. Lalu, akan kembali ke halaman awal.