

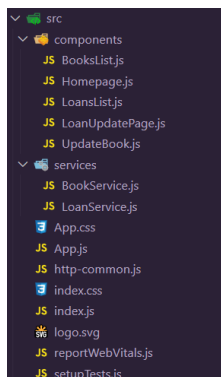
BAB 4

Tahapan Pembuatan Front-end

Front-end merupakan bagian dari aplikasi yang memuat sektor tampilan dari sebuah aplikasi. *Front-end* bisa juga disebut sebagai *client* atau penerima layanan dari *back-end* atau *server-side*, pada bagian ini kumpulan data-data akan ditampilkan berdasarkan perangkat yang digunakan (Liu & Gupta, 2019). Pada bagian ini akan diarahkan bagaimana cara kita untuk membuat bagian dari penampilan aplikasi agar data yang ditampilkan dapat mudah dimengerti dan mempermudah manajemen data melewati integrasi antara *server-side* dan *client*. Berikut adalah tahapan-tahapan yang dilewati untuk membuat *client* pada buku tutorial ini.

4.1. Pembuatan Front-End Buku

1. Menyiapkan Struktur Folder dan File



Gambar 1 Struktur Front-end

Hal yang harus dipersiapkan pertama kali adalah menyiapkan terlebih dahulu Folder dan File dalam project React yang sudah dibuat pada tahap

sebelumnya. Hal ini dilakukan untuk mempermudah pada saat akan memasukan kode agar lebih terurut.

2. Menyiapkan *dependency*

```
npm install axios  
npm install bootstrap  
npm install react-router-dom@5.2.0
```

Kolom 1 Install Dependency

Sebelum memasuki pembuatan kode lakukan instalasi *dependency* seperti pada Kolom 3 terlebih dahulu, agar dapat memenuhi kebutuhan komponen yang dibutuhkan dalam membuat aplikasi. Kegunaan dari *dependency* tersebut adalah yang pertama Axios digunakan untuk berkomunikasi dengan API berdasarkan URL yang diberikan, lalu bootstrap digunakan untuk mempercantik tampilan dari website, dan react-router-dom digunakan untuk memberikan rute terhadap aplikasi kepada modul-modul yang sudah dibuat agar dapat diakses.

3. Menyiapkan Koneksi API

```
import axios from "axios";  
  
export default axios.create({  
  baseURL: "http://localhost:3000/api",  
  headers: {  
    "Content-type": "application/json"  
  }  
});
```

Kolom 2 Persiapan http-common.js

Proses selanjutnya adalah menyiapkan file “http-common.js” seperti pada Kolom 4. *Import* axios dibutuhkan karena akan berkomunikasi dengan API dengan mengambil BaseURL dan header untuk menentukan tipe header dari

output API.

4. Menyiapkan Rute API

```
import http from "../http-common";

const getAll = () => {
  return http.get("/books");
};

const get = (id) => {
  return http.get(`/books/${id}`);
};

const create = (data) => {
  return http.post("/books", data);
};

const update = (id, data) => {
  return http.put(`/books/${id}`, data);
};

const remove = (id) => {
  return http.delete(`/books/${id}`);
};

const removeAll = () => {
  return http.delete(`/books`);
};

const findByTitle = (title) => {
  return http.get(`/books?title=${title}`);
};
```

Kolom 3 Rute Untuk HTTP Method

Pada Kolom 5 merupakan sekumpulan *function* yang digunakan untuk memberikan rute tambahan pada Base URL yang ada pada “http-common.js”. Beberapa rute digunakan untuk mengambil data dengan GET method, PUT method untuk mengubah data, POST method untuk menambahkan data, dan DELETE method untuk menghapus data.

```
const BookService = {
  getAll,
  get,
  create,
  update,
  remove,
  removeAll,
  findByTitle,
};

export default BookService;
```

Kolom 4 constraint BookService

Pada Kolom 6 *constraint* BookService digunakan untuk menampung semua rute yang sudah dibuat dengan mendefinisikannya kembali.

5. Membuat Kode Untuk Tampilan Book List

```
import React, { useState, useEffect } from "react";
import BookService from "../services/BookService";
import { Link } from "react-router-dom";

const BooksList = () => {
  const [books, setBooks] = useState([]);
  const [currentBook, setCurrentBook] = useState(null);
  const [judul, setJudul] = useState("");
  const [penulis, setPenulis] = useState("");
  const [penerbit, setPenerbit] = useState("");
  const [tanggalRilis, setTanggalRilis] = useState("");

  useEffect(() => {
    retrieveBooks();
  }, []);
```

Kolom 5 Import dan State untuk Books

Pada Kolom 7 digunakan untuk *import* kebutuhan komponen yang dibutuhkan untuk melakukan manajemen *state* pada halaman tertentu. *useEffect* digunakan untuk memberikan efek terhadap komponen tertentu jika mengalami aksi tertentu seperti pada saat mengalami klik atau kondisi apapun, dan *useState* untuk manajemen kondisi dalam halaman tertentu.

```

const retrieveBooks = () => {
  BookService.getAll()
    .then((response) => {
      setBooks(response.data);
    })
    .catch((error) => {
      console.log(error);
    });
};

const saveBook = () => {
  const book = {
    Judul: judul,
    Penulis: penulis,
    Penerbit: penerbit,
    Tanggal_rilis: tanggalRilis,
  };

  BookService.create(book)
    .then((response) => {
      console.log(response.data);
      retrieveBooks();
    })
    .catch((error) => {
      console.log(error);
    });
};

const updateBook = () => {
  const updatedBook = {
    id: currentBook.ID,
    Judul: judul,
    Penulis: penulis,
    Penerbit: penerbit,
    Tanggal_rilis: tanggalRilis,
  };

  BookService.update(currentBook.ID, updatedBook)
    .then((response) => {
      console.log(response.data);
      retrieveBooks();
      setCurrentBook(null);
      setJudul("");
      setPenulis("");
      setPenerbit("");
      setTanggalRilis("");
    })
    .catch((error) => {
      console.log(error);
    });
};

```

Pada Kolom 8 Merupakan sekumpulan *function* yang masing-masing digunakan untuk melakukan pengambilan atau manipulasi terhadap data. *Function* “retrieveBook” digunakan untuk mengambil semua data buku yang diambil oleh API, “saveBook” digunakan untuk menyimpan data buku dengan method POST, dan “updateBook” digunakan untuk mengubah data dari buku dengan mengambil ID dari buku tersebut terlebih dahulu dalam *state* management.

```
const handleInputChange = (event) => {
  const { name, value } = event.target;
  if (name === "judul") {
    setJudul(value);
  } else if (name === "penulis") {
    setPenulis(value);
  } else if (name === "penerbit") {
    setPenerbit(value);
  } else if (name === "tanggal_rilis") {
    setTanggalRilis(value);
  }
};
```

Kolom 7 State HandleInputChange

Pada Kolom 9 digunakan untuk manajemen kondisi pada saat kolom tertentu diubah.

```
return (
  <div className="container">
    <div className="row">
      <div className="col-md-6">
        <div className="mb-3">
          <label className="form-label">Judul:</label>
          <input
            type="text"
            className="form-control"
            name="judul"
            value={judul}
            onChange={handleInputChange}
          />
        </div>
      </div>
    </div>
  )
```

```

<div className="mb-3">
  <label className="form-label">Penulis:</label>
  <input
    type="text"
    className="form-control"
    name="penulis"
    value={penulis}
    onChange={handleInputChange}
  />
</div>

<div className="mb-3">
  <label className="form-label">Penerbit:</label>
  <input
    type="text"
    className="form-control"
    name="penerbit"
    value={penerbit}
    onChange={handleInputChange}
  />
</div>

<div className="mb-3">
  <label className="form-label">Tanggal Rilis:</label>
  <input
    type="date"
    className="form-control"
    name="tanggal_rilis"
    value={tanggalRilis}
    onChange={handleInputChange}
  />
</div>

<div className="mb-3">
  <button className="btn btn-primary me-2" onClick={saveBook}>
    Create
  </button>
</div>
</div>
</div>

```

Kolom 8 Form Code

Pada Kolom 10 digunakan untuk membuat tampilan form dari beberapa *field* berdasarkan model dari API. Manajemen *state* digunakan disini dengan melakukan pemasangan function dengan mendefinisikan properti pada *value* dan kondisi *onChange*

parameter dari buku menggunakan *function* “books” dijadikan alias lalu dipanggil parameternya pada masing-masing kolom.

6. Set Up Index dan App JS

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter } from "react-router-dom";

import App from "./App";
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById("root")
);

reportWebVitals();
```

Kolom 10 Pengaturan Index.js

Setelah pembuatan modul untuk tampilan dan integrasi sudah selesai, selanjutnya melakukan pengaturan pada Index JS seperti pada Kolom 12. Tambahkan *import* untuk “BrowserRouter” dan sisipkan pada render dan disimpan dengan “<App />” didalamnya.

```
import React from "react";
import { Switch, Route, Link } from "react-router-dom";
import "bootstrap/dist/css/bootstrap.min.css";
import "@fortawesome/fontawesome-free/css/all.css";
import "@fortawesome/fontawesome-free/js/all.js";
import "./App.css";

import BooksList from "./components/BooksList";
```

Kolom 11 Import Kebutuhan

Lakukan import terlebih dahulu seperti pada Kolom 13 untuk memanggil modul BookList dan memenuhi kebutuhan komponen agar dapat menjalankan modul

tertentu.

```
import UpdateBook from "../components/UpdateBook";
import LoansList from "../components/LoansList";
import LoanUpdatePage from "../components/LoanUpdatePage";
import Homepage from "../components/Homepage";

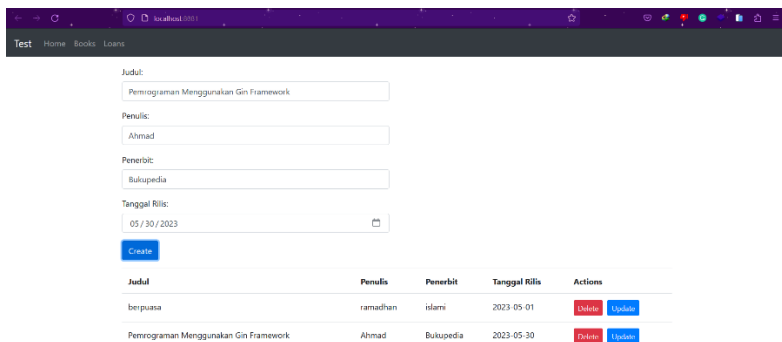
function App() {
  return (
    <div>
      <nav className="navbar navbar-expand navbar-dark bg-dark">
        <a href="/books" className="navbar-brand">
          Test
        </a>
        <div className="navbar-nav mr-auto">
          <li className="nav-item">
            <Link to={"/homepage"} className="nav-link">
              Home
            </Link>
          </li>
          <li className="nav-item">
            <Link to={"/books"} className="nav-link">
              Books
            </Link>
          </li>
          <li className="nav-item">
            <Link to={"/Loans"} className="nav-link">
              Loans
            </Link>
          </li>
        </div>
      </nav>

      <div className="container mt-3">
        <Switch>
          <Route exact path={['/', '/books']} component={BooksList} />
          <Route exact path="/loans" component={LoansList} />
          <Route exact path="/books/update/:id" component={UpdateBook} />
          <Route exact path="/loans/update/:id" component={LoanUpdatePage} />
        </Switch>
      </div>
    </div>
  );
}

export default App;
```

Lakukan *import* untuk modul lainnya dan buatlah rute menggunakan tag “react-router-dom” yaitu “Route path” dan masukkan URL pada setiap route, setelah pembuatan rute buat navbar agar dapat melakukan navigasi dengan mudah dalam aplikasi dengan menggunakan class navbar dan nav-item untuk kumpulan link dari rute yang sudah disiapkan.

7. Mencoba Pada Browser



Gambar 2 Tampilan BooksList

Setelah kode tampilan dibuat buka URL Localhost:8081, maka tampilan akan terlihat seperti pada Gambar 38 yaitu halaman manajemen buku yang dibuat dalam file “BooksList.js”.

8. Pembuatan Halaman UpdateBook

```
import React, { useState, useEffect } from "react";
import { useParams } from "react-router-dom";
import BookService from "../services/BookService";

const UpdateBook = () => {
  const { id } = useParams();
  const [judul, setJudul] = useState("");
  const [penulis, setPenulis] = useState("");
  const [penerbit, setPenerbit] = useState("");
  const [tanggalRilis, setTanggalRilis] = useState("");
```

Kolom 13 Import Dependency dan Pembuatan State

Pada Kolom 15 merupakan *import* untuk *dependency* yang dibutuhkan untuk menjalankan modul “UpdateBook.js”.

```
const retrieveBook = () => {
  BookService.get(id)
    .then((response) => {
      const { Judul, Penulis, Penerbit, Tanggal_rilis } = response.data;
      setJudul(Judul);
      setPenulis(Penulis);
      setPenerbit(Penerbit);
      setTanggalRilis(Tanggal_rilis);
    })
    .catch((error) => {
      console.log(error);
    });
};

const updateBook = () => {
  const updatedBook = {
    id: id,
    Judul: judul,
    Penulis: penulis,
    Penerbit: penerbit,
    Tanggal_rilis: tanggalRilis,
  };

  BookService.update(id, updatedBook)
    .then((response) => {
      console.log(response.data);
      // Redirect to the book list page after successful update
      window.Location.href = "/books";
    })
    .catch((error) => {
      console.log(error);
    });
};

const handleInputChange = (event) => {
  const { name, value } = event.target;
  if (name === "judul") {
    setJudul(value);
  } else if (name === "penulis") {
    setPenulis(value);
  } else if (name === "penerbit") {
    setPenerbit(value);
  } else if (name === "tanggal_rilis") {
    setTanggalRilis(value);
  }
};
```

Kolom 14 Pembuatan Function State

Selanjutnya pada Kolom 16 merupakan pembuatan *function* untuk mendeklarasikan state dan menampung buku berdasarkan ID menggunakan

function "retrieveBook". Lalu pembuatan *function* untuk menampung *field* pada buku, dan terakhir ada *handler* untuk perubahan pada input yang didefinisikan dengan "handleInputChange".

```
return (
  <div className="container">
    <h1>Update Book</h1>
    <div className="mb-3">
      <label className="form-label">Judul:</label>
      <input
        type="text"
        className="form-control"
        name="judul"
        value={judul}
        onChange={handleInputChange}
      />
    </div>

    <div className="mb-3">
      <label className="form-label">Penulis:</label>
      <input
        type="text"
        className="form-control"
        name="penulis"
        value={penulis}
        onChange={handleInputChange}
      />
    </div>

    <div className="mb-3">
      <label className="form-label">Penerbit:</label>
      <input
        type="text"
        className="form-control"
        name="penerbit"
        value={penerbit}
        onChange={handleInputChange}
      />
    </div>
  </div>
)
```

```

<div className="mb-3">
  <label className="form-label">Tanggal Rilis:</label>
  <input
    type="text"
    className="form-control"
    name="tanggal_rilis"
    value={tanggalRilis}
    onChange={handleInputChange}
  />
</div>

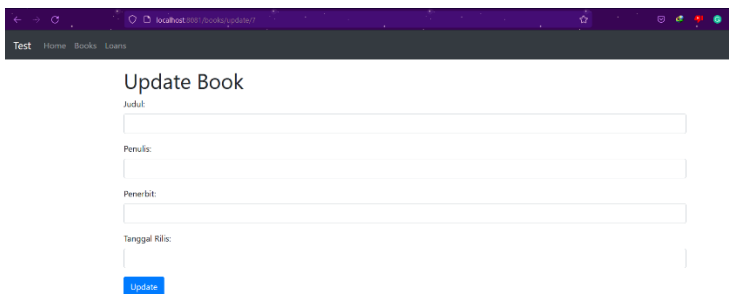
<div>
  <button className="btn btn-primary" onClick={updateBook}>
    Update
  </button>
</div>
</div>
);
};

export default UpdateBook;

```

Kolom 15 Code Interface UpdateBook

Pada Kolom 17 merupakan kode untuk tampilan Update Book dengan menggunakan bootstrap dan pendeklarasian HandlerInputChange pada tiap kolom dan pemberian function dalam kondisi “onClick” maka akan melakukan update terhadap buku.



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/books/update/1'. The browser's navigation bar includes 'Test', 'Home', 'Books', and 'Loans'. The main content area is titled 'Update Book' and contains four text input fields labeled 'Judul:', 'Penulis:', 'Penerbit:', and 'Tanggal Rilis:'. Below these fields is a blue button labeled 'Update'.

Gambar 3 Tampilan Update Book

Pada Gambar 39 merupakan tampilan dari UpdateBook, yang berfungsi untuk mengubah data buku berdasarkan ID yang diambil dari URL

4.2. Pembuatan Front-End Peminjaman Buku

1. Import dan Pembuatan State

```
import React, { useState, useEffect } from "react";
import LoanService from "../services/LoanService";
import { Link } from "react-router-dom";

const LoansList = () => {
  const [loans, setLoans] = useState([]);
  const [currentLoan, setCurrentLoan] = useState(null);
  const [userID, setUserID] = useState("");
  const [bookID, setBookID] = useState("");
  const [tanggalPinjam, setTanggalPinjam] = useState("");
  const [dikembalikan, setDikembalikan] = useState(false);

  useEffect(() => {
    retrieveLoans();
  }, []);

  const retrieveLoans = () => {
    LoanService.getAll()
      .then((response) => {
        setLoans(response.data);
      })
      .catch((error) => {
        console.log(error);
      });
  };
};
```

Kolom 16 Import Untuk LoansList

Import *dependency* terlebih dahulu seperti pada Kolom 18 agar dapat menjalankan program. Lalu buat State dengan menggunakan function `useState` untuk menyimpan kondisi dan `useEffect` untuk memberikan efek dalam memberikan aksi dan function `retrieveLoans` digunakan untuk mengambil data-data pinjaman.

```

const saveLoan = () => {
  const loan = {
    UserID: userID,
    BooksID: bookID,
    Tanggal_pinjam: tanggalPinjam,
    Dikembalikan: dikembalikan
  };

  LoanService.create(loan)
    .then((response) => {
      console.log(response.data);
      retrieveLoans();
    })
    .catch((error) => {
      console.log(error);
    });
};

const deleteLoan = (id) => {
  LoanService.remove(id)
    .then((response) => {
      console.log(response.data);
      retrieveLoans();
    })
    .catch((error) => {
      console.log(error);
    });
};

const removeAllLoans = () => {
  LoanService.removeAll()
    .then((response) => {
      console.log(response.data);
      retrieveLoans();
    })
    .catch((error) => {
      console.log(error);
    });
};

const handleInputChange = (event) => {
  const { name, value } = event.target;
  if (name === "userID") {
    setUserID(parseInt(value));
  } else if (name === "bookID") {
    setBookID(parseInt(value));
  } else if (name === "tanggalPinjam") {
    setTanggalPinjam(value);
  } else if (name === "dikembalikan") {
    setDikembalikan(event.target.checked);
  }
};

```

Kolom 17 Kumpulan Function

2. Membuat Tampilan dan Assign Function

Selanjutnya adalah menambahkan *function* seperti pada Kolom 19, *function* tersebut berfungsi untuk menyimpan buku pada saveLoan dengan mengambil input dari form yang dideklarasikan pada tahapan selanjutnya. Lalu Remove All Loans digunakan untuk menghapus semua data pinjaman buku dan handleInputChange berfungsi untuk merekam kondisi apabila ada perubahan dalam sebuah form.

```
return (
  <div className="container">
    <div className="row">
      <div className="col-md-6">
        <div className="mb-3">
          <label className="form-label">User ID:</label>
          <input
            type="number"
            className="form-control"
            name="userID"
            value={userID}
            onChange={handleInputChange}
          />
        </div>

        <div className="mb-3">
          <label className="form-label">Book ID:</label>
          <input
            type="number"
            className="form-control"
            name="bookID"
            value={bookID}
            onChange={handleInputChange}
          />
        </div>

        <div className="mb-3">
          <label className="form-label">Tanggal Pinjam:</label>
          <input
            type="date"
            className="form-control"
            name="tanggalPinjam"
            value={tanggalPinjam}
            onChange={handleInputChange}
          />
        </div>
      </div>
    </div>
  </div>
)
```

```

<div className="mb-3 form-check">
  <input
    type="checkbox"
    className="form-check-input"
    name="dikembalikan"
    checked={dikembalikan}
    onChange={handleInputChange}
  />
  <label className="form-check-label">Dikembalikan</label>
</div>

<div className="mb-3">
  <button className="btn btn-primary me-2" onClick={saveLoan}>
    Create
  </button>
</div>
</div>
</div>

<div className="row">
  <div className="col-md-12">
    <table className="table">
      <thead>
        <tr>
          <th>User ID</th>
          <th>Book ID</th>
          <th>Tanggal Pinjam</th>
          <th>Dikembalikan</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {loans.map((loan) => (
          <tr key={loan.ID}>
            <td>{loan.UserID}</td>
            <td>{loan.BooksID}</td>
            <td>{loan.Tanggal_pinjam}</td>
            <td>{loan.Dikembalikan ? "Yes" : "No"}</td>
            <td>
              <button
                className="btn btn-danger btn-sm me-2"
                onClick={() => deleteLoan(loan.ID)}
              >
                Delete
              </button>
              &nbsp;&nbsp;&nbsp;
              <Link
                to={` /loans/update/${loan.ID}`}
                className="btn btn-primary btn-sm"
              >
                Update
              </Link>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
</div>
</div>
);
});

export default LoansList;

```

Kolom 18 Code untuk Tampilan Tambah Pinjaman

Terakhir tambahkan bagian struktur HTML untuk menampilkan data-

data dari pinjaman. Data akan ditampilkan dalam bentuk tabel dan kondisi akan disimpan dalam masing-masing label, untuk tampilan dari Update pinjaman akan memiliki halaman tersendiri yang akan dijelaskan dalam tahap selanjutnya.

```
import React, { useState, useEffect } from "react";
import LoanService from "../services/LoanService";

const LoanUpdatePage = (props) => {
  const [loan, setLoan] = useState(null);
  const [userID, setUserID] = useState("");
  const [bookID, setBookID] = useState("");
  const [tanggalPinjam, setTanggalPinjam] = useState("");
  const [dikembalikan, setDikembalikan] = useState(false);

  useEffect(() => {
    retrieveLoan();
  }, []);

  const retrieveLoan = () => {
    const loanId = props.match.params.id;
    LoanService.get(loanId)
      .then((response) => {
        setLoan(response.data);
        setUserID(response.data.UserID);
        setBookID(response.data.BooksID);
        setTanggalPinjam(response.data.Tanggal_pinjam);
        setDikembalikan(response.data.Dikembalikan);
      })
      .catch((error) => {
        console.log(error);
      });
  };

  const updateLoan = () => {
    const updatedLoan = {
      id: loan.id,
      UserID: userID,
      BooksID: bookID,
      Tanggal_pinjam: tanggalPinjam,
      Dikembalikan: dikembalikan,
    };

    LoanService.update(loan.id, updatedLoan)
      .then((response) => {
        console.log(response.data);
        props.history.push("/loans");
      })
      .catch((error) => {
        console.log(error);
      });
  };
};
```

Kolom 19 Pendeklarasian Function Tambah Pinjaman

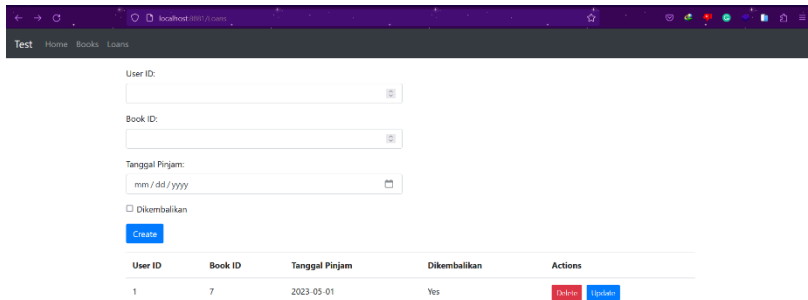
Tahap selanjutnya adalah pembuatan tampilan untuk UpdateLoans yang diawali dengan import dan pembuatan function untuk menyimpan *state*

dari halaman dan pendeklarasian *field* pada tiap *function*.

```
const handleInputChange = (event) => {
  const { name, value } = event.target;
  if (name === "userID") {
    setUserID(value);
  } else if (name === "bookID") {
    setBookID(value);
  } else if (name === "tanggalPinjam") {
    setTanggalPinjam(value);
  } else if (name === "dikembalikan") {
    setDikembalikan(event.target.checked);
  }
};

return (
  <div className="container">
    <h1>Update Loan</h1>
    {loan ? (
      <div>
        <div className="mb-3">
          <label className="form-label">User ID:</label>
          <input
            type="text"
            className="form-control"
            name="userID"
            value={userID}
            onChange={handleInputChange}
          />
        </div>
        <div className="mb-3">
          <label className="form-label">Book ID:</label>
          <input
            type="text"
            className="form-control"
            name="bookID"
            value={bookID}
            onChange={handleInputChange}
          />
        </div>
        <div className="mb-3">
          <label className="form-label">Tanggal Pinjam:</label>
          <input
            type="date"
            className="form-control"
            name="tanggalPinjam"
            value={tanggalPinjam}
            onChange={handleInputChange}
          />
        </div>
      </div>
    ) : null}
  </div>
);
```


tag HTML yang terdapat pada setiap struktur form masing-masing.



User ID:

Book ID:

Tanggal Pinjam:

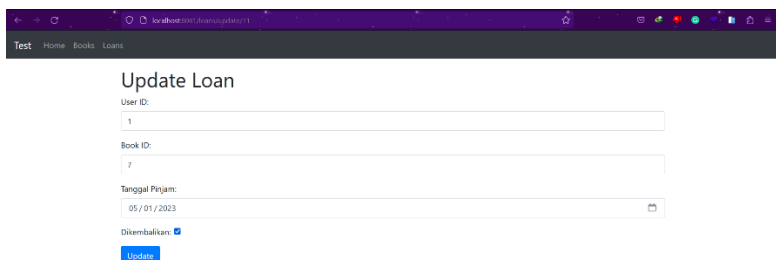
☐ Dikembalikan

Create

User ID	Book ID	Tanggal Pinjam	Dikembalikan	Actions
1	7	2023-05-01	Yes	Delete Update

Gambar 4 Tampilan Tambah Pinjaman

Pada Gambar 40 merupakan tampilan dari halaman peminjaman buku yang dapat diakses dengan menekan tombol Loans pada Navbar atau pada URL “localhost:8081/loans”. Field UserID dan BookID akan ditampilkan dalam bentuk tabel dan dalam tabel akan ada tombol aksi yang digunakan untuk menghapus data pinjaman buku.



Update Loan

User ID:

Book ID:

Tanggal Pinjam:

☒ Dikembalikan

Update

Gambar 5 Update Loan

Gambar 41 adalah tampilan dari UpdateLoan yang digunakan untuk mengubah data pinjaman. Halaman ini dapat diakses dengan menekan tombol “update” dalam halaman list pinjaman dan pinjaman akan ditampilkan

berdasarkan id yang dimuat dalam URL.