LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL XIII

Node.js: Pengenalan

Oleh:

(Izzaty Zahara Br Barus) (2311104052)

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

1.1 Dasar Teori

Node.js adalah platform runtime berbasis JavaScript yang berjalan di server. Dikembangkan oleh Ryan Dahl pada tahun 2009, Node.js memungkinkan penggunaan JavaScript di sisi server, di luar lingkungan browser, yang sebelumnya lebih dominan untuk pengembangan front-end. Node.js menggunakan mesin JavaScript V8 milik Google yang diintegrasikan dengan lingkungan runtime di server. Teknologi ini memungkinkan JavaScript untuk berjalan di luar browser dan dieksekusi dengan performa tinggi. Node.js menerapkan model event-driven dan non-blocking I/O, yang artinya sistem dapat menangani banyak permintaan secara bersamaan tanpa harus menunggu satu permintaan selesai sebelum memproses yang lain. Node.js beroperasi dalam satu thread utama, tetapi karena mendukung asynchronous programming, server dapat menangani ribuan permintaan simultan tanpa harus membuat banyak thread seperti dalam model multi-threaded tradisional. Hal ini membuat Node.js sangat efisien untuk aplikasi yang memerlukan operasi I/O tinggi, seperti aplikasi web real-time dan API. Node.js didukung oleh ekosistem package yang sangat besar yang dikenal sebagai NPM (Node Package Manager). NPM memudahkan pengembang untuk mengelola dependensi proyek dan menggunakan modul yang sudah tersedia dalam komunitas. Ribuan modul yang dikembangkan oleh komunitas dapat diinstal dan digunakan dengan cepat, mempercepat pengembangan aplikasi. Node.js memiliki keunggulan unik karena digunakan oleh banyak pengembang front-end yang menulis JavaScript untuk browser kini dapat menulis kode di sisi server tanpa perlu mempelajari bahasa yang sama sekali berbeda.

1.2 Tujuan

Adapun tujuan dari pelaksanaan praktikum ini adalah sebagai berikut:

1. Untuk memahami konsep dasar RESTful API menggunakan Node.js dan Express.
2. Untuk mengimplementasikan operasi **CRUD (Create, Read, Update, Delete)** pada sebuah aplikasi berbasis web.
3. Untuk menghubungkan aplikasi backend Node.js dengan database **MySQL**.
4. Untuk melatih penggunaan metode HTTP seperti **GET, POST, PUT, dan DELETE**.
5. Untuk memahami pemisahan antara **frontend (HTML, CSS, JavaScript)** dan **backend (Node.js)**.

6. Untuk membangun aplikasi sederhana pengelolaan data mahasiswa berbasis REST API.
7. Untuk meningkatkan pemahaman mahasiswa mengenai komunikasi data antara client dan server.

1.3 Manfaat

Manfaat yang diperoleh dari pelaksanaan praktikum ini antara lain:

1. Mahasiswa dapat memahami cara kerja RESTful API dalam pengembangan aplikasi web.
2. Mahasiswa mampu membuat dan mengelola server backend menggunakan Node.js dan Express.
3. Mahasiswa dapat mengintegrasikan database MySQL dengan aplikasi backend.
4. Mahasiswa memperoleh pengalaman langsung dalam membangun aplikasi CRUD berbasis web.
5. Mahasiswa mampu mengimplementasikan konsep client-server secara nyata.
6. Mahasiswa dapat mengembangkan keterampilan pemrograman backend yang dibutuhkan dalam dunia kerja.
7. Mahasiswa menjadi lebih terampil dalam melakukan debugging dan pengujian API.

BAB II

HASIL PRAKTIKUM

2.1 Unguided

2.1.1 Soal 1

1. Syntax

app.js

```
const express = require("express");
const cors = require("cors");
const path = require("path");
const crud = require("./crud");

const app = express();
app.use(cors());
app.use(express.json());
```

```
// 👉 INI YANG PENTING
app.use(express.static(path.join(__dirname,
"public")));

const PORT = 3000;

// CREATE
app.post("/mahasiswa", (req, res) => {
  const { nama, nim, jurusan, email } = req.body;
  crud.createMahasiswa(nama, nim, jurusan, email, err
=> {
    if (err) return res.status(500).send(err);
    res.send("Mahasiswa ditambahkan");
  });
});

// READ
app.get("/mahasiswa", (req, res) => {
  crud.getAllMahasiswa((err, rows) => {
    if (err) return res.status(500).send(err);
    res.json(rows);
  });
});

// UPDATE
app.put("/mahasiswa/:id", (req, res) => {
  const { id } = req.params;
  const { nama, nim, jurusan, email } = req.body;
  crud.updateMahasiswa(id, nama, nim, jurusan, email,
err => {
    if (err) return res.status(500).send(err);
    res.send("Mahasiswa diupdate");
  });
});

// DELETE
```

```
app.delete("/mahasiswa/:id", (req, res) => {
  crud.deleteMahasiswa(id, err => {
    if (err) return res.status(500).send(err);
    res.send("Mahasiswa dihapus");
  });
});

app.listen(PORT, () => {
  console.log(`🚀 Server berjalan di
http://localhost:${PORT}`);
});
```

crud.js

```
const db = require("./db");

// CREATE
function createMahasiswa(nama, nim, jurusan, email,
cb) {
  db.query(
    "INSERT INTO mahasiswa (nama,nim,jurusan,email)
VALUES (?,?,?,?)",
    [nama, nim, jurusan, email],
    cb
  );
}

// READ
function getAllMahasiswa(cb) {
  db.query("SELECT * FROM mahasiswa", cb);
}

// UPDATE
function updateMahasiswa(id, nama, nim, jurusan,
email, cb) {
  db.query(
    "UPDATE mahasiswa SET nama=?, nim=?, jurusan=?,
email=? WHERE id=?",
```

```
    [nama, nim, jurusan, email, id],
    cb
  );
}

// DELETE
function deleteMahasiswa(id, cb) {
  db.query("DELETE FROM mahasiswa WHERE id=?", [id],
cb);
}

module.exports = {
  createMahasiswa,
  getAllMahasiswa,
  updateMahasiswa,
  deleteMahasiswa,
};
```

db.js

```
const mysql = require("mysql");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "akademik",
});

connection.connect((err) => {
  if (err) {
    console.log("❌ Gagal koneksi database");
    console.log(err);
    return;
  }
  console.log("✅ Database terkoneksi");
});
```

```
module.exports = connection;
```

Package.json

```
{
  "name": "restfulapi",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit
1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^5.2.1",
    "mysql": "^2.18.1"
  }
}
```

public/index.html

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <title>CRUD Mahasiswa</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
```

```
<div class="container">
    <h1>Data Mahasiswa</h1>

    <form id="formMahasiswa">
        <input type="hidden" id="id">

        <input type="text" id="nama"
placeholder="Nama" required>
        <input type="text" id="nim" placeholder="NIM"
required>
        <input type="text" id="jurusan"
placeholder="Jurusan" required>
        <input type="email" id="email"
placeholder="Email" required>

        <button type="submit">Simpan</button>
    </form>

    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Nama</th>
                <th>NIM</th>
                <th>Jurusan</th>
                <th>Email</th>
                <th>Aksi</th>
            </tr>
        </thead>
        <tbody id="dataMahasiswa"></tbody>
    </table>
</div>

<script src="script.js"></script>
</body>
</html>
```

public/script.js

```
const API_URL = "http://localhost:3000/mahasiswa";

const form = document.getElementById("formMahasiswa");
const tbody =
document.getElementById("dataMahasiswa");

const idInput = document.getElementById("id");
const namaInput = document.getElementById("nama");
const nimInput = document.getElementById("nim");
const jurusanInput =
document.getElementById("jurusan");
const emailInput = document.getElementById("email");

function loadData() {
    fetch(API_URL)
        .then(res => res.json())
        .then(data => {
            tbody.innerHTML = "";
            data.forEach(m => {
                tbody.innerHTML += `
                <tr>
                    <td>${m.id}</td>
                    <td>${m.nama}</td>
                    <td>${m.nim}</td>
                    <td>${m.jurusan}</td>
                    <td>${m.email}</td>
                    <td>
                        <button class="btn-edit"
onclick="editData(${m.id}, '${m.nama}', '${m.nim}',
'${m.jurusan}', '${m.email}')">Edit</button>
                        <button class="btn-delete"
onclick="deleteData(${m.id})">Hapus</button>
                    </td>
                </tr>`;
            });
```

```
        });
}

form.addEventListener("submit", function (e) {
    e.preventDefault();

    const data = {
        nama: namaInput.value,
        nim: nimInput.value,
        jurusan: jurusanInput.value,
        email: emailInput.value
    };

    if (idInput.value === "") {
        // CREATE
        fetch(API_URL, {
            method: "POST",
            headers: { "Content-Type":
"application/json" },
            body: JSON.stringify(data)
        }).then(() => loadData());
    } else {
        // UPDATE
        fetch(`${API_URL}/${idInput.value}`, {
            method: "PUT",
            headers: { "Content-Type":
"application/json" },
            body: JSON.stringify(data)
        }).then(() => loadData());
    }

    form.reset();
    idInput.value = "";
});

function deleteData(id) {
    fetch(`${API_URL}/${id}`, {
```

```
        method: "DELETE"
    }).then(() => loadData());
}

function editData(id, nama, nim, jurusan, email) {
    idInput.value = id;
    namaInput.value = nama;
    nimInput.value = nim;
    jurusanInput.value = jurusan;
    emailInput.value = email;
}

loadData();
```

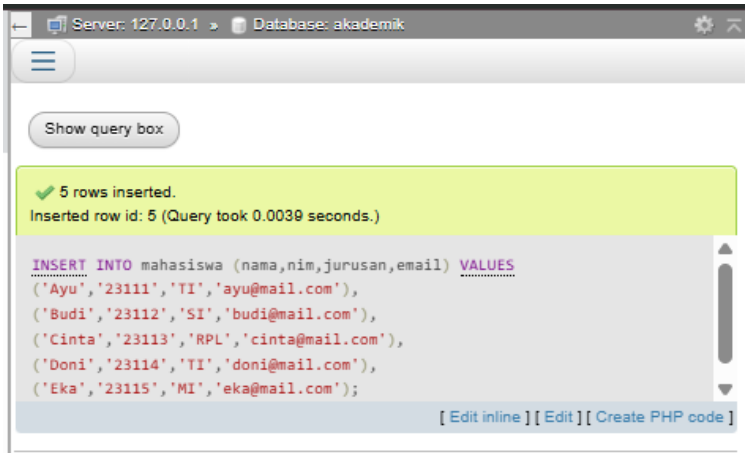
public/style.css

```
body {
    font-family: Arial, sans-serif;
    background: #f4f6f8;
}

.container {
    width: 800px;
    margin: 40px auto;
    background: #fff;
    padding: 20px;
    border-radius: 8px;
}

h1 {
    text-align: center;
}

form {
    display: grid;
    gap: 10px;
    margin-bottom: 20px;
```

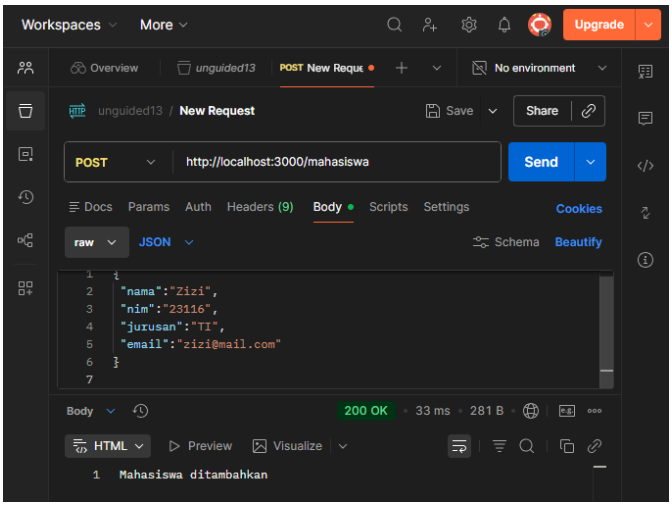
```
}

input {
    padding: 8px;
    font-size: 14px;
}
button {
    padding: 10px;
    background: #007bff;
    color: white;
    border: none;
    cursor: pointer;
}
button:hover {
    background: #0056b3;
}
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    border: 1px solid #ddd;
    padding: 8px;
    text-align: center;
}

th {
    background: #007bff;
    color: white;
}

.btn-delete {
    background: red;
    padding: 5px;
}

.btn-edit {
```

```
    background: orange;
    padding: 5px;
}
```

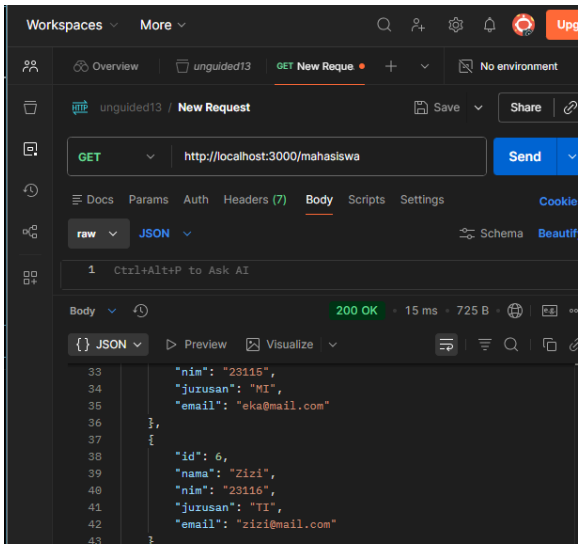
2. Output



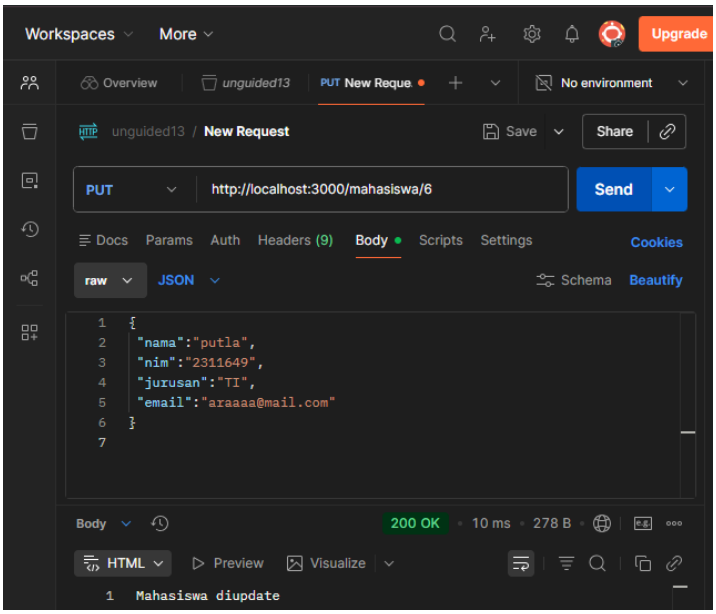
post


Workspaces
More
Upgrade
Overview
unguided13
POST New Reque
No environment
unguided13 / New Request
Save
Share
POST
http://localhost:3000/mahasiswa
Send
Docs Params Auth Headers (9) Body Scripts Settings
Cookies
raw
JSON
Schema
Beautify
{
"nama":"Zizi",
"nim":"23116",
"jurusan":"TI",
"email":"zizi@mail.com"
}
Body
200 OK
33 ms
281 B
HTML
Preview
Visualize
Mahasiswa ditambahkan

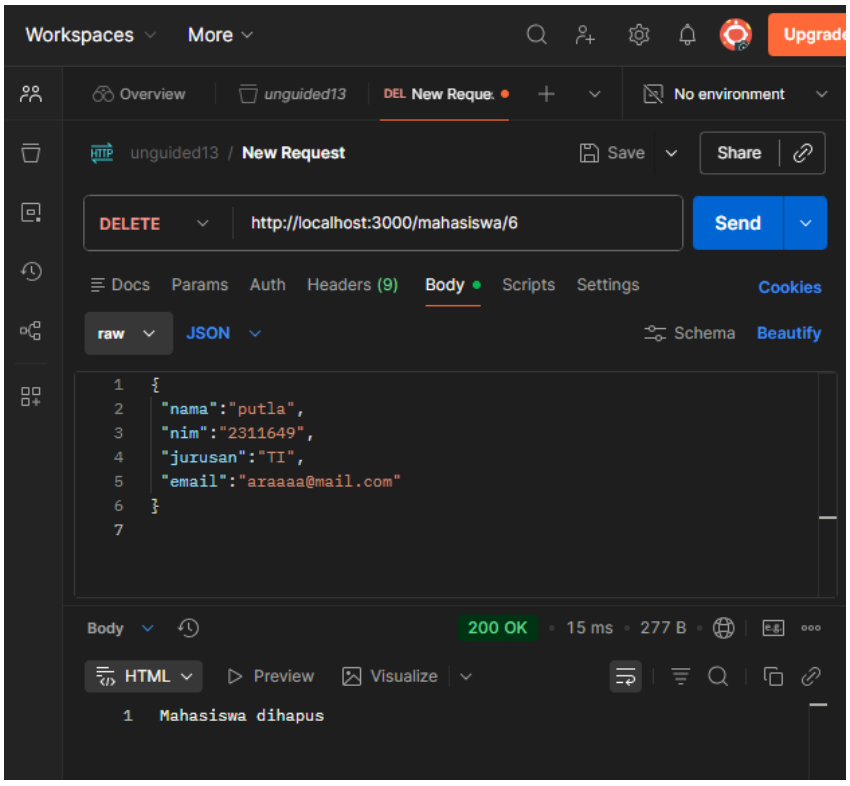
Get


Workspaces
More
Overview
unguided13
GET New Reque
No environment
unguided13 / New Request
Save
Share
GET
http://localhost:3000/mahasiswa
Send
Docs Params Auth Headers (7) Body Scripts Settings
raw
JSON
Schema
Ctrl+Alt+P to Ask AI
Body
200 OK
15 ms
725 B
JSON
Preview
Visualize
"nim": "23115",
"jurusan": "MI",
"email": "eka@mail.com"
},
{
"id": 6,
"nama": "Zizi",
"nim": "23116",
"jurusan": "TI",
"email": "zizi@mail.com"

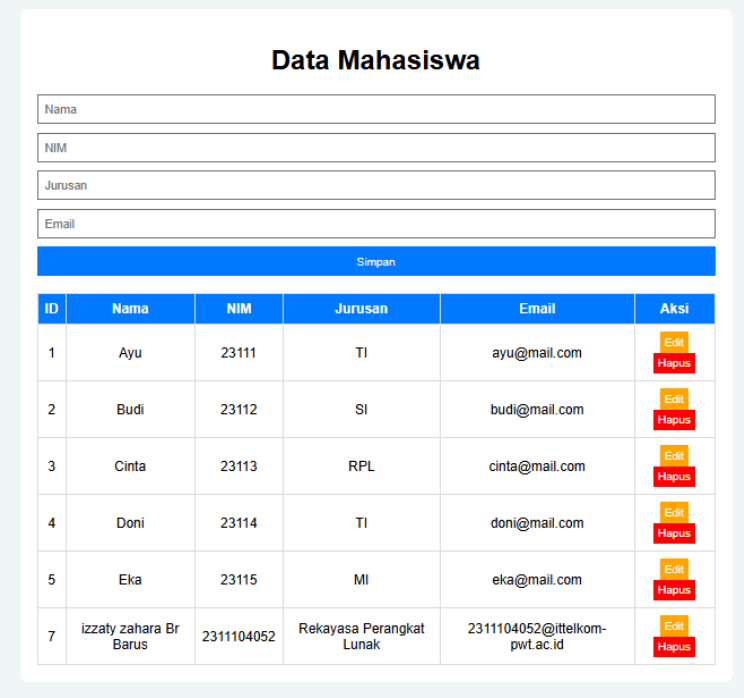
Put


Workspaces
More
Upgrade
Overview
unguided13
PUT New Reque
No environment
unguided13 / New Request
Save
Share
PUT
http://localhost:3000/mahasiswa/6
Send
Docs Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Schema Beautify
{
"nama":"putla",
"nim":"2311649",
"jurusan":"TI",
"email":"araaaa@mail.com"
}
Body
200 OK
10 ms
278 B
HTML
Preview
Visualize
Mahasiswa diupdate

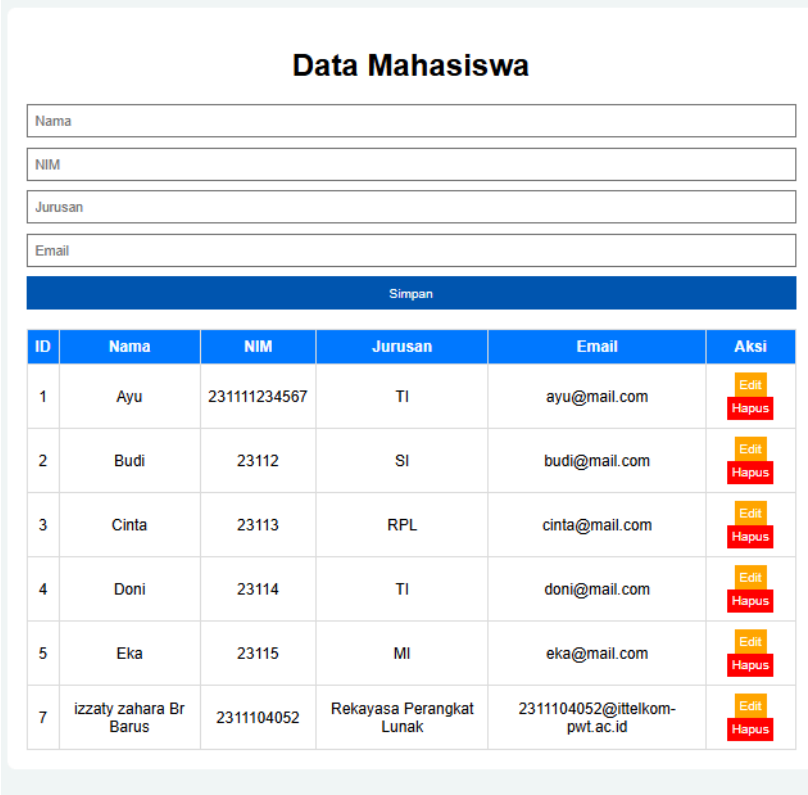
Delete


Workspaces
More
Upgrade
Overview
unguided13
DEL New Reque
No environment
unguided13 / New Request
Save
Share
DELETE
http://localhost:3000/mahasiswa/6
Send
Docs Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Schema Beautify
{
"nama":"putla",
"nim":"2311649",
"jurusan":"TI",
"email":"araaaa@mail.com"
}
Body
200 OK
15 ms
277 B
HTML
Preview
Visualize
Mahasiswa dihapus

Menambah data



Edit data



BAB III

KESIMPULAN & SARAN

3.1 Kesimpulan

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa pembuatan RESTful API menggunakan Node.js dan Express berhasil diimplementasikan dengan baik. Aplikasi yang dibangun mampu menjalankan operasi CRUD (Create, Read, Update, Delete) terhadap data mahasiswa yang terhubung dengan database MySQL. Selain itu, integrasi antara frontend berbasis HTML dan JavaScript dengan backend Node.js dapat berjalan dengan lancar melalui penggunaan metode HTTP yang sesuai. Praktikum ini memberikan pemahaman yang lebih mendalam mengenai konsep client-server serta cara kerja RESTful API dalam pengembangan aplikasi web modern.

3.2 Saran

Adapun saran yang dapat diberikan untuk pengembangan selanjutnya adalah agar aplikasi dapat ditingkatkan dengan menambahkan fitur keamanan seperti validasi input dan autentikasi pengguna. Selain itu, tampilan antarmuka dapat dikembangkan menggunakan framework frontend agar lebih interaktif dan responsif. Penggunaan struktur folder yang lebih modular serta penanganan error yang lebih baik juga disarankan agar aplikasi menjadi lebih mudah dikembangkan dan dipelihara di masa mendatang.