# **UNIVERSITAS GUNADARMA**

# **PRAKTIKUM PEMROGRAMAN WEB**



**MANUAL BOOK**

“Participant List”

Nama Anggota :

1. Daffa Ramzy Saputra (50422371)
2. Pradipa Wibi Herdani (51422293)

Kelas : 3IA08

Fakultas : Teknologi Industri

Jurusan : Teknik Informatika

PJ : Muhammad Rizqi Fajri

**Ditulis Guna Melengkapi Sebagian Syarat**

**Praktikum Pemrograaman Web**

**Universitas Gunadarma**

**2025**

# **DAFTAR ISI**

[**UNIVERSITAS GUNADARMA** 1](#_Toc187838499)

[**PRAKTIKUM PEMROGRAMAN WEB** 1](#_Toc187838500)

[**DAFTAR ISI** 2](#_Toc187838501)

[**BAB I** 3](#_Toc187838502)

[**PENDAHULUAN** 3](#_Toc187838503)

[**1.1 Latar Belakang** 3](#_Toc187838504)

[**1.2 Tujuan** 3](#_Toc187838505)

[**BAB II** 5](#_Toc187838506)

[**PEMBAHASAN** 5](#_Toc187838507)

[**2.1 Tahapan Pembuatan Website Game Catalog** 5](#_Toc187838508)

[**2.2 Teknologi yang Digunakan** 7](#_Toc187838509)

[**2.3 Keunggulan Website Katalog Game** 7](#_Toc187838510)

[**BAB III** 8](#_Toc187838511)

[**ANALISA DAN PERANCANGAN** 8](#_Toc187838512)

[**BAB IV** 9](#_Toc187838513)

[**PENUTUP** 9](#_Toc187838514)

[**4.1 Kesimpulan** 9](#_Toc187838515)

[**4.2 Saran** 9](#_Toc187838516)

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Dalam pengelolaan data, kebutuhan akan sistem yang mampu menyimpan, menampilkan, memperbarui, dan menghapus informasi menjadi sangat penting, terutama dalam lingkungan bisnis, pendidikan, dan pemerintahan. Salah satu solusi yang sering digunakan untuk memenuhi kebutuhan ini adalah pengembangan website berbasis CRUD (Create, Read, Update, Delete).

Website CRUD adalah aplikasi berbasis web yang memungkinkan pengguna untuk melakukan empat operasi utama dalam manajemen data. Operasi ini mencakup:

* Create – Penambahan data baru ke dalam sistem.
* Read – Penampilan data secara real-time untuk memenuhi kebutuhan informasi pengguna.
* Update – Pembaruan data yang telah ada untuk menjaga keakuratan dan relevansinya.
* Delete – Penghapusan data yang tidak lagi diperlukan.

Pengembangan website CRUD memberikan banyak manfaat, termasuk kemudahan dalam pengelolaan data, penghematan waktu, serta aksesibilitas dari berbagai perangkat yang terhubung ke internet. Website ini sering digunakan dalam berbagai skenario, seperti sistem manajemen pelanggan, aplikasi akademik, inventaris barang, dan platform administrasi internal.

Dalam perancangan sebuah website CRUD, penting untuk mempertimbangkan antarmuka pengguna yang intuitif, keamanan data yang terjamin, serta efisiensi sistem. Dengan teknologi modern seperti framework berbasis JavaScript, PHP, atau Python, pengembangan website CRUD dapat dilakukan dengan lebih cepat dan fleksibel sesuai kebutuhan pengguna.

Website CRUD yang baik tidak hanya mempermudah manajemen data, tetapi juga memberikan pengalaman pengguna yang lebih baik, meningkatkan produktivitas, serta mendukung tujuan organisasi atau individu dalam mencapai hasil yang maksimal.

### **1.2 Tujuan**

Dalam Tujuan kami dalam membuat situs web katalog game ini adalah: Tujuan utama kami dalam membuat website CRUD (Create, Read, Update, Delete) diantaranya:

* 1. Efisiensi dalam Pengelolaan Data
* Mempermudah proses penyimpanan, pembaruan, dan penghapusan data secara terstruktur dan sistematis.
* Mengurangi kesalahan manual dalam manajemen data.
  1. Penyediaan Akses Data yang Mudah
* Memberikan akses real-time kepada pengguna untuk melihat dan mengelola data kapan saja dan di mana saja.
* Memastikan data mudah dicari dan disajikan secara informatif melalui antarmuka pengguna.
  1. Meningkatkan Produktivitas
* Mengotomatiskan tugas-tugas yang berulang, sehingga pengguna dapat fokus pada pekerjaan yang lebih strategis.
* Menghemat waktu dengan proses yang lebih cepat dibandingkan dengan metode manual.
  1. Keamanan Data
* Menyediakan kontrol akses yang memungkinkan hanya pengguna yang berwenang untuk mengelola data.
* Menjamin integritas data dengan implementasi fitur validasi dan backup.
  1. Fleksibilitas dan Skalabilitas
* Mendukung pengelolaan data dalam berbagai skala, mulai dari database kecil hingga sistem besar yang melibatkan banyak pengguna.
* Memudahkan penyesuaian fitur berdasarkan kebutuhan bisnis atau pengguna.
  1. Meningkatkan Transparansi
* Menyediakan jejak audit atau log aktivitas untuk melacak perubahan pada data.
* Memastikan pengguna dapat mengetahui asal, perubahan, atau status data dengan mudah.
  1. Pengambilan Keputusan yang Lebih Baik
* Memberikan data yang terstruktur dan up-to-date sebagai dasar pengambilan keputusan.
* Mendukung analisis data melalui penyediaan fitur pencarian dan laporan.
  1. Dukungan Kolaborasi
* Memungkinkan beberapa pengguna bekerja bersama dalam sistem yang sama tanpa tumpang tindih.
* Memfasilitasi komunikasi dan koordinasi melalui data yang terpusat.

# **BAB II**

## **PEMBAHASAN**

### **2.1 Tahapan Pembuatan Website Participant List**

Website ini dibangun dengan memanfaatkan teknologi modern yang dibagi menjadi dua bagian utama, yaitu frontend dan backend:

#### **2.1.1 Frontend**

Frontend merupakan bagian antarmuka pengguna yang bertanggung jawab atas tampilan dan interaksi pengguna. Teknologi yang digunakan meliputi:

* React.js adalah library JavaScript yang digunakan untuk membangun antarmuka pengguna (User Interface/UI) secara efisien dan interaktif, terutama pada aplikasi web berbasis satu halaman (Single Page Applications).

#### **2.1.2 Backend**

Backend bertanggung jawab atas logika bisnis dan pengelolaan data. Teknologi yang digunakan meliputi:

* Node.js adalah runtime JavaScript yang berjalan di server, memungkinkan pengembang untuk menggunakan JavaScript untuk pengembangan backend. Dibangun di atas mesin V8 JavaScript Engine dari Google Chrome, Node.js menawarkan kinerja tinggi dan efisiensi.
* Express.js adalah framework berbasis Node.js yang dirancang untuk mempermudah pengembangan aplikasi web dan API. Express menyediakan antarmuka yang minimalis dan fleksibel, sehingga pengembang dapat membuat aplikasi dengan cepat tanpa harus menulis banyak kode boilerplate.
* MySQL adalah sistem manajemen basis data relasional (Relational Database Management System/RDBMS) berbasis SQL (Structured Query Language) yang digunakan untuk menyimpan, mengelola, dan mengambil data secara efisien.

### **2.2 Tahapan Pembuatan Website Participant List**

* Pengguna Menambahkan Data: Pengguna dapat menekan tombol Add New lalu di dalamnya pengguna bisa menambahkan nama, email, dan jenis kelamin.
* Pengguna menyimpan Data: Setelah selesai mengisikan data, pengguna bisa menekan save di menu tampilan Add New.
* Pengguna Mengubah Data: Selanjutnya pengguna bisa menekan tombol edit dan merubah data yang sudah ada untuk diperbaharui dengan mengisikan formulir yang disediakan dan menekan tombol Update.
* Pengguna Menghapus Data: Terakhir pengguna bisa menekan delete untuk menghapus sebuah data yang salah atau semisal peserta tersebut sudah tidak diperlukan lagi.

### **2.3 Fitur Utama**

* Pencatatan Daftar Peserta: Fitur ini digunakan untuk mencatat Daftar Peserta baru.
* Pengubahan Daftar Peserta: Fitur ini digunakan ketika ada Daftar Peserta Ujian yang salah dalam penginputan atau Peserta tersebut sudah keluar dari sekolah.

### **2.4 Keunggulan Teknologi**

* Responsif dan Cepat: Website ini dirancang dengan model yang sederhana juga ringan, sehingga pengguna dapat menambahkan atau mengubah data dengan cepat.
* Kemudahan Akses: Pengguna dapat dengan mudah melihat bagian yang ingin digunakan karena website ini sangat mudah digunakan walaupun orang awam dalam teknologi sekalipun.

Dengan pembahasan ini, diharapkan pengguna memahami bagaimana platform ini bekerja serta keunggulan yang ditawarkan.

# **BAB III**

## **ANALISA DAN PERANCANGAN**

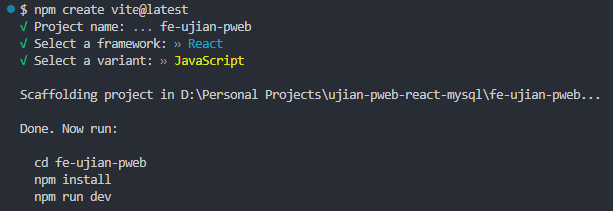
### **3.1 Frontend**

Pada bagian ini, akan dijelaskan tahapan pengembangan frontend menggunakan ReactJS, yang menjadi antarmuka utama bagi pengguna untuk mengakses dan mengelola data dalam aplikasi Participant List.

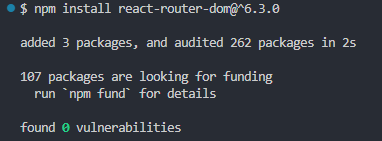
#### **3.1.1 Instalasi Project**

Langkah pertama dalam membangun frontend adalah menyiapkan proyek ReactJS. Bagian ini akan menjelaskan proses instalasi dan konfigurasi awal proyek.

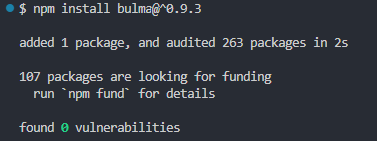
* Instalasi Vite React



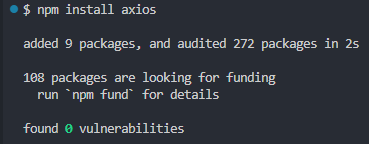
* Instalasi React Router DOM



* Instalasi Bulma



* Instalasi Axios



#### **3.1.1 Components**

ReactJS menggunakan pendekatan berbasis komponen untuk membangun antarmuka yang modular dan mudah dikelola. Bagian ini akan membahas struktur komponen yang digunakan dalam proyek, beserta fungsinya masing-masing.

* Add User

|  |
| --- |
| **import** **React**, { useState } **from** "react";  **import** axios **from** "axios";  **import** { useNavigate } **from** "react-router-dom";  **const** **AddUser** = () => {  **const** [name, setName] = **useState**("");  **const** [email, setEmail] = **useState**("");  **const** [gender, setGender] = **useState**("Male");  **const** navigate = **useNavigate**();  **const** **saveUser** = **async** (e) => {  e.**preventDefault**();  **try** {  **await** axios.**post**("http://localhost:5000/users", {  name,  email,  gender,  });  **navigate**("/");  } **catch** (error) {  console.**log**(error);  }  };  **return** (  <**div** className="columns mt-5 is-centered">  <**div** className="column is-half">  <**form** onSubmit={saveUser}>  <**div** className="field">  <**label** className="label">Name</**label**>  <**div** className="control">  <**input**  type="text"  className="input"  value={name}  onChange={(e) => setName(e.target.value)}  placeholder="Name"  />  </**div**>  </**div**>  <**div** className="field">  <**label** className="label">Email</**label**>  <**div** className="control">  <**input**  type="text"  className="input"  value={email}  onChange={(e) => setEmail(e.target.value)}  placeholder="Email"  />  </**div**>  </**div**>  <**div** className="field">  <**label** className="label">Gender</**label**>  <**div** className="control">  <**div** className="select is-fullwidth">  <**select**  value={gender}  onChange={(e) => setGender(e.target.value)}  >  <**option** value="Male">Male</**option**>  <**option** value="Female">Female</**option**>  </**select**>  </**div**>  </**div**>  </**div**>  <**div** className="field">  <**button** type="submit" className="button is-success">  Save  </**button**>  </**div**>  </**form**>  </**div**>  </**div**>  );  };  **export** **default** **AddUser**; |

* Edit User

|  |
| --- |
| **import** **React**, { useState, useEffect } **from** "react";  **import** axios **from** "axios";  **import** { useNavigate, useParams } **from** "react-router-dom";  **const** **EditUser** = () => {  **const** [name, setName] = **useState**("");  **const** [email, setEmail] = **useState**("");  **const** [gender, setGender] = **useState**("Male");  **const** navigate = **useNavigate**();  **const** { id } = **useParams**();  **useEffect**(() => {  **getUserById**();  }, []);  **const** **updateUser** = **async** (e) => {  e.**preventDefault**();  **try** {  **await** axios.**patch**(`http://localhost:5000/users/${id}`, {  name,  email,  gender,  });  **navigate**("/");  } **catch** (error) {  console.**log**(error);  }  };  **const** **getUserById** = **async** () => {  **const** response = **await** axios.**get**(`http://localhost:5000/users/${id}`);  **setName**(response.data.name);  **setEmail**(response.data.email);  **setGender**(response.data.gender);  };  **return** (  <**div** className="columns mt-5 is-centered">  <**div** className="column is-half">  <**form** onSubmit={updateUser}>  <**div** className="field">  <**label** className="label">Name</**label**>  <**div** className="control">  <**input**  type="text"  className="input"  value={name}  onChange={(e) => setName(e.target.value)}  placeholder="Name"  />  </**div**>  </**div**>  <**div** className="field">  <**label** className="label">Email</**label**>  <**div** className="control">  <**input**  type="text"  className="input"  value={email}  onChange={(e) => setEmail(e.target.value)}  placeholder="Email"  />  </**div**>  </**div**>  <**div** className="field">  <**label** className="label">Gender</**label**>  <**div** className="control">  <**div** className="select is-fullwidth">  <**select**  value={gender}  onChange={(e) => setGender(e.target.value)}  >  <**option** value="Male">Male</**option**>  <**option** value="Female">Female</**option**>  </**select**>  </**div**>  </**div**>  </**div**>  <**div** className="field">  <**button** type="submit" className="button is-success">  Update  </**button**>  </**div**>  </**form**>  </**div**>  </**div**>  );  };  **export** **default** **EditUser**; |

* Users List

|  |
| --- |
| **import** **React**, { useState, useEffect } **from** "react";  **import** axios **from** "axios";  **import** { **Link** } **from** "react-router-dom";  **const** **UsersList** = () => {  **const** [users, setUser] = **useState**([]);  **useEffect**(() => {  **getUsers**();  }, []);  **const** **getUsers** = **async** () => {  **const** response = **await** axios.**get**("http://localhost:5000/users");  **setUser**(response.data);  };  **const** **deleteUser** = **async** (id) => {  **try** {  **await** axios.**delete**(`http://localhost:5000/users/${id}`);  **getUsers**();  } **catch** (error) {  console.**log**(error);  }  };  **return** (  <**div** className="columns mt-5 is-centered">  <**div** className="column is-half">  <**Link** to={`add`} className="button is-success">  Add New  </**Link**>  <**table** className="table is-striped is-fullwidth">  <**thead**>  <**tr**>  <**th**>No</**th**>  <**th**>Name</**th**>  <**th**>Email</**th**>  <**th**>Gender</**th**>  <**th**>Actions</**th**>  </**tr**>  </**thead**>  <**tbody**>  {users.map((user, index) => (  <**tr** key={user.id}>  <**td**>{index + 1}</**td**>  <**td**>{user.name}</**td**>  <**td**>{user.email}</**td**>  <**td**>{user.gender}</**td**>  <**td**>  <**Link**  to={`edit/${user.id}`}  className="button is-small is-info mr-2"  >  Edit  </**Link**>  <**button**  onClick={() => deleteUser(user.id)}  className="button is-small is-danger"  >  Delete  </**button**>  </**td**>  </**tr**>  ))}  </**tbody**>  </**table**>  </**div**>  </**div**>  );  };  **export** **default** **UsersList**; |

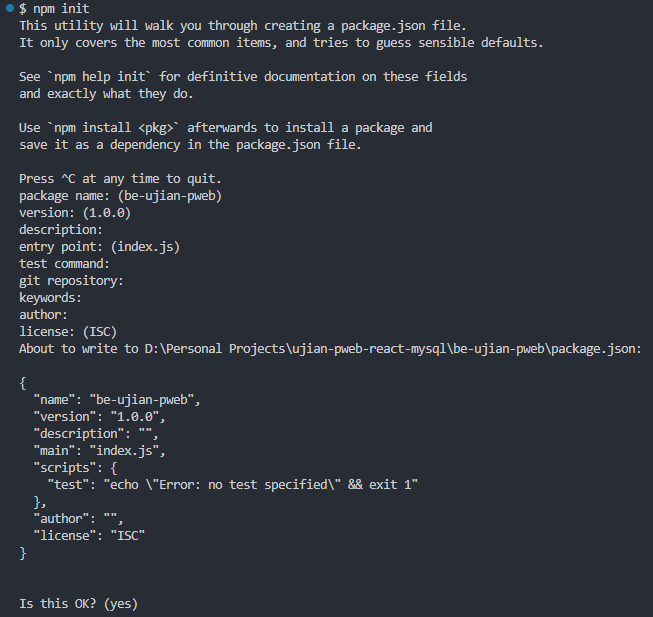
### **3.2 Backend**

Bagian backend bertugas menangani logika server, manajemen data, dan komunikasi dengan database menggunakan Express, MySQL, dan Sequelize. Penjelasan di bagian ini mencakup proses instalasi, konfigurasi, hingga pengelolaan data melalui API.

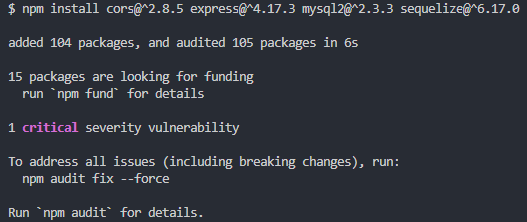
#### **3.2.1 Instalasi Project**

Langkah awal dalam pengembangan backend adalah melakukan instalasi dan konfigurasi proyek Express. Bagian ini akan memandu Anda melalui proses tersebut.

* Inisialisasi package manager NPM



* Intalasi Express, Cors, dan Sequelize



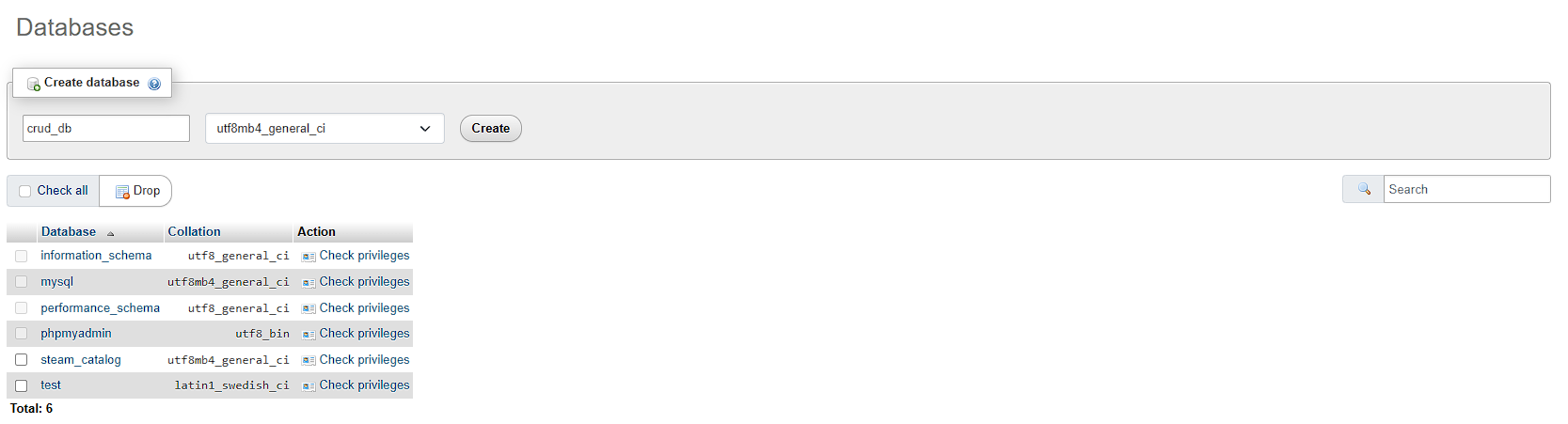
#### **3.2.2 Config**

Konfigurasi backend melibatkan pengaturan koneksi database dan pengaturan lainnya yang diperlukan untuk memastikan aplikasi dapat berfungsi dengan baik. Bagian ini akan menjelaskan detail konfigurasi yang digunakan.

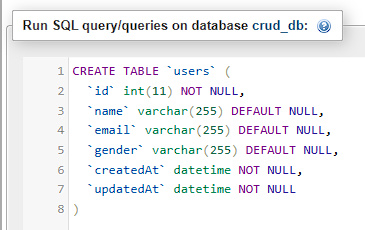
* Config Sequelize

|  |
| --- |
| **import** {**Sequelize**} **from** "sequelize";  **const** db = **new** **Sequelize**('crud\_db','root','',{  host: 'localhost',  dialect: 'mysql'  });  **export** **default** db; |

* Membuat Database



* Membuat Table



#### **3.2.3 Controllers**

Controllers bertugas mengelola logika aplikasi, termasuk menangani permintaan dan mengirimkan respons yang sesuai. Bagian ini akan menjelaskan struktur dan fungsi dari controller yang digunakan.

* User Controller

|  |
| --- |
| **import** **User** **from** "../models/UserModel.js";  **export** **const** **getUsers** = **async**(req, res) =>{  **try** {  **const** response = **await** **User**.**findAll**();  res.**status**(200).**json**(response);  } **catch** (error) {  console.**log**(error.message);  }  }  **export** **const** **getUserById** = **async**(req, res) =>{  **try** {  **const** response = **await** **User**.**findOne**({  where:{  id: req.params.id  }  });  res.**status**(200).**json**(response);  } **catch** (error) {  console.**log**(error.message);  }  }  **export** **const** **createUser** = **async**(req, res) =>{  **try** {  **await** **User**.**create**(req.body);  res.**status**(201).**json**({msg: "User Created"});  } **catch** (error) {  console.**log**(error.message);  }  }  **export** **const** **updateUser** = **async**(req, res) =>{  **try** {  **await** **User**.**update**(req.body,{  where:{  id: req.params.id  }  });  res.**status**(200).**json**({msg: "User Updated"});  } **catch** (error) {  console.**log**(error.message);  }  }  **export** **const** **deleteUser** = **async**(req, res) =>{  **try** {  **await** **User**.**destroy**({  where:{  id: req.params.id  }  });  res.**status**(200).**json**({msg: "User Deleted"});  } **catch** (error) {  console.**log**(error.message);  }  } |

#### **3.2.4 Models**

Models merepresentasikan struktur data yang disimpan di database. Bagian ini akan membahas bagaimana model dibuat menggunakan Sequelize untuk mempermudah interaksi dengan MySQL.

* User Model

|  |
| --- |
| **import** {**Sequelize**} **from** "sequelize";  **import** db **from** "../config/Database.js";  **const** {**DataTypes**} = **Sequelize**;  **const** **User** = db.**define**('users',{  name: **DataTypes**.STRING,  email: **DataTypes**.STRING,  gender: **DataTypes**.STRING  },{  freezeTableName:true  });  **export** **default** **User**;  (**async**()=>{  **await** db.**sync**();  })(); |

#### **3.2.5 Routes**

Routes bertugas menghubungkan endpoint API dengan controller yang sesuai. Bagian ini akan menjelaskan bagaimana rute diatur untuk mendukung fungsi-fungsi utama dalam aplikasi.

* User Route

|  |
| --- |
| **import** express **from** "express";  **import** {  getUsers,  getUserById,  createUser,  updateUser,  deleteUser  } **from** "../controllers/UserController.js";  **const** router = express.**Router**();  router.**get**('/users', getUsers);  router.**get**('/users/:id', getUserById);  router.**post**('/users', createUser);  router.**patch**('/users/:id', updateUser);  router.**delete**('/users/:id', deleteUser);  **export** **default** router; |

#### **3.2.6 Index.js**

|  |
| --- |
| **import** express **from** "express";  **import** cors **from** "cors";  **import** **UserRoute** **from** "./routes/UserRoute.js";  **const** app = **express**();  app.**use**(**cors**());  app.**use**(express.**json**());  app.**use**(**UserRoute**);  app.**listen**(5000, ()=> console.**log**('Server up and running...')); |

Link GitHub Repository: https://github.com/pemudalocal6/crud-ujian-pweb

# **BAB IV**

## **PENUTUP**

### **4.1 Kesimpulan**

Website Daftar Peserta adalah solusi digital yang dirancang untuk mempermudah proses pendaftaran dan pengelolaan data peserta secara efisien dan terorganisir. Dengan fitur seperti formulir pendaftaran online, daftar peserta yang dapat diakses real-time, dan pengelolaan data yang terintegrasi, website ini mampu menghemat waktu, meningkatkan akurasi data, dan mempercepat pengambilan keputusan.

Keunggulan lain dari website daftar peserta meliputi kemudahan akses dari berbagai perangkat, keamanan data yang lebih baik, serta kemampuan untuk menghasilkan laporan yang relevan. Dengan memanfaatkan teknologi ini, organisasi atau penyelenggara acara dapat mengoptimalkan proses administrasi mereka dan memberikan pengalaman yang lebih baik bagi para peserta.

Kesimpulannya, website daftar peserta adalah alat yang penting dan efektif dalam mendukung berbagai kegiatan yang membutuhkan manajemen peserta, seperti seminar, workshop, perlombaan, atau acara besar lainnya.

### **4.2 Saran**

Untuk pengembangan lebih lanjut, berikut beberapa saran yang dapat dipertimbangkan:

1. Peningkatan Tampilan: Mengembangkan desain antarmuka dengan memperhatikan tren UI/UX terbaru untuk memberikan pengalaman yang lebih menarik bagi pengguna.
2. Keamanan Data: Terapkan enkripsi data untuk melindungi informasi pribadi peserta. Gunakan autentikasi dua faktor (2FA) untuk akses admin guna meningkatkan keamanan.
3. Skalabilitas dan Kinerja: Pastikan website dapat menangani jumlah pengguna yang besar tanpa mengalami gangguan, terutama pada saat pendaftaran dibuka. Optimalkan kecepatan loading untuk meningkatkan kenyamanan pengguna.
4. Pencarian dan Filter Peserta: Tambahkan fitur pencarian dan filter berdasarkan kategori seperti nama, tanggal pendaftaran, atau status pembayaran untuk mempermudah navigasi data.