LAMPIRAN

Lampiran 1 Surat Kesediaan Membimbing Tugas Akhir Pembimbing 1

SURAT KESEDIAAN MEMBIMBING TA

Yang bertanda tangan di bawah ini :

Nama	: Ida afriliana ST M.Kom
NIDN	: 0624047703
NIPY	: 12.013.168
Jabatan Struktural	: Ka.Prodi DIII Teknik Komputer
Jabatan Fungsional	: Lektor

Dengan ini menyatakan bersedia untuk menjadi pembimbing I pada Tugas Akhir mahasiswa berikut :

Nama	: Nabilla Rifda Rizqi
NIM	: 21040144
Program Studi	: DIII Teknik Komputer

Judul TA : SISTEM INFORMASI PEMESANAN KEDAI KOPI BERBASIS WEBSITE

Demikian pernyataan ini dibuat agar dapat dilaksanakan sebagaimana mestinya.

Tegal, 10 Juni 2024

Mengetahui
Ka. Prodi DIII Teknik Komputer,

Ida Afriliana.ST.M.Kom
NIPY. 12.013.168

Dosen Pembimbing I,

Ida Afriliana.ST.M.Kom
NIPY. 12.013.168

Lampiran 2 Surat Kesediaan Membimbing Tugas Akhir Pembimbing 2

SURAT KESEDIAAN MEMBIMBING TA

Yang bertanda tangan di bawah ini :

Nama	: Abdul Basit, S.Kom., MT
NIDN	: 0608129106
NIPY	: 01.015.098
Jabatan Struktural	: Sektetaris Prodi
Jabatan Fungsional	: Asisten Ahli

Dengan ini menyatakan bersedia untuk menjadi pembimbing II pada Tugas Akhir mahasiswa berikut :

Nama	: Nabilla Rifda Rizqi
NIM	: 21040144
Program Studi	: DIII Teknik Komputer

Judul TA : SISTEM INFORMASI PEMESANAN KEDAI KOPI BERBASIS WEBSITE

Demikian pernyataan ini dibuat agar dapat dilaksanakan sebagaimana mestinya.

Tegal, 21 Juni 2024

Mengetahui
Ka. Prodi DIII Teknik Komputer,

Ida Afriliana.ST.M.Kom
NIPY. 12.013.168

Dosen Pembimbing II,

Abdul Basit, S.Kom., MT
NIPY. 01.015.098

Lampiran 3 Surat Observasi

POLITEKNIK HARAPAN BERSAMA

D-3 Teknik Komputer

No. : 012.03/KMP.PHB/VII/2024
Lampiran : -
Perihal : Permohonan Izin Observasi Tugas Akhir (TA)

Kepada Yth.
Pimpinan WIJIKOPI
Jl. Werkudoro No.74, Slerok, Kec. Tegal Tim., Kota Tegal, Jawa Tengah 52125

Dengan Hormat,

Sehubungan dengan tugas mata kuliah Tugas Akhir (TA) yang akan diselenggarakan di semester VI (Genap) Program Studi D III Teknik Komputer Politeknik Harapan Bersama Tegal, Maka dengan ini kami mengajukan izin observasi pengambilan data di WIJIKOPI yang Bapak / Ibu Pimpin, untuk kepentingan dalam pembuatan produk Tugas Akhir, dengan Mahasiswa sebagai berikut:

No.	NIM	Nama	No. HP
1	21040144	NABILLA RIFDA RIZQI	085875124472

Demikian surat permohonan ini kami sampaikan atas izin dan kerjasamanya kami sampaikan terima kasih.

Tegal, 08 Juli 2024
Ka. Prodi DIII Teknik Komputer
Politeknik Harapan Bersama Tegal

Ida Afriliana, ST, M.Kom
NIPY. 12.013.168

Jl. Mataram No. 9 Kota Tegal 52143, Jawa Tengah, Indonesia
(0283)357000
komputer@poltektegal.ac.id
poltektegal.ac.id

Lampiran 4 Source Code

```
import { useState, useContext, useEffect, } from "react";
import CategoryList from "../components/products/CategoryList";
import FavoriteProducts from
"../components/products/FavoriteProducts";
import ProductList from "../components/products/ProductList";
import { ProductContext } from "../context/ProductContext";

const Products = () => {
  const { categories } = useContext(ProductContext);
  const [activeComponent, setActiveComponent] = useState('');
  const isAdmin = localStorage.getItem("role") == "admin" ?
true : false;

  useEffect(() => {
    setActiveComponent(categories[0]?.name);
  }, [categories]);

  const handleCategoryClick = (category) => {
    setActiveComponent(category);
  };

  return (
    <div className="flex flex-col sm:flex-row mt-8">
      {/* Favorite Menu or Category List */}
      {localStorage.getItem("role") != "admin" ? (
        <FavoriteProducts />
      ) : (
        <CategoryList />
      )}
      {/* Content Menu*/}
      <div
        className={`${
          localStorage.getItem("role") == "admin"
            ? "sm:basis-3/4"
            : "sm:basis-2/3"
        } flex flex-col mt-5 sm:mt-0 p-3`}
      >
        {/* Categories button */}
        <div className="flex h-[50px] w-full sm:w-[70%]
justify-evenly items-center">
          {categories.map((category, index) => (
            <button
              key={index}
```

```
    onClick={() => handleCategoryClick(category.name)}
              className={`mx-2 text-base sm:text-lg ${
                category.name == activeComponent
                  ? "font-medium text-primary border-b border-
primary"
                  : ""
              }`}
            >
              {category.name}
            </button>
          ))}
        </div>
        <ProductList category={activeComponent}
isAdmin={isAdmin} />
      </div>
    </div>
  );
};

export default Products;

/* eslint-disable react/prop-types */
import { FontAwesomeIcon } from "@fortawesome/react-
fontawesome";
import { Link, useNavigate } from "react-router-dom";
import { faPencilAlt, faTrash } from "@fortawesome/free-solid-
svg-icons";
import Swal from "sweetalert2";
import { UseDeleteProduct } from "./handleDelete";
import { useContext } from "react";
import { ProductContext } from '../../context/ProductContext';

export default function ProductList({ category, isAdmin }) {
  const { products, addProductToCart } =
useContext(ProductContext);
  const navigate = useNavigate();
  const newProducts = products?.filter(product =>
product.category.name == category);
  const handleDelete = UseDeleteProduct();

  const handleOrderClick = (product) => {
    if (!localStorage.getItem("username")) {
      Swal.fire({
        title: "Please sign in to place your order!",
        icon: "info",
```

```
        confirmButtonText: "Ok",
      });
    } else {
      addProductToCart(product);
      Swal.fire({
        title: "Product has been added to the cart.",
        icon: "success",
        confirmButtonText: "Ok",
      });
    }
  };

  return (
    <>
      <div className="flex sm:justify-start justify-center w-
[100%]">
        <table className="table-auto">
          <tbody className="flex flex-col w-[100%]">
            {newProducts?.map((product, index) => (
              <tr
                key={index}
                className="flex w-[100%] justify-start items-
center"
              >
                <td className="basis-1/4">
                  <img
                    src={product.image}
                    alt={product.name}
                    className="mx-2 w-[100px] sm:mx-10 my-5
sm:w-[130px]"
                  />
                </td>
                <td className="basis-1/4">
                  <p className="mx-5 sm:text-start text-center
text-xs sm:text-base sm:mx-10 my-5">
                    {product.name}
                  </p>
                </td>
                <td className="basis-1/4">
                  <p className="mx-5 sm:text-start text-center
text-xs sm:text-base sm:mx-10 my-5">
                    {product.price}
                  </p>
                </td>
                {isAdmin ? (
```

```
                  <td className="basis-1/4 flex justify-around
p-4">
                    <div className="flex">
                      <button
                        onClick={() =>
                          handleDelete(product)
                        }
                        className="bg-[#F41A1A] text-black w-7
h-7 rounded mr-2"
                      >
                        <FontAwesomeIcon icon={faTrash} />
                      </button>
                      <Link
to={`/updateproduct/${product.id}`}>
                        <button className="bg-[#3fff00] text-
black w-7 h-7 rounded">
                          <FontAwesomeIcon icon={faPencilAlt}
/>
                        </button>
                      </Link>
                    </div>
                  </td>
                ) : (
                  <td className="basis-1/4">
                    {/* Tombol untuk menambahkan produk ke
keranjang */}
                    <button
                      onClick={() => handleOrderClick(product)}
                      className="rectangle w-[40px] h-[40px]
bg-secondary border border-[#747474] mx-4 sm:mx-12 my-5"
                    >
                      <p className="text-2xl text-white">+</p>
                    </button>
                  </td>
                )}
              </tr>
            ))}
          </tbody>
        </table>
      </div>
      {!isAdmin && (
        <div className="flex justify-end pe-10 w-[100%] my-5">
          <button
            onClick={() => navigate('/checkout')}
```

```
            className="text-base sm:text-2xl font-bold bg-
[#F4991A] border border-[#321313] w-[120px] h-[40px] flex
justify-center items-center"
          >
            Next &gt;
          </button>
        </div>
      )}
    </>
  );
}
```

```
const { Product, Category } = require('../models');
const response = require('../services/response');
const upload = require('../middlewares/upload');

const getProducts = async (req, res) => {
    try {
        const products = await Product.findAll({
            include: {
                model: Category,
                as: 'category',
            },
            attributes: {
                exclude: ['category_id'],
            },
        });
        const data = products.map((product) => {
            return {
                ...product.toJSON(),
                image:
`uploads/images/products/${product.image}`,
            };
        });
        response(200, true, data, 'Success get all products',
res);
    } catch (error) {
        console.error(error);
        response(400, false, error, 'Failed to get products',
res);
    }
};
```

```
const getProductsByCategory = async (req, res) => {
    const { categoryId } = req.params;
    try {
        const checkCategory = await
Category.findByPk(categoryId);
        if (!checkCategory) {
            return response(400, false, '', 'category id not
found', res);
        }
        const products = await Product.findAll({
            where: {
                category_id: categoryId,
            },
        });
        const data = products.map((product) => {
            return {
                ...product.toJSON(),
                image:
`uploads/images/products/${product.image}`,
            };
        });
        return response(
            200,
            true,
            data,
            'Success get product by id category',
            res
        );
    } catch (error) {
        console.error(error);
        response(400, false, error, 'Failed to get products',
res);
    }
};

const getProductById = async (req, res) => {
    try {
        const { id } = req.params;
        const product = await Product.findByPk(id, {
            include: {
                model: Category,
                as: 'category',
            },
        });
        const data = {
```

```
            ...product.toJSON(),
            image: `uploads/images/products/${product.image}`,
        };
        response(200, true, data, 'Success get product by id',
res);
    } catch (error) {
        console.error(error);
        response(400, false, error, 'Failed to get the
product', res);
    }
};

const createProduct = async (req, res) => {
    upload(req, res, async (err) => {
        if (err) {
            console.error(err);
            return response(
                400,
                false,
                err,
                'Failed to create new product, image error',
                res
            );
        }
        try {
            const product = {
                ...req.body,
            };
            const image = req?.file?.filename;
            const data = {
                ...product,
                image,
            };
            const newProduct = await Product.create(data);
            response(201, true, newProduct, 'Succes to add new
product', res);
        } catch (error) {
            console.error(error);
            response(400, false, error, 'Failed to add
product', res);
        }
    });
};

const updateProduct = async (req, res) => {
```

```
    upload(req, res, async (err) => {
        if (err) {
            console.error(err);
            return response(
                400,
                false,
                err,
                'Failed to create new product, image error',
                res
            );
        }
        try {
            const { id } = req.params;
            const edit = {
                ...req.body,
            };
            const image = req?.file?.filename;
            const data = {
                ...edit,
                image,
            };
            const product = await Product.findByPk(id);
            if (!product) {
                return response(
                    400,
                    false,
                    '',
                    'Failed to edit product, id product not
found',
                    res
                );
            }
            const editedProduct = await product.update(data);
            response(201, true, editedProduct, 'Success edited
product', res);
        } catch (error) {
            console.error(error);
            response(400, false, error, 'Failed to edit
product', res);
        }
    });
};

const deleteProduct = async (req, res) => {
    try {
```

```
        const { id } = req.params;
        const product = await Product.findByPk(id);
        if (!product) {
            return response(
                400,
                false,
                '',
                'Failed to delete product, id product not
found',
                res
            );
        }
        const deletedProduct = await product.destroy();
        response(201, true, deletedProduct, 'Success delete
product', res);
    } catch (error) {
        console.error(error);
        response(400, false, error, 'Failed to delete product',
res);
    }
};

const restoreProduct = async (req, res) => {
    try {
        const { id } = req.params;
        const product = await Product.findByPk(id, { paranoid:
false });
        if (!product) {
            return response(
                400,
                false,
                '',
                'Failed to restore product, id product not
found',
                res
            );
        }
        const deletedProduct = await product.restore();
        response(
            201,
            true,
            deletedProduct,
            'Success restore deleted product',
            res
        );
```

```
    } catch (error) {
        console.error(error);
        response(400, false, error, 'Failed to delete product',
res);
    }
};

module.exports = {
    getProducts,
    getProductsByCategory,
    getProductById,
    createProduct,
    updateProduct,
    deleteProduct,
    restoreProduct,
};
```

Lampiran 5 Foto Dokumentasi

Lampiran 6 Wawancara Observasi

1. Apakah di Wijikopi ini memiliki *website*?

   Narasumber: Saat ini, Wijikopi belum memiliki *website*.

2. Untuk sistem *order* di Wijikopi bagaimana?

   Narasumber : Saat ini, sistem *ordernya* masih dilakukan melalui kasir.

3. Bagaimana pengumpulan data di Wijikopi dilakukan?

   Narasumber: Pengumpulan data dilakukan melalui aplikasi kasir yang mencakup informasi stok barang, bahan-bahan, dan manajemen.

4. Apa saja sistem pembayaran yang dapat digunakan di Wijikopi?

   Narasumber Sistem pembayaran yang dapat digunakan di Wijikopi meliputi QRIS, tunai, dan edc.

5. Apa kelebihan dan kekurangan dari Wijikopi?

   Narasumber: Kekurangannya termasuk tempat yang kurang update dari *coffee shop* yang lainya dan keterbatasan tempat parkir. Namun, kelebihannya adalah produk kopi berkualitas dengan pelayanan yang nyaman.

6. Berapa sering perubahan menu dilakukan di Wijikopi?

   Narasumber: Biasanya perubahan menu dilakukan setiap tiga bulan sekali setelah pengajuan menu baru dipantau terlebih dahulu.jika bagus pembelianya maka akan di masukan ke menu regular, jika tidak menu akan di *down*.