LAMPIRAN

Code Program:

```
const { user } = require("../models");
const bcrypt = require("bcrypt");
const salt = 10;

function encryptPassword(password) {
  return new Promise((resolve, reject) => {
    bcrypt.hash(password, salt, (err, encryptedPassword) => {
      if (!!err) {
        reject(err);
        return;
      }
      resolve(encryptedPassword);
    });
  });
}

module.exports = {
  async getAllUserData(req, res) {
    const findAll = () => {
      return user.findAll();
    };
    try {
      const dataUsers = await findAll();
      if (!dataUsers) {
        res.status(404).json({
          status: "Failed",
          message: "Data not found",
        });
      }
      res.status(200).json({
        status: "Success",
        message: "Get All Data User Success",
        data: dataUsers,
      });
    } catch (error) {
      res.status(500).json({
        status: "Error",
        message: error.message,
      });
    }
  },

  async getUserById(req, res) {
```

```
    try {
      const idUser = req.params.id;
      const findUserId = () => {
        return user.findOne({
          where: { id: idUser },
        });
      };

      const dataUsersId = await findUserId();

      if (!dataUsersId) {
        res.status(404).json({
          status: "Failed",
          message: "User not found",
        });
      }
      res.status(200).json({
        status: "Success",
        message: "Get Data User Successfully",
        data: dataUsersId,
      });
    } catch (error) {
      res.status(500).json({
        status: "Error",
        message: error.message,
      });
    }
  },

  async updateUserData(req, res) {
    const idUser = req.params.id;

    const findUserId = async () => {
      return await user.findOne({
        where: { id: idUser },
      });
    };

    const dataUsersId = await findUserId();

    user
      .update(
        {
          name: req.body.name,
          phone: req.body.phone,
```

```
          email: req.body.email,
        },
        {
          where: { id: req.params.id },
        }
      )
      .then(() => {
        res.status(200).json({
          status: "Success",
          message: "Update Data User Successfully",
        });
      })
      .catch((err) => {
        res.status(422).json(err);
      });
  },

  async deleteUser(req, res) {
    try {
      const idUser = req.params.id;
      user
        .destroy({
          where: { id: idUser },
        })
        .then(() => {
          res.status(200).json({
            status: "Success",
            message: "User Data deleted successfully",
          });
        })
        .catch((err) => {
          res.status(422).json(err);
        });
    } catch (error) {
      res.status(500).json({
        status: "Error",
        message: error.message,
      });
    }
  },
};
```

```
const { Product } = require("../models");
const { v4: uuid } = require("uuid");
const { Op } = require("sequelize");

module.exports = {
  async createproduct(req, res) {
    try {
      const {
        kodebarang,
        namabarang,
        image1,
        image2,
        image3,
        typebarang,
        deskripsibarang,
        stockbarang,
        satuanbarang,
        price,
      } = req.body;

      const productForm = await Product.create({
        id: uuid(),
        kodebarang: kodebarang,
        namabarang: namabarang,
        image: image1,
        image2: image2,
        image3: image3,
        typebarang: typebarang,
        deskripsibarang: deskripsibarang,
        stockbarang: stockbarang,
        satuanbarang: satuanbarang,
        price: price,
      });

      res.status(201).json({
        status: "Success",
        message: "Created Product Success",
        data: productForm,
      });
    } catch (error) {
      res.status(400).json({
        status: "Failed",
        message: error.message,
      });
    }
```

```
  },

  async getAllProductData(req, res) {
    const findAll = () => {
      return Product.findAll();
    };
    try {
      const dataProduct = await findAll();
      if (!dataProduct) {
        res.status(404).json({
          status: "Failed",
          message: "Data not found",
        });
      }
      res.status(200).json({
        status: "Success",
        message: "Get All Data User Success",
        data: dataProduct,
      });
    } catch (error) {
      res.status(500).json({
        status: "Error",
        message: error.message,
      });
    }
  },

  async getProductByName(req, res) {
    const namabarang = req.query.namabarang ? req.query.namabarang :
"";

    //Mencari nama barang
    const querySearch = {
      namabarang: {
        [Op.iLike]: `%${namabarang}`,
      },
    };
    const findAll = () => {
        return product.findAll({
          where: querySearch,
        });
      };
      try {
        const dataProduct = await findAll();
        if (!dataProduct) {
```

```
        res.status(404).json({
          status: "Failed",
          message: "Data not found",
          data: {},
        });
      }
      res.status(200).json({
        status: "Success",
        message: "Get Data Product Success",
        data: dataDestFav,
      });
    } catch (error) {
      res.status(500).json({
        status: "Error",
        message: error.message,
      });
    }
  },

async getProductById(req, res) {
  try {
    const idProduct = req.params.id;
    const findProductId = () => {
      return Product.findOne({
        where: { id: idProduct },
      });
    };

    const dataProductId = await findProductId();

    if (!dataProductId) {
      res.status(404).json({
        status: "Failed",
        message: "Data not found",
      });
    }
    res.status(200).json({
      status: "Success",
      message: "Get Data Product Successfully",
      data: dataProductId,
    });
  } catch (error) {
    res.status(500).json({
      status: "Error",
      message: error.message,
```

```
    });
  }
},

async deleteProduct(req, res) {
  try {
    const idProduct = req.params.id;
    Product
      .destroy({
        where: { id: idProduct },
      })
      .then(() => {
        res.status(200).json({
          status: "Success",
          message: "Product Data deleted successfully",
        });
      })
      .catch((err) => {
        res.status(422).json(err);
      });
  } catch (error) {
    res.status(500).json({
      status: "Error",
      message: error.message,
    });
  }
},

async updateProductData(req, res) {
  const idProduct = req.params.id;

  const findProductId = async () => {
    return await product.findOne({
      where: { id: idProduct },
    });
  };

  const dataproductsId = await findProductId();

  product
    .update(
      {
        image: req.body.image,
        image2: req.body.image2,
        image3: req.body.image3,
```

```
          stockbarang : req.body.stockbarang,
          deskripsibarang : req.body.deskripsibarang,
          price: req.body.price,
        },
        {
          where: { id: req.params.id },
        }
      )
      .then(() => {
        res.status(200).json({
          status: "Success",
          message: "Update Data Product Successfully",
        });
      })
      .catch((err) => {
        res.status(422).json(err);
      });
  },
};
```

```
const { Checkout } = require("../models");
const { Alamat } = require("../models");
const { Product } = require("../models");
const { user } = require("../models");
const { Notif } = require("../models");
const { v4: uuid } = require("uuid");
const { Op } = require("sequelize");

module.exports = {
  async createCheckout(req, res) {
    try {
      const {
        productId,
        total_barang,
        alamat,
        hargaOngkir,
      } = req.body;

      // Get the current authenticated user ID
      const usersId = req.user.id; // Ganti `req.user.id` dengan
cara yang sesuai untuk mengakses ID pengguna saat ini

      // check if the provided productID exists in the Product Table
      const product = await Product.findOne({
        where: {
          id: productId,
        },
      });

      if (!product) {
        res.status(400).json({
          status: "Failed",
          message: "Invalid productID, Product does not exist",
        });
        return;
      }

      // Calculate the total price
      const totalOngkir = 70000
      const TotalPrice = (total_barang * product.price) +
totalOngkir;

      // Create product checkout
      const checkout = await Checkout.create({
        id: uuid(),
```

```
        usersId: usersId,
        productId: productId,
        total_barang,
        hargaOngkir: totalOngkir,
        total_price: TotalPrice,
      });

      // Create Alamat
      for (const alamatData of alamat) {
        await Alamat.create({
          id: uuid(),
          checkoutsId: checkout.id,
          productId: productId,
          name: alamatData.name,
          email: alamatData.email,
          phone: alamatData.phone,
          provinsi: alamatData.provinsi,
          kodepos: alamatData.Kodepos,
          alamatLengkap: alamatData.alamatLengkap,
        });
      }

      res.status(201).json({
        status: "Success",
        message: "Checkout created successfully",
      });
    } catch (error) {
      console.error(error);
      res.status(500).json({
        message: "Internal server error",
      });
    }
  },

  async getAllCheckoutData(req, res) {
    try {
      const idUser = req.user.id; // Mengambil ID pengguna dari
token
      const checkoutData = await Checkout.findAll({
        where: {
          usersId: idUser, // Menggunakan ID pengguna dalam kondisi
WHERE
        },
        include: [
          {
```

```
          model: Alamat,
        },
        {
          model: Product,
          as: "Product",
          where: {
            id: { [Op.col]: "Checkout.productId" },
          },
        },
      ],
    });

    if (checkoutData.length === 0) {
      // jika transaction tidak ada
      res.status(404).json({
        message: "No Checkout data found",
        data: [],
      });
      return;
    }

    const formattedCheckoutData = checkoutData.map((checkout) => {
      const productPrice = checkout.Product
        ? checkout.Product.price
        : 0;
      const totalBarang = checkout.total_barang;
      const totalOngkir = checkout.hargaOngkir;
      const totalPrice =
        (productPrice * totalBarang) + totalOngkir;

      return {
        id: checkout.id,
        usersId: checkout.usersId,
        productId: checkout.productId,
        total_barang: checkout.total_barang,
        createdAt: checkout.createdAt,
        updatedAt: checkout.updatedAt,
        Product: checkout.Product,
        hargaOngkir: checkout.hargaOngkir,
        total_price: totalPrice,
        alamat: checkout.Alamat,
      };
    });

    res.status(200).json({
```

```
        status: "Success",
        message: "Checkout data retrieved successfully",
        data: formattedCheckoutData,
      });
    } catch (error) {
      console.log(error);
      res.status(500).json({
        message: error,
      });
    }
  },
  async getDataCheckoutById(req, res) {
    try {
      const idCheckout = req.params.id; // Mengambil ID pengguna
dari token
      const checkoutData = await Checkout.findAll({
        where: {
          id: idCheckout, // Menggunakan ID pengguna dalam kondisi
WHERE
        },
        include: [
          {
            model: Alamat, // model alamat associate to checkout
          },
          {
            model: Product,
            as: "Product",
            where: {
              id: { [Op.col]: "Checkout.productId" },
            },
          },
        ],
      });

      if (checkoutData.length === 0) {
        // jika transaction tidak ada
        res.status(404).json({
          message: "No checkout data found",
          data: [],
        });
        return;
      }

      const formattedCheckoutData = checkoutData.map((checkout) => {
        const productPrice = checkout.Product
```

```
          ? checkout.Product.price
          : 0;
        const totalBarang = checkout.total_barang;
        const totalOngkir = checkout.hargaOngkir;
        const totalPrice =
          (productPrice * totalBarang) + totalOngkir;

          return {
            id: checkout.id,
            usersId: checkout.usersId,
            productId: checkout.productId,
            total_barang: checkout.total_barang,
            createdAt: checkout.createdAt,
            updatedAt: checkout.updatedAt,
            Product: checkout.Product,
            total_price: totalPrice,
            alamat: checkout.Alamat,
          };
        });

      res.status(200).json({
        message: "Checkout data by id retrieved successfully",
        data: formattedCheckoutData,
      });
    } catch (error) {
      console.log(error);
      res.status(500).json({
        message: error,
      });
    }
  },
  async updateCheckoutData(req, res) {
    const idCheckout = req.params.id;

    const findDataCheckoutId = async () => {
      return await Checkout.findOne({
        where: {
          id: idCheckout,
        },
      });
    };

    Checkout.update({
      id: uuid(),
      name: req.body.name,
```

```
      email: req.body.email,
      phone: req.body.phone,
      provinsi: req.body.provinsi,
      kodepos: req.body.Kodepos,
      alamatLengkap: req.body.alamatLengkap,
    })
      .then(() => {
        res.status(200).json({
          status: "Success",
          message: "Update Data Checkout Successfully",
        });
      })
      .catch((err) => {
        res.status(422).json(err);
      });
  },
  async deleteCheckout(req, res) {
    try {
      const idCheckout = req.params.id;
      Checkout.destroy({
        where: {
          id: idCheckout,
        },
      })
        .then(() => {
          res.status(200).json({
            status: "Success",
            message: "Checkout Data Deleted successfully",
          });
        })
        .catch((err) => {
          res.status(422).json(err);
        });
    } catch (error) {
      res.status(500).json({
        status: "Erro",
        message: error.message,
      });
    }
  },
  async deleteAllDataCheckout(req, res) {
    Checkout.destroy({ truncate: true })
      .then(() => {
        res.status(200).json({
          status: "Success",
```

```
          message: "Checkout Data deleted successfully",
        });
      })
      .catch((error) => {
        res.status(422).json(error);
      });
  },
};
```

```
const { Alamat } = require("../models");

module.exports = {
  async deleteAllDataPass(req, res) {
    Alamat.destroy({ truncate: true })
      .then(() => {
        res.status(200).json({
          status: "Success",
          message: "Alamat Data deleted successfully",
        });
      })
      .catch((error) => {
        res.status(422).json(error);
      });
  },
};
```

```
const { Product } = require("../models");
const { Transaction } = require("../models");
const { Checkout } = require("../models");
const { Alamat } = require("../models");
const { user } = require("../models");
const { Op } = require("sequelize");
const { v4: uuid } = require("uuid");

module.exports = {
  async getAllTransactionData(req, res) {
    try {
      const idUser = req.user.id; // Mengambil ID pengguna dari
token
      const checkoutData = await Checkout.findAll({
        where: {
          usersId: idUser, // Menggunakan ID pengguna dalam kondisi
WHERE
        },
        include: [
          {
            model: Alamat,
          },
          {
            model: Product,
            as: "Product",
            where: {
              id: { [Op.col]: "Checkout.productId" },
            },
            required: false,
          },
        ],
        order: [["createdAt", "DESC"]], // Menambahkan pengurutan
berdasarkan createdAt secara menurun (data terbaru)
      });

      if (checkoutData.length === 0) {
        // jika transaksi tidak ada
        res.status(404).json({
          message: "No transaction data found",
          data: [],
        });
        return;
      }

      const formattedCheckoutData = checkoutData.map((checkout) => {
```

```
        const productPrice = checkout.Product
          ? checkout.Product.price
          : 0;
        const totalBarang = checkout.total_barang;
        const totalOngkir = checkout.hargaOngkir;
        const totalPrice =
          (productPrice * totalBarang) + totalOngkir;

        return {
          id: checkout.id,
          usersId: checkout.usersId,
          productId: checkout.productId,
          total_barang: checkout.total_barang,
          createdAt: checkout.createdAt,
          updatedAt: checkout.updatedAt,
          Product: checkout.Product,
          total_price: totalPrice,
          Alamat: checkout.Alamat,
        };
      });

      res.status(200).json({
        status: "Success",
        message: "Transaction data successfully obtained",
        data: formattedCheckoutData,
      });
    } catch (error) {
      console.log(error);
      res.status(500).json({
        message: error,
      });
    }
  },
  async getDataTransactionById(req, res) {
    try {
      const usersId = req.user.id; // Menggunakan ID pengguna saat
ini

      const transactions = await Transaction.findAll({
        where: {
          usersId,
        },
        include: [
          {
            model: Product,
```

```
            as: "products",
          },
          {
            model: Checkout,
            as: "checkouts",
          },
        ],
      });

      res.status(200).json({
        data: transactions,
      });
    } catch (error) {
      console.error(error);
      res.status(500).json({
        message: "Internal server error",
      });
    }
  },

  async getAllTransactionDataAdmin(req, res) {
    try {
      const checkoutData = await Checkout.findAll({
        include: [
          {
            model: Alamat,
          },
          {
            model: Product,
            as: "product",
            where: {
              id: { [Op.col]: "Checkout.productId" },
            },
            required: false,
          },
        ],
        order: [["createdAt", "DESC"]],
      });

      if (checkoutData.length === 0) {
        res.status(404).json({
          message: "No transaction data found",
          data: [],
        });
        return;
```

```
    }

    const formattedCheckoutData = checkoutData.map((checkout) => {
      const productPrice = checkout.Product
        ? checkout.Product.price
        : 0;
      const totalBarang = checkout.total_barang;
      const totalOngkir = checkout.hargaOngkir;
      const totalPrice =
        (productPrice * totalBarang) + totalOngkir;

      return {
        id: checkout.id,
        usersId: checkout.usersId,
        productId: checkout.productId,
        total_barang: checkout.total_barang,
        createdAt: checkout.createdAt,
        updatedAt: checkout.updatedAt,
        Product: checkout.Product,
        total_price: totalPrice,
        Alamat: checkout.Alamat,
      };
    });

    res.status(200).json({
      status: "Success",
      message: "Transaction data successfully obtained",
      data: formattedCheckoutData,
    });
  } catch (error) {
    console.log(error);
    res.status(500).json({
      message: error,
    });
  }
},

async updateDataTrans(req, res) {
  try {
    const idDataTrans = req.params.id;

    // Mengambil data user dari model User berdasarkan ID user
    const iduser = await user.findByPk(req.body.userId);
```

```
      // Mengambil data prpduct dari model Ticket berdasarkan ID
prpduct
      const product = await Product.findByPk(req.body.productId);
      // Menghitung total amount berdasarkan price prpduct dan
quantity
      const amount = product.price * req.body.quantity;

      // Membuat transaksi baru dengan data yang diambil
      const transaction = await Transaction.update(
        {
          id: uuid(),
          usersId: iduser.id,
          productId: ticket.id,
          amounts: amount,
          date: req.body.date,
          status: "Success",
        },
        {
          where: { id: idDataTrans },
        }
      );

      res.status(200).json({
        status: "Success",
        message: "Update Data Transaction Successfully",
        data: transaction,
      });
    } catch (error) {
      res.status(500).json({
        status: "Failed",
        message: error.message,
      });
    }
  },

  async deleteDataTrans(req, res) {
    try {
      const idDataTrans = req.params.id;
      Transaction.destroy({
        where: {
          id: idDataTrans,
        },
      })
        .then(() => {
          res.status(200).json({
```

```
            status: "Success",
            message: "Transaction Data deleted successfully",
          });
        })
        .catch((err) => {
          res.status(422).json(err);
        });
    } catch (error) {
      res.status(500).json({
        status: "Error",
        message: error.message,
      });
    }
  },

  async deleteAllDataTrans(req, res) {
    Transaction.destroy({ truncate: true })
      .then(() => {
        res.status(200).json({
          status: "Success",
          message: "Transaction Data deleted successfully",
        });
      })
      .catch((error) => {
        res.status(422).json(error);
      });
  },
};
```

```
const { Payment } = require("../models");
const { Notif } = require("../models");
const { v4: uuid } = require("uuid");

module.exports = {
  async createPayment(req, res) {
    try {
      const { cardNumber, cardHolderName, cvc, expiration, country }
= req.body;
      const usersId = req.user.id;
      const paymentForm = await Payment.create({
        id: uuid(),
        usersId,
        cardNumber: cardNumber,
        cardHolderName: cardHolderName,
        cvc: cvc,
        expiration: expiration,
        country: country,
        status: true,
      });

      // create notification
      const message = `Ticket payment successful! Enjoy your trip`;

      const notif = await Notif.create({
        id: uuid(),
        message: message,
        usersId: usersId,
        read: false,
      });

      res.status(201).json({
        status: "Success",
        message: "Created payment Success",
        data: paymentForm,
      });
    } catch (error) {
      res.status(400).json({
        status: "Failed",
        message: error.message,
      });
    }
  },

  async getAllPaymentData(req, res) {
```

```
    const findAll = () => {
      return Payment.findAll();
    };
    try {
      const dataPayment = await findAll();
      if (!dataPayment) {
        res.status(404).json({
          status: "Failed",
          message: "Data not found",
        });
      }
      res.status(200).json({
        status: "Success",
        message: "Get All Data Payment Success",
        data: dataPayment,
      });
    } catch (error) {
      res.status(500).json({
        status: "Error",
        message: error.message,
      });
    }
  },

  async getPaymentById(req, res) {
    try {
      const idPayment = req.params.id;
      const findPaymentId = () => {
        return Payment.findOne({
          where: { id: idPayment },
        });
      };

      const dataPaymentId = await findPaymentId();
      if (!dataPaymentId) {
        res.status(404).json({
          status: "Failed",
          message: "Data not found",
          data: {},
        });
      }
      res.status(200).json({
        status: "Success",
        message: "Get Data Payment Successfully",
        data: dataPaymentId,
```

```
      });
    } catch (error) {
      res.status(500).json({
        status: "Error",
        message: error.message,
        data: {},
      });
    }
  },

  async deleteAllDataPayment(req, res) {
    Payment.destroy({ truncate: true })
      .then(() => {
        res.status(200).json({
          status: "Success",
          message: "Payment Data deleted successfully",
        });
      })
      .catch((error) => {
        res.status(422).json(error);
      });
  },
};
```

Berikut adalah tempat produksi dari SA Jaya Meubel: