- .env

DATABASE\_URL="mysql://root:@localhost:3306/baturraden\_tourism"

# Midtrans Configuration

MIDTRANS\_MERCHANT\_ID=G691677310

# Untuk API route (backend)

MIDTRANS\_SERVER\_KEY=SB-Mid-server-I7rTkeqTul4rd2Y8XAPvqfRN

# Untuk frontend Snap.js (bisa dipakai di layout.tsx)

MIDTRANS\_CLIENT\_KEY=SB-Mid-client-uWw4CxhB3ZL\_v4\_P

# Authtoken si ngrok

# ngrok config add-authtoken 2ppPJhbo2C4bG7to0WISQyAYhzi\_makmHr4ScGJff9CfCgfF

# About Mailtrap

# Credentials

# Host --> sandbox.smtp.mailtrap.io

# Port --> 25, 465, 587 or 2525

# Username ---> 1fab3abbe59fbc

# Password --> a6ddc5cd4efbbb

# Auth --> PLAIN, LOGIN and CRAM-MD5

# TLS --> Optional (STARTTLS on all ports)

- .env.local

# FIREBASE CONFIG

NEXT\_PUBLIC\_FIREBASE\_API\_KEY=AIzaSyA1fZ4GKNyRm4jNPP3OEzIZ1bsArWmsdDc

NEXT\_PUBLIC\_FIREBASE\_AUTH\_DOMAIN=mybaturraden.firebaseapp.com

NEXT\_PUBLIC\_FIREBASE\_PROJECT\_ID=mybaturraden

NEXT\_PUBLIC\_FIREBASE\_STORAGE\_BUCKET=mybaturraden.appspot.com

NEXT\_PUBLIC\_FIREBASE\_MESSAGING\_SENDER\_ID=802881383396

NEXT\_PUBLIC\_FIREBASE\_APP\_ID=1:802881383396:web:245fef8417dba0d91e35f1

# MIDTRANS (Frontend)

NEXT\_PUBLIC\_MIDTRANS\_CLIENT\_KEY=SB-Mid-client-uWw4CxhB3ZL\_v4\_P

- schema.prisma

generator client {

  provider = "prisma-client-js"

}

datasource db {

  provider = "mysql"

  url      = env("DATABASE\_URL")

}

model category\_tourism {

  category\_id   String         @id @db.Char(5)

  category\_name String         @db.VarChar(20)

  category\_desc String?        @db.VarChar(150)

  category\_icon String?        @db.VarChar(255)

  spots         spot\_tourism[]

}

model facility {

  facility\_id       String             @id @db.Char(10)

  facility\_name     String             @db.VarChar(100)

  facility\_icon     String?            @db.VarChar(255)

  facility\_tourisms facility\_tourism[]

}

model facility\_tourism {

  facility\_tourism\_id String        @id @db.Char(10)

  spot\_id             String?       @db.Char(10)

  facility\_id         String?       @db.Char(10)

  spot                spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "facility\_tourism\_ibfk\_1")

  facility            facility?     @relation(fields: [facility\_id], references: [facility\_id], onDelete: Cascade, onUpdate: NoAction, map: "facility\_tourism\_ibfk\_2")

  @@index([facility\_id], map: "facility\_id")

  @@index([spot\_id], map: "spot\_id")

}

model gallery {

  gallery\_id      String        @id @db.Char(10)

  spot\_id         String?       @db.Char(10)

  gallery\_img     String?       @db.VarChar(255)

  gallery\_caption String?       @db.VarChar(255)

  spot            spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "gallery\_ibfk\_1")

  @@index([spot\_id], map: "spot\_id")

}

model operating\_hours {

  operating\_id  String        @id @db.Char(10)

  spot\_id       String?       @db.Char(10)

  operating\_day OperatingDay

  hours\_open    DateTime      @db.Time(0)

  hours\_closed  DateTime      @db.Time(0)

  spot          spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "operating\_hours\_ibfk\_1")

  @@index([spot\_id], map: "spot\_id")

}

model payment {

  payment\_id       Int             @id @default(autoincrement())

  transaction\_id   Int?

  method\_id        Int?

  reference\_number String          @unique(map: "reference\_number") @db.VarChar(50)

  payment\_amount   Decimal         @db.Decimal(10, 2)

  service\_fee      Decimal         @db.Decimal(10, 2)

  total\_payment    Decimal?        @db.Decimal(10, 2)

  payment\_status   PaymentStatus?  @default(pending)

  payment\_time     DateTime?       @default(now()) @db.Timestamp(0)

  expiration\_time  DateTime?       @db.Timestamp(0)

  response\_data    Json?

  transaction      transaction?    @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade, onUpdate: NoAction, map: "payment\_ibfk\_1")

  method           payment\_method? @relation(fields: [method\_id], references: [method\_id], onUpdate: NoAction, map: "payment\_ibfk\_2")

  @@index([method\_id], map: "method\_id")

  @@index([transaction\_id], map: "transaction\_id")

}

model payment\_method {

  method\_id       Int               @id @default(autoincrement())

  provider\_id     String?           @db.Char(10)

  method\_name     String            @db.VarChar(50)

  method\_code     String            @unique(map: "method\_code") @db.VarChar(50)

  method\_icon     String?           @db.VarChar(255)

  service\_fee     Decimal           @default(0.00) @db.Decimal(10, 2)

  minimum\_payment Decimal           @default(0.00) @db.Decimal(10, 2)

  maximum\_payment Decimal           @db.Decimal(10, 2)

  is\_active       Boolean?          @default(true)

  payments        payment[]

  provider        payment\_provider? @relation(fields: [provider\_id], references: [provider\_id], onUpdate: NoAction, map: "payment\_method\_ibfk\_1")

  @@index([provider\_id], map: "provider\_id")

}

model payment\_provider {

  provider\_id   String           @id @db.Char(10)

  provider\_name String           @db.VarChar(50)

  provider\_desc String?          @db.VarChar(255)

  provider\_logo String?          @db.VarChar(255)

  is\_active     Boolean?         @default(true)

  methods       payment\_method[]

}

model reviews {

  reviews\_id         String        @id @db.Char(10)

  spot\_id            String?       @db.Char(10)

  user\_id            String?       @db.Char(10)

  reviews\_rating     Int?          @db.TinyInt

  reviews\_desc       String?       @db.Text

  reviews\_created\_at DateTime?     @default(now()) @db.Timestamp(0)

  spot               spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "reviews\_ibfk\_1")

  user               users?        @relation(fields: [user\_id], references: [user\_id], onDelete: Cascade, onUpdate: NoAction, map: "reviews\_ibfk\_2")

  @@index([spot\_id], map: "spot\_id")

  @@index([user\_id], map: "user\_id")

}

model spot\_tourism {

  spot\_id           String             @id @db.Char(10)

  category\_id       String?            @db.Char(5)

  village\_id        String?            @db.Char(10)

  spot\_name         String             @db.VarChar(60)

  spot\_desc         String?            @db.Text

  spot\_address      String?            @db.VarChar(200)

  spot\_maps         String?            @db.VarChar(255)

  spot\_thumbnail    String?            @db.VarChar(100)

  spot\_rating       Decimal?           @default(0.00) @db.Decimal(3, 2)

  spot\_contact      String?            @db.VarChar(15)

  facilities facility\_tourism[]

  galleries         gallery[]

  operating\_hours   operating\_hours[]

  reviews           reviews[]

  category          category\_tourism?  @relation(fields: [category\_id], references: [category\_id], onUpdate: NoAction, map: "spot\_tourism\_ibfk\_1")

  village           village?           @relation(fields: [village\_id], references: [village\_id], onUpdate: NoAction, map: "spot\_tourism\_ibfk\_2")

  tickets           ticket[]

  @@index([category\_id], map: "category\_id")

  @@index([village\_id], map: "village\_id")

}

model ticket {

  ticket\_id           String               @id @db.Char(20)

  spot\_id             String?              @db.Char(10)

  ticket\_name         String?              @db.VarChar(100)

  ticket\_price        Decimal              @db.Decimal(10, 2)

  ticket\_desc         String?              @db.Text

  ticket\_stock        Int?                 @default(0)

  code                String               @db.VarChar(10)

  url\_qr              String               @db.Text

  spot                spot\_tourism?        @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade, onUpdate: NoAction, map: "ticket\_ibfk\_1")

  transaction\_details transaction\_detail[]

  @@index([spot\_id], map: "spot\_id")

}

model transaction {

  transaction\_id      Int                  @id @default(autoincrement())

  user\_id             String?              @db.Char(10)

  transaction\_code    String               @unique(map: "transaction\_code") @db.VarChar(50)

  total\_price         Decimal              @db.Decimal(10, 2)

  transaction\_status  TransactionStatus?   @default(pending)

  transaction\_date    DateTime?            @default(now()) @db.Timestamp(0)

  visit\_date          DateTime             @db.Date

  payments            payment[]

  user                users?               @relation(fields: [user\_id], references: [user\_id], onDelete: Cascade, onUpdate: NoAction, map: "transaction\_ibfk\_1")

  transaction\_details transaction\_detail[]

  @@index([user\_id], map: "user\_id")

}

model transaction\_detail {

  transaction\_detail\_id Int          @id @default(autoincrement())

  transaction\_id        Int?

  ticket\_id             String?      @db.Char(20)

  detail\_quantity       Int

  detail\_price          Decimal      @db.Decimal(10, 2)

  detail\_subtotal       Decimal?     @db.Decimal(10, 2)

  is\_used                TicketStatus @default(unverified) // Gunakan enum di sini

  transaction           transaction? @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade, onUpdate: NoAction, map: "transaction\_detail\_ibfk\_1")

  ticket                ticket?      @relation(fields: [ticket\_id], references: [ticket\_id], onDelete: Cascade, onUpdate: NoAction, map: "transaction\_detail\_ibfk\_2")

  @@index([ticket\_id], map: "ticket\_id")

  @@index([transaction\_id], map: "transaction\_id")

}

model users {

  user\_id      String        @id @db.Char(10)

  name         String        @db.VarChar(100)

  email        String        @unique(map: "email") @db.VarChar(100)

  password     String        @db.VarChar(255)

  phone\_number String?       @db.VarChar(20)

  address      String?       @db.Text

  role         Role

  reviews      reviews[]

  transactions transaction[]

}

model village {

  village\_id   String         @id @db.Char(10)

  village\_name String         @db.VarChar(100)

  village\_city String         @db.VarChar(100)

  spots        spot\_tourism[]

}

enum OperatingDay {

  Senin

  Selasa

  Rabu

  Kamis

  Jumat

  Sabtu

  Minggu

}

enum PaymentStatus {

  pending

  success

  failed

}

enum TransactionStatus {

  pending

  paid

  cancelled

}

enum Role {

  admin

  user

  tourism\_owner

}

enum TicketStatus {

  unverified

  verified

}

- data dari baturraden\_tourism.sql

-- phpMyAdmin SQL Dump

-- version 5.2.1

-- https://www.phpmyadmin.net/

--

-- Host: localhost:3306

-- Generation Time: May 13, 2025 at 01:05 AM

-- Server version: 8.4.3

-- PHP Version: 8.3.16

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Database: `baturraden\_tourism`

--

-- --------------------------------------------------------

--

-- Table structure for table `category\_tourism`

--

CREATE TABLE `category\_tourism` (

`category\_id` char(5) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`category\_name` varchar(20) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`category\_desc` varchar(150) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`category\_icon` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `category\_tourism`

--

INSERT INTO `category\_tourism` (`category\_id`, `category\_name`, `category\_desc`, `category\_icon`) VALUES

('C01', 'Air Terjun', 'Tempat wisata air terjun yang indah', NULL),

('C02', 'Bukit', 'Tempat wisata di perbukitan dengan pemandangan mempesona', NULL);

-- --------------------------------------------------------

--

-- Table structure for table `facility`

--

CREATE TABLE `facility` (

`facility\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`facility\_name` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`facility\_icon` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `facility`

--

INSERT INTO `facility` (`facility\_id`, `facility\_name`, `facility\_icon`) VALUES

('F01', 'Area Parkir', 'mermaid-diagram-2025-03-21-141112\_1746283216784.png'),

('F02', 'Toilet', 'activity\_diagram\_tiket\_1746663764851.png'),

('F03', 'Warung', 'WhatsApp\_Image\_2025-03-05\_at\_10.13\_1746664175020.jpg');

-- --------------------------------------------------------

--

-- Table structure for table `facility\_tourism`

--

CREATE TABLE `facility\_tourism` (

`facility\_tourism\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`spot\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`facility\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `gallery`

--

CREATE TABLE `gallery` (

`gallery\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`spot\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`gallery\_img` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`gallery\_caption` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `gallery`

--

INSERT INTO `gallery` (`gallery\_id`, `spot\_id`, `gallery\_img`, `gallery\_caption`) VALUES

('GVHKUJF3KC', 'S01', '/uploads/gallery/S01/1746754165522-mermaid-diagram-2025-03-21-141112.png', 'aSADFBNVCXZ');

-- --------------------------------------------------------

--

-- Table structure for table `operating\_hours`

--

CREATE TABLE `operating\_hours` (

`operating\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`spot\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`operating\_day` enum('Senin','Selasa','Rabu','Kamis','Jumat','Sabtu','Minggu') COLLATE utf8mb4\_unicode\_ci NOT NULL,

`hours\_open` time NOT NULL,

`hours\_closed` time NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `payment`

--

CREATE TABLE `payment` (

`payment\_id` int NOT NULL,

`transaction\_id` int DEFAULT NULL,

`method\_id` int DEFAULT NULL,

`reference\_number` varchar(50) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`payment\_amount` decimal(10,2) NOT NULL,

`service\_fee` decimal(10,2) NOT NULL,

`total\_payment` decimal(10,2) DEFAULT NULL,

`payment\_status` enum('pending','success','failed') COLLATE utf8mb4\_unicode\_ci DEFAULT 'pending',

`payment\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,

`expiration\_time` timestamp NULL DEFAULT NULL,

`response\_data` json DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `payment\_method`

--

CREATE TABLE `payment\_method` (

`method\_id` int NOT NULL,

`provider\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`method\_name` varchar(50) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`method\_code` varchar(50) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`method\_icon` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`service\_fee` decimal(10,2) NOT NULL DEFAULT '0.00',

`minimum\_payment` decimal(10,2) NOT NULL DEFAULT '0.00',

`maximum\_payment` decimal(10,2) NOT NULL,

`is\_active` tinyint(1) DEFAULT '1'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `payment\_provider`

--

CREATE TABLE `payment\_provider` (

`provider\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`provider\_name` varchar(50) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`provider\_desc` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`provider\_logo` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`is\_active` tinyint(1) DEFAULT '1'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `reviews`

--

CREATE TABLE `reviews` (

`reviews\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`spot\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`user\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`reviews\_rating` tinyint DEFAULT NULL,

`reviews\_desc` text COLLATE utf8mb4\_unicode\_ci,

`reviews\_created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

-- --------------------------------------------------------

--

-- Table structure for table `spot\_tourism`

--

CREATE TABLE `spot\_tourism` (

`spot\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`category\_id` char(5) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`village\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`spot\_name` varchar(60) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`spot\_desc` text COLLATE utf8mb4\_unicode\_ci,

`spot\_address` varchar(200) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`spot\_maps` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`spot\_thumbnail` varchar(100) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`spot\_rating` decimal(3,2) DEFAULT '0.00',

`spot\_contact` varchar(15) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `spot\_tourism`

--

INSERT INTO `spot\_tourism` (`spot\_id`, `category\_id`, `village\_id`, `spot\_name`, `spot\_desc`, `spot\_address`, `spot\_maps`, `spot\_thumbnail`, `spot\_rating`, `spot\_contact`) VALUES

('S01', 'C01', 'V001', 'Curug Sitalang', 'Air terjun yang indah', 'Desa Sitalang, Baturraden', 'https://maps.google.com', '/uploads/destination/1746631054053-WhatsApp Image 2025-04-21 at 11.26.54\_42e6387d.jpg', 0.00, ''),

('S02', 'C02', 'V002', 'Bukit Pancur', 'Pemandangan bukit yang mempesona', 'Jl. Bukit Pancur, Baturraden', 'https://maps.google.com', '/uploads/destination/1746664291108-IMG\_5833\_\_.PNG', 0.00, '');

-- --------------------------------------------------------

--

-- Table structure for table `ticket`

--

CREATE TABLE `ticket` (

`ticket\_id` char(20) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`spot\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`ticket\_name` varchar(100) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`ticket\_price` decimal(10,2) NOT NULL,

`ticket\_desc` text COLLATE utf8mb4\_unicode\_ci,

`ticket\_stock` int DEFAULT '0',

`code` varchar(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`url\_qr` text COLLATE utf8mb4\_unicode\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `ticket`

--

INSERT INTO `ticket` (`ticket\_id`, `spot\_id`, `ticket\_name`, `ticket\_price`, `ticket\_desc`, `ticket\_stock`, `code`, `url\_qr`) VALUES

('TICKET-1746759229025', 'S01', 'Tiket Curug Sitalang 2a', 2000.00, NULL, 100, 'QR003', 'data:image/png;base64,'),

('TICKET001', 'S01', 'Tiket Curug Sitalang', 5000.00, 'Tiket masuk ke Curug Sitalang', 100, 'QR001', ''),

('TICKET002', 'S02', 'Tiket Bukit Pancur', 70000.00, 'Tiket masuk ke Bukit Pancur', 50, 'QR002', '');

-- --------------------------------------------------------

--

-- Table structure for table `transaction`

--

CREATE TABLE `transaction` (

`transaction\_id` int NOT NULL,

`user\_id` char(10) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`transaction\_code` varchar(50) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`total\_price` decimal(10,2) NOT NULL,

`transaction\_status` enum('pending','paid','cancelled') COLLATE utf8mb4\_unicode\_ci DEFAULT 'pending',

`transaction\_date` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,

`visit\_date` date NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `transaction`

--

INSERT INTO `transaction` (`transaction\_id`, `user\_id`, `transaction\_code`, `total\_price`, `transaction\_status`, `transaction\_date`, `visit\_date`) VALUES

(2, 'U001', 'TX001', 50000.00, 'paid', '2025-05-07 11:39:13', '2025-05-08'),

(3, 'U002', 'TX002', 140000.00, 'paid', '2025-05-07 11:39:13', '2025-05-08');

-- --------------------------------------------------------

--

-- Table structure for table `transaction\_detail`

--

CREATE TABLE `transaction\_detail` (

`transaction\_detail\_id` int NOT NULL,

`transaction\_id` int DEFAULT NULL,

`ticket\_id` char(20) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`detail\_quantity` int NOT NULL,

`detail\_price` decimal(10,2) NOT NULL,

`detail\_subtotal` decimal(10,2) DEFAULT NULL,

`is\_used` enum('unverified','verified') COLLATE utf8mb4\_unicode\_ci NOT NULL DEFAULT 'unverified'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `transaction\_detail`

--

INSERT INTO `transaction\_detail` (`transaction\_detail\_id`, `transaction\_id`, `ticket\_id`, `detail\_quantity`, `detail\_price`, `detail\_subtotal`, `is\_used`) VALUES

(4, 3, 'TICKET001', 1, 50000.00, 50000.00, 'unverified'),

(5, 2, 'TICKET002', 2, 70000.00, 140000.00, 'verified');

-- --------------------------------------------------------

--

-- Table structure for table `users`

--

CREATE TABLE `users` (

`user\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`name` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`email` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`password` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`phone\_number` varchar(20) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`address` text COLLATE utf8mb4\_unicode\_ci,

`role` enum('admin','user','tourism\_owner') COLLATE utf8mb4\_unicode\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `users`

--

INSERT INTO `users` (`user\_id`, `name`, `email`, `password`, `phone\_number`, `address`, `role`) VALUES

('U001', 'John Doe', 'john@example.com', 'password123', '08123456789', 'Jalan Raya Baturraden', 'user'),

('U002', 'Jane Doe', 'jane@example.com', 'password456', '08987654321', 'Jalan Merdeka Baturraden', 'user'),

('U969431208', 'GreykoJezly 10', 'greykojezly@gmail.com', '-', NULL, NULL, 'user');

-- --------------------------------------------------------

--

-- Table structure for table `village`

--

CREATE TABLE `village` (

`village\_id` char(10) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`village\_name` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`village\_city` varchar(100) COLLATE utf8mb4\_unicode\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `village`

--

INSERT INTO `village` (`village\_id`, `village\_name`, `village\_city`) VALUES

('V001', 'Desa Sitalang', 'Purwokerto'),

('V002', 'Desa Pancur', 'Purwokerto');

-- --------------------------------------------------------

--

-- Table structure for table `\_prisma\_migrations`

--

CREATE TABLE `\_prisma\_migrations` (

`id` varchar(36) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`checksum` varchar(64) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`finished\_at` datetime(3) DEFAULT NULL,

`migration\_name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`logs` text COLLATE utf8mb4\_unicode\_ci,

`rolled\_back\_at` datetime(3) DEFAULT NULL,

`started\_at` datetime(3) NOT NULL DEFAULT CURRENT\_TIMESTAMP(3),

`applied\_steps\_count` int UNSIGNED NOT NULL DEFAULT '0'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Dumping data for table `\_prisma\_migrations`

--

INSERT INTO `\_prisma\_migrations` (`id`, `checksum`, `finished\_at`, `migration\_name`, `logs`, `rolled\_back\_at`, `started\_at`, `applied\_steps\_count`) VALUES

('0a5c0e90-8866-40b1-b9fd-5ee16b492a9e', 'e25e7b651d8ffc77d64e992b0ce11f8a7d5be25ea4e17432a8cc1648419051e9', '2025-05-07 04:30:42.540', '20250504082239\_n', NULL, NULL, '2025-05-07 04:30:42.267', 1),

('88087e28-15c6-4998-b366-afdcb12b6c3e', 'd0c04dd3d12f7b8ff6a94610f2e94005d5523a77a62bf232df4b52f3400f1fba', '2025-05-07 04:30:42.567', '20250505140618\_add\_used\_at\_to\_transaction\_detail', NULL, NULL, '2025-05-07 04:30:42.543', 1),

('8bec6d2d-e9f2-4bd1-8044-883ba6e59e0f', 'af8ddb0b1d1cad8c522edb15a5e595f8e99fa3d00152c2c5359f31f79e94903d', '2025-05-07 04:30:42.264', '20250428135537\_baturraden\_tourism', NULL, NULL, '2025-05-07 04:30:40.323', 1),

('915b1cfc-67f3-42b9-838b-ae16f933afc7', '7f063d42543076d9a70abba7b982e8cc75000d1eed49b6f101e867f3e03f14c0', '2025-05-07 11:43:27.989', '20250507114323\_change\_name\_is\_used', NULL, NULL, '2025-05-07 11:43:27.929', 1),

('a0bee08f-c2e6-4dd9-bc0c-aac47c17a98c', '0d14ac6c7c8283b4b9a185b31abe22b4bcd44187d0169d198f5899a1923cb976', '2025-05-07 04:30:42.594', '20250507035052\_add\_ticket\_status', NULL, NULL, '2025-05-07 04:30:42.570', 1);

--

-- Indexes for dumped tables

--

--

-- Indexes for table `category\_tourism`

--

ALTER TABLE `category\_tourism`

ADD PRIMARY KEY (`category\_id`);

--

-- Indexes for table `facility`

--

ALTER TABLE `facility`

ADD PRIMARY KEY (`facility\_id`);

--

-- Indexes for table `facility\_tourism`

--

ALTER TABLE `facility\_tourism`

ADD PRIMARY KEY (`facility\_tourism\_id`),

ADD KEY `facility\_id` (`facility\_id`),

ADD KEY `spot\_id` (`spot\_id`);

--

-- Indexes for table `gallery`

--

ALTER TABLE `gallery`

ADD PRIMARY KEY (`gallery\_id`),

ADD KEY `spot\_id` (`spot\_id`);

--

-- Indexes for table `operating\_hours`

--

ALTER TABLE `operating\_hours`

ADD PRIMARY KEY (`operating\_id`),

ADD KEY `spot\_id` (`spot\_id`);

--

-- Indexes for table `payment`

--

ALTER TABLE `payment`

ADD PRIMARY KEY (`payment\_id`),

ADD UNIQUE KEY `reference\_number` (`reference\_number`),

ADD KEY `method\_id` (`method\_id`),

ADD KEY `transaction\_id` (`transaction\_id`);

--

-- Indexes for table `payment\_method`

--

ALTER TABLE `payment\_method`

ADD PRIMARY KEY (`method\_id`),

ADD UNIQUE KEY `method\_code` (`method\_code`),

ADD KEY `provider\_id` (`provider\_id`);

--

-- Indexes for table `payment\_provider`

--

ALTER TABLE `payment\_provider`

ADD PRIMARY KEY (`provider\_id`);

--

-- Indexes for table `reviews`

--

ALTER TABLE `reviews`

ADD PRIMARY KEY (`reviews\_id`),

ADD KEY `spot\_id` (`spot\_id`),

ADD KEY `user\_id` (`user\_id`);

--

-- Indexes for table `spot\_tourism`

--

ALTER TABLE `spot\_tourism`

ADD PRIMARY KEY (`spot\_id`),

ADD KEY `category\_id` (`category\_id`),

ADD KEY `village\_id` (`village\_id`);

--

-- Indexes for table `ticket`

--

ALTER TABLE `ticket`

ADD PRIMARY KEY (`ticket\_id`),

ADD KEY `spot\_id` (`spot\_id`);

--

-- Indexes for table `transaction`

--

ALTER TABLE `transaction`

ADD PRIMARY KEY (`transaction\_id`),

ADD UNIQUE KEY `transaction\_code` (`transaction\_code`),

ADD KEY `user\_id` (`user\_id`);

--

-- Indexes for table `transaction\_detail`

--

ALTER TABLE `transaction\_detail`

ADD PRIMARY KEY (`transaction\_detail\_id`),

ADD KEY `ticket\_id` (`ticket\_id`),

ADD KEY `transaction\_id` (`transaction\_id`);

--

-- Indexes for table `users`

--

ALTER TABLE `users`

ADD PRIMARY KEY (`user\_id`),

ADD UNIQUE KEY `email` (`email`);

--

-- Indexes for table `village`

--

ALTER TABLE `village`

ADD PRIMARY KEY (`village\_id`);

--

-- Indexes for table `\_prisma\_migrations`

--

ALTER TABLE `\_prisma\_migrations`

ADD PRIMARY KEY (`id`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `payment`

--

ALTER TABLE `payment`

MODIFY `payment\_id` int NOT NULL AUTO\_INCREMENT;

--

-- AUTO\_INCREMENT for table `payment\_method`

--

ALTER TABLE `payment\_method`

MODIFY `method\_id` int NOT NULL AUTO\_INCREMENT;

--

-- AUTO\_INCREMENT for table `transaction`

--

ALTER TABLE `transaction`

MODIFY `transaction\_id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;

--

-- AUTO\_INCREMENT for table `transaction\_detail`

--

ALTER TABLE `transaction\_detail`

MODIFY `transaction\_detail\_id` int NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

--

-- Constraints for dumped tables

--

--

-- Constraints for table `facility\_tourism`

--

ALTER TABLE `facility\_tourism`

ADD CONSTRAINT `facility\_tourism\_ibfk\_1` FOREIGN KEY (`spot\_id`) REFERENCES `spot\_tourism` (`spot\_id`) ON DELETE CASCADE,

ADD CONSTRAINT `facility\_tourism\_ibfk\_2` FOREIGN KEY (`facility\_id`) REFERENCES `facility` (`facility\_id`) ON DELETE CASCADE;

--

-- Constraints for table `gallery`

--

ALTER TABLE `gallery`

ADD CONSTRAINT `gallery\_ibfk\_1` FOREIGN KEY (`spot\_id`) REFERENCES `spot\_tourism` (`spot\_id`) ON DELETE CASCADE;

--

-- Constraints for table `operating\_hours`

--

ALTER TABLE `operating\_hours`

ADD CONSTRAINT `operating\_hours\_ibfk\_1` FOREIGN KEY (`spot\_id`) REFERENCES `spot\_tourism` (`spot\_id`) ON DELETE CASCADE;

--

-- Constraints for table `payment`

--

ALTER TABLE `payment`

ADD CONSTRAINT `payment\_ibfk\_1` FOREIGN KEY (`transaction\_id`) REFERENCES `transaction` (`transaction\_id`) ON DELETE CASCADE,

ADD CONSTRAINT `payment\_ibfk\_2` FOREIGN KEY (`method\_id`) REFERENCES `payment\_method` (`method\_id`) ON DELETE SET NULL;

--

-- Constraints for table `payment\_method`

--

ALTER TABLE `payment\_method`

ADD CONSTRAINT `payment\_method\_ibfk\_1` FOREIGN KEY (`provider\_id`) REFERENCES `payment\_provider` (`provider\_id`) ON DELETE SET NULL;

--

-- Constraints for table `reviews`

--

ALTER TABLE `reviews`

ADD CONSTRAINT `reviews\_ibfk\_1` FOREIGN KEY (`spot\_id`) REFERENCES `spot\_tourism` (`spot\_id`) ON DELETE CASCADE,

ADD CONSTRAINT `reviews\_ibfk\_2` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE;

--

-- Constraints for table `spot\_tourism`

--

ALTER TABLE `spot\_tourism`

ADD CONSTRAINT `spot\_tourism\_ibfk\_1` FOREIGN KEY (`category\_id`) REFERENCES `category\_tourism` (`category\_id`) ON DELETE SET NULL,

ADD CONSTRAINT `spot\_tourism\_ibfk\_2` FOREIGN KEY (`village\_id`) REFERENCES `village` (`village\_id`) ON DELETE SET NULL;

--

-- Constraints for table `ticket`

--

ALTER TABLE `ticket`

ADD CONSTRAINT `ticket\_ibfk\_1` FOREIGN KEY (`spot\_id`) REFERENCES `spot\_tourism` (`spot\_id`) ON DELETE CASCADE;

--

-- Constraints for table `transaction`

--

ALTER TABLE `transaction`

ADD CONSTRAINT `transaction\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`) ON DELETE CASCADE;

--

-- Constraints for table `transaction\_detail`

--

ALTER TABLE `transaction\_detail`

ADD CONSTRAINT `transaction\_detail\_ibfk\_1` FOREIGN KEY (`transaction\_id`) REFERENCES `transaction` (`transaction\_id`) ON DELETE CASCADE,

ADD CONSTRAINT `transaction\_detail\_ibfk\_2` FOREIGN KEY (`ticket\_id`) REFERENCES `ticket` (`ticket\_id`) ON DELETE CASCADE;

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

- lib/pdf/generateTicketPdf.ts

import { PDFDocument } from 'pdf-lib';

export async function generateTicketPdf(

  transactionCode: string,

  qrCodeUrl: string,

  additionalData: string

): Promise<Uint8Array> {

  const [userName, destinationName, date] = additionalData.split(',');

  const formattedDate = new Date(date).toLocaleString('id-ID', { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric', hour: 'numeric', minute: 'numeric', second: 'numeric' });

  // Buat PDF Document baru

  const pdfDoc = await PDFDocument.create();

  // Tambahkan halaman

  const page = pdfDoc.addPage([600, 400]);

  // Menambahkan teks ke halaman

  page.drawText(`Transaction Code: ${transactionCode}`, { x: 50, y: 350, size: 18 });

  page.drawText(`Name: ${userName}`, { x: 50, y: 320, size: 18 });

  page.drawText(`Destination: ${destinationName}`, { x: 50, y: 290, size: 18 });

  page.drawText(`Date: ${formattedDate}`, { x: 50, y: 260, size: 18 });

  // Generate QR Code dan embed ke dalam PDF

  const qrImage = await pdfDoc.embedPng(qrCodeUrl);

  const qrDims = qrImage.scale(0.70); // Skalakan QR code

  // Tentukan lokasi QR code di pojok kiri bawah

  page.drawImage(qrImage, {

    x: 350,  // Posisi X QR code di pojok kiri

    y: 50,  // Posisi Y QR code di pojok bawah

    width: qrDims.width,  // Ukuran lebar QR Code

    height: qrDims.height,  // Ukuran tinggi QR Code

  });

  // Simpan PDF dalam bentuk byte array

  const pdfBytes = await pdfDoc.save();

  return pdfBytes; // Mengembalikan PDF dalam bentuk byte array

}

- lib/qrcode.ts

import \* as QRCode from 'qrcode';  // Tetap menggunakan import seperti yang direkomendasikan

export async function generateQRCode(data: string): Promise<string> {

  const options = {

    errorCorrectionLevel: 'H',  // Level koreksi error

    width: 300,                 // Ukuran QR code

  };

  try {

    // Menggunakan QRCode.toDataURL dengan dua argumen yang benar

    const qrCodeUrl = await QRCode.toDataURL(data, options);

    console.log("Generated QR Code URL:", qrCodeUrl); // Debug QR Code yang dihasilkan

    return qrCodeUrl;

  } catch (err) {

    console.error("Error generating QR Code:", err); // Log jika terjadi error

    throw new Error(`Failed to generate QR code: ${err}`);

  }

}

- utils.ts

import { clsx, type ClassValue } from "clsx"

import { twMerge } from "tailwind-merge"

export function cn(...inputs: ClassValue[]) {

  return twMerge(clsx(inputs))

}

- pages/api/admin/payment/route.ts

import { NextApiRequest, NextApiResponse } from 'next';

import prisma from '@lib/prisma';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  if (req.method === 'GET') {

    try {

      const payments = await prisma.payment.findMany({

        orderBy: {

          payment\_time: 'desc',

        },

        include: {

          transaction: true,

          method: true,

        },

      });

      res.status(200).json(payments);

    } catch (error) {

      console.error('Error fetching payments:', error);

      res.status(500).json({ error: 'Failed to fetch payments' });

    }

  } else {

    res.status(405).json({ error: 'Method Not Allowed' });

  }

}

- pages/api/admin/payment/status.ts

// pages/api/admin/payment/status.ts

import { NextApiRequest, NextApiResponse } from 'next';

import \* as midtransClient from 'midtrans-client';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  if (req.method !== 'GET') {

    return res.status(405).json({ error: 'Method Not Allowed' });

  }

  const { order\_id } = req.query;

  // Setup Midtrans CoreApi

  const snap = new midtransClient.Snap({

    isProduction: false, // Set to true in production

    serverKey: process.env.MIDTRANS\_SERVER\_KEY, // Replace with your Midtrans server key

    clientKey: process.env.MIDTRANS\_CLIENT\_KEY, // Replace with your Midtrans client key

  });

  try {

    // Check the status of the transaction using the order\_id

    const statusResponse = await snap.transaction.status(order\_id as string);

    // Send the transaction status response back

    res.status(200).json(statusResponse);

  } catch (error) {

    console.error('Error fetching status:', error);

    res.status(500).json({ error: 'Failed to fetch payment status' });

  }

}

- pages/api/admin/transaction/detail/[id].ts

import { NextApiRequest, NextApiResponse } from 'next'

import prisma from '@lib/prisma'

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  const { id } = req.query

  if (req.method === 'GET') {

    try {

      const details = await prisma.transaction\_detail.findMany({

        where: { transaction\_id: Number(id) },

        include: {

          ticket: {

            select: {

              ticket\_name: true,

              ticket\_price: true

            }

          }

        }

      })

      res.status(200).json(details)

    } catch (error) {

      console.error('Error:', error)

      res.status(500).json({ error: 'Gagal mengambil detail transaksi' })

    }

  } else {

    res.status(405).json({ error: 'Method Not Allowed' })

  }

}

- pages/api/admin/transaction/update/[id].ts

import { NextApiRequest, NextApiResponse } from 'next';

import prisma from '@lib/prisma';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  const { id } = req.query;

  if (req.method === 'PUT') {

    const { transaction\_status } = req.body;

    try {

      const updatedTransaction = await prisma.transaction.update({

        where: {

          transaction\_id: Number(id),

        },

        data: {

          transaction\_status,

        },

      });

      res.status(200).json(updatedTransaction);

    } catch (error) {

      console.error('Error updating transaction:', error);

      res.status(500).json({ error: 'Failed to update transaction status' });

    }

  } else {

    res.status(405).json({ error: 'Method Not Allowed' });

  }

}

- pages/api/admin/transaction/callback.ts

import { NextApiRequest, NextApiResponse } from 'next';

import prisma from '@lib/prisma';

import { generateQRCode } from '@lib/qrcode';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  const { order\_id, transaction\_status } = req.body;

  if (transaction\_status === 'settlement' || transaction\_status === 'capture') {

    const transaction = await prisma.transaction.update({

      where: { transaction\_code: order\_id },

      data: { transaction\_status: 'paid' },

      include: {

        transaction\_details: true,

      },

    });

    for (const detail of transaction.transaction\_details) {

      const qrData = `https://baturraden-tourism.com/use/${detail.transaction\_detail\_id}`;

      const qrImage = await generateQRCode(qrData);

      await prisma.transaction\_detail.update({

        where: { transaction\_detail\_id: detail.transaction\_detail\_id },

        data: {

          qr\_url: qrImage,

        },

      });

    }

    return res.status(200).json({ message: 'Transaction updated & QR generated' });

  }

  return res.status(200).json({ message: 'Ignored non-paid status' });

}

- pages/api/admin/transaction/route.ts

import { NextApiRequest, NextApiResponse } from 'next';

import prisma from '@lib/prisma';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  if (req.method === 'GET') {

    try {

      const transactions = await prisma.transaction.findMany({

        include: {

          user: {

            select: {

              name: true

            }

          }

        },

        orderBy: {

          transaction\_date: 'desc',

        },

      });

      res.status(200).json(transactions);

    } catch (error) {

      console.error('Error fetching transactions:', error);

      res.status(500).json({ error: 'Failed to fetch transactions' });

    }

  } else {

    res.status(405).json({ error: 'Method Not Allowed' });

  }

}

- pages/api/payment/callback.ts

// pages/api/admin/payment/callback.ts

import { NextApiRequest, NextApiResponse } from 'next'

interface MidtransCallbackResponse {

  transaction\_status: string;

  order\_id: string;

  payment\_type: string;

}

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  if (req.method !== 'POST') return res.status(405).json({ error: 'Method Not Allowed' })

  const response: MidtransCallbackResponse = req.body;

  const isValid = validateMidtransSignature(response)

  if (isValid) {

    res.status(200).json({ status: 'success' })

  } else {

    res.status(400).json({ error: 'Invalid callback data' })

  }

}

const validateMidtransSignature = (response: MidtransCallbackResponse) => {

  console.log(response);

  return true;

}

- pages/api/payment/createTransaction.ts

import { NextApiRequest, NextApiResponse } from 'next';

import \* as midtransClient from 'midtrans-client';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  if (req.method !== 'POST') {

    return res.status(405).json({ error: 'Method Not Allowed' });

  }

  const { amount, order\_id } = req.body;

  try {

    const coreApi = new midtransClient.CoreApi({

      serverKey: process.env.MIDTRANS\_SERVER\_KEY, // Gunakan server key

      clientKey: process.env.MIDTRANS\_CLIENT\_KEY, // Gunakan client key

      isProduction: false, // Gunakan false untuk sandbox

    });

    const parameter = {

      payment\_type: 'credit\_card',

      transaction\_details: {

        order\_id: order\_id,

        gross\_amount: amount,

      },

      credit\_card: {

        secure: true,

        token\_id: 'token-id-dari-tokenisasi',  // Ini adalah token yang Anda dapatkan setelah tokenisasi di frontend

      },

    };

    const chargeResponse = await coreApi.charge(parameter);

    res.status(200).json(chargeResponse); // Mengirimkan hasil dari Midtrans

  } catch (error) {

    console.error('Payment error:', error);

    res.status(500).json({ error: 'Payment processing failed' });

  }

}

console.log('Server Key:', process.env.MIDTRANS\_SERVER\_KEY);

console.log('Client Key:', process.env.MIDTRANS\_CLIENT\_KEY);

- pages/api/payment/tokenize.ts

import type { NextApiRequest, NextApiResponse } from 'next';

import MidtransClient from 'midtrans-client';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  if (req.method !== 'POST') return res.status(405).json({ error: 'Method Not Allowed' });

  const { order\_id, amount } = req.body;

  const snap = new MidtransClient.Snap({

    isProduction: false,

    serverKey: process.env.MIDTRANS\_SERVER\_KEY!, // backend

  });

  const parameter = {

    transaction\_details: {

      order\_id,

      gross\_amount: amount,

    },

    credit\_card: {

      secure: true,

    },

  };

  try {

    const transaction = await snap.createTransaction(parameter);

    res.status(200).json({ token\_id: transaction.token });

  } catch (error) {

    console.error("Error creating Snap token:", error);

    res.status(500).json({ error: "Server error" });

  }

}

- pages/api/qrcode/[code].ts

// pages/api/qrcode/[code].ts

import type { NextApiRequest, NextApiResponse } from "next";

import QRCode from "qrcode";

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  const { code } = req.query;

  if (typeof code !== "string" || !code.trim()) {

    return res.status(400).json({ error: "Kode tidak valid" });

  }

  try {

    const qrDataUrl = await QRCode.toDataURL(code);

    const base64Data = qrDataUrl.replace(/^data:image\/png;base64,/, "");

    const buffer = Buffer.from(base64Data, "base64");

    res.setHeader("Content-Type", "image/png");

    res.setHeader("Content-Disposition", `inline; filename="${code}.png"`);

    res.status(200).send(buffer);

  } catch (error) {

    console.error("Gagal generate QR", error);

    res.status(500).json({ error: "Gagal generate QR code" });

  }

}

- pages/api/ticket/pdf.ts

import { NextApiRequest, NextApiResponse } from 'next';

import { generateTicketPdf } from '@lib/pdf/generateTicketPdf'; // Import fungsi generate PDF

import prisma from '@lib/prisma';

import { generateQRCode } from '@lib/qrcode';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  const { id } = req.query; // ID transaksi QR Code

  if (typeof id !== 'string') {

    return res.status(400).json({ error: 'ID tidak valid' });

  }

  try {

    // Ambil data transaksi dari database

    const transaction = await prisma.transaction.findUnique({

      where: { transaction\_code: id },

      include: {

        transaction\_details: {

          include: {

            ticket: true, // Sertakan relasi dengan ticket di sini

          },

        },

      },

    });

    if (!transaction) {

      return res.status(404).json({ error: 'Kode transaksi tidak ditemukan' });

    }

    // Menyiapkan data untuk PDF

    const userName = transaction.user\_id; // Ambil data pengguna

    const destinationName = transaction.transaction\_details[0]?.ticket?.ticket\_name; // Ambil nama tiket

    const date = transaction.transaction\_date;

    // Membuat URL QR Code untuk transaksi

    const qrCodeUrl = await generateQRCode(id); // Menghasilkan QR Code untuk transaksi

    // Generate PDF

    const pdfBytes = await generateTicketPdf(id, qrCodeUrl, `${userName},${destinationName},${date}`);

    // Set header dengan nama file dinamis (menggunakan nomor transaksi)

    res.setHeader('Content-Type', 'application/pdf');

    res.setHeader('Content-Disposition', `attachment; filename="tiket-${id}.pdf"`); // Ganti nama file sesuai ID transaksi

    // Kirimkan PDF dalam format buffer

    res.end(Buffer.from(pdfBytes)); // Gunakan res.end() untuk mengirim buffer PDF

  } catch (error) {

    console.error('[PDF ERROR]', error);

    res.status(500).json({ error: 'Gagal menghasilkan PDF' });

  }

}

- src/app/admin/payment/page.tsx

import { PaymentTable } from "@/components/admin/payment/PaymentTable"

export default function PaymentPage() {

  return (

    <div className="p-6">

      <h1 className="text-2xl font-bold mb-4">Data Pembayaran</h1>

      <PaymentTable />

    </div>

  )

}

- src/app/admin/transaction/page.tsx

import { TransactionTable } from "@/components/admin/transaction/TransactionTable"

export default function TransactionPage() {

  return (

    <div className="p-6">

      <h1 className="text-2xl font-bold mb-4">Transaksi</h1>

      <TransactionTable />

    </div>

  )

}

- src/app/user/payment/page.tsx

"use client";

import { useState, useEffect } from "react";

import type { MidtransTransactionResult } from "@/types/midtrans";

export default function CheckoutPage() {

  useEffect(() => {

    const script = document.createElement('script');

    script.src = 'https://app.sandbox.midtrans.com/snap/snap.js';

    script.setAttribute('data-client-key', process.env.NEXT\_PUBLIC\_MIDTRANS\_CLIENT\_KEY!);

    document.body.appendChild(script);

    return () => {

      document.body.removeChild(script);

    };

  }, []);

  const [loading, setLoading] = useState(false);

  const handlePayment = async () => {

    setLoading(true);

    const ticket\_id = localStorage.getItem("selected\_ticket\_id");

    if (!ticket\_id) {

      alert("Tiket belum dipilih");

      return;

    }

    const order\_id = `ORDER-${Date.now()}`;

    const response = await fetch("/api/payment/tokenize", {

      method: "POST",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify({ order\_id, ticket\_id }), // kirim ticket\_id

    });

    try {

      const response = await fetch("/api/payment/tokenize", {

        method: "POST",

        headers: {

          "Content-Type": "application/json",

        },

        body: JSON.stringify({ order\_id, amount }),

      });

      const data = await response.json();

      // Debugging response

      console.log('Response from API:', data);

      if (data.token\_id) {

        // Snap will auto popup

        window.snap.pay(data.token\_id, {

          onSuccess: (result: MidtransTransactionResult) => {

            console.log("Payment success:", result);

          },

          onPending: (result: MidtransTransactionResult) => {

            console.log("Payment pending:", result);

          },

          onError: (error: Error) => {

            console.error("Payment error:", error);

          },

          onClose: () => {

            console.log("User closed payment popup");

          },

        });

      } else {

        console.error("No token received:", data);

      }

    } catch (error) {

      console.error("Failed to fetch payment token:", error);

    }

    if (typeof window.snap === "undefined") {

      console.error("Midtrans snap.js not loaded yet");

      return;

    }

    setLoading(false);

  };

  return (

    <div className="p-8">

      <h1 className="text-2xl font-bold mb-4">Checkout Tiket Wisata</h1>

      <button

        className="bg-blue-600 text-white px-4 py-2 rounded"

        onClick={handlePayment}

        disabled={loading}

      >

        {loading ? "Processing..." : "Bayar Sekarang"}

      </button>

    </div>

  );

}

src/app/user/destination/[slug]/page.tsx

import { fetchDestinationDetail } from "@/modules/destination/controller/fetchDetail";

import { DestinationData } from "@/types/destination"; // Import tipe data destinasi

import MapEmbed from "@/components/user/destination/MapEmbed";

import FacilityList from "@/components/user/destination/FacilityList";

import GalleryList from "@/components/user/destination/GalleryList";

import OperatingHoursList from "@/components/user/destination/OperatingHoursList";

import ReviewList from "@/components/user/destination/ReviewList";

import TicketList from "@/components/user/destination/TicketList";

import { notFound } from "next/navigation";

import { Facility } from "@/types/facility";

import { Gallery } from "@/types/gallery";

import { OperatingHour } from "@/types/operating-hours";

import { Review } from "@/types/review";

import { Ticket } from "@/types/ticket";

import { Decimal } from "@prisma/client/runtime/library";

export const dynamic = "force-dynamic";

export const fetchCache = "force-no-store";

export default async function DetailDestinationPage({ params }: { params: { slug: string } }) {

const data: DestinationData | null = await fetchDestinationDetail(params.slug);

if (!data) return notFound();

// Menangani data dengan tipe DestinationData

const filteredFacilities = data.facilities

.filter((f: { facility: Facility | null }) => f.facility !== null)

.map(f => ({

facility: {

facility\_name: f.facility?.facility\_name ?? '',

facility\_icon: f.facility?.facility\_icon ?? null

}

}));

const galleries: Gallery[] = data.galleries.map((gallery: { gallery\_id: string, spot\_id: string | null, gallery\_img: string | null, gallery\_caption: string | null }) => ({

gallery\_id: gallery.gallery\_id,

spot\_id: gallery.spot\_id ?? '',

gallery\_img: gallery.gallery\_img ?? '',

gallery\_caption: gallery.gallery\_caption ?? 'No caption',

}));

const operatingHours: OperatingHour[] = data.operating\_hours.map((operating: { operating\_id: string, spot\_id: string | null, operating\_day: string, hours\_open: Date, hours\_closed: Date }) => ({

operating\_id: operating.operating\_id,

spot\_id: operating.spot\_id,

operating\_day: operating.operating\_day,

hours\_open: new Date(operating.hours\_open).toLocaleTimeString(),

hours\_closed: new Date(operating.hours\_closed).toLocaleTimeString(),

}));

const reviews: Review[] = data.reviews.map((review: { reviews\_id: string, spot\_id: string | null, user\_id: string | null, reviews\_rating: number | null, reviews\_desc: string | null, reviews\_created\_at: Date | null }) => ({

reviews\_id: review.reviews\_id,

spot\_id: review.spot\_id,

user\_id: review.user\_id,

reviews\_rating: review.reviews\_rating ?? 0,

reviews\_desc: review.reviews\_desc ?? 'No description',

reviews\_created\_at: review.reviews\_created\_at ? review.reviews\_created\_at.toISOString() : '',

}));

const tickets: Ticket[] = data.tickets.map((ticket: { ticket\_id: string, spot\_id: string | null, ticket\_name: string | null, ticket\_price: Decimal, ticket\_desc: string | null, ticket\_stock: number | null, code: string, url\_qr: string }) => ({

ticket\_id: ticket.ticket\_id,

spot\_id: ticket.spot\_id ?? '',

ticket\_name: ticket.ticket\_name ?? 'No ticket name',

ticket\_price: parseFloat(ticket.ticket\_price.toString()),

ticket\_desc: ticket.ticket\_desc ?? 'No description',

ticket\_stock: ticket.ticket\_stock ?? 0,

code: ticket.code,

url\_qr: ticket.url\_qr,

}));

return (

<div className="max-w-5xl mx-auto px-4 py-8 space-y-6">

<h1 className="text-3xl font-bold">{data.spot\_name}</h1>

<p className="text-gray-600">{data.spot\_desc}</p>

<div>

<h2 className="text-lg font-medium mb-1">Alamat</h2>

<p className="text-sm text-gray-700">{data.spot\_address}</p>

</div>

{data.spot\_maps && <MapEmbed mapsUrl={data.spot\_maps} />}

<div>

<h2 className="text-lg font-medium mb-1">Kategori Wisata</h2>

{data.category ? <div>{data.category.category\_name}</div> : <div>No category available</div>}

</div>

{filteredFacilities.length > 0 && <FacilityList facilities={filteredFacilities} />}

{galleries.length > 0 && <GalleryList items={galleries} />}

{operatingHours.length > 0 && <OperatingHoursList hours={operatingHours} />}

{reviews.length > 0 && <ReviewList reviews={reviews} />}

{tickets.length > 0 && <TicketList tickets={tickets} />}

</div>

);

}

- src/components/admin/payment/PaymentButton.tsx, nantinya akan dipindah ke user

// src/components/admin/payment/PaymentButton.tsx

import { useState } from 'react'

import { Button } from '@/components/ui/button'

export default function PaymentButton({ orderId, amount }: { orderId: string, amount: number }) {

  const [isLoading, setIsLoading] = useState(false)

  const handlePayment = async () => {

    setIsLoading(true)

    const response = await fetch('/api/admin/payment/createTransaction', {

      method: 'POST',

      headers: { 'Content-Type': 'application/json' },

      body: JSON.stringify({ order\_id: orderId, amount }),

    })

    const data = await response.json()

    if (data.redirect\_url) {

      window.location.href = data.redirect\_url

    } else {

      alert('Gagal mengarahkan ke halaman pembayaran')

    }

    setIsLoading(false)

  }

  return (

    <Button onClick={handlePayment} disabled={isLoading}>

      {isLoading ? 'Proses Pembayaran...' : 'Bayar Sekarang'}

    </Button>

  )

}

- src/components/admin/payment/PaymentTable.tsx

'use client'

import { useEffect, useState } from "react"

import { ColumnDef } from "@tanstack/react-table"

import { DataTable } from "@/components/ui/data-table"

import { Badge } from "@/components/ui/badge"

type Payment = {

  payment\_id: number

  transaction\_id: number

  reference\_number: string

  payment\_amount: number

  service\_fee: number

  total\_payment: number

  payment\_status: string

  payment\_time: string

}

export function PaymentTable() {

  const [data, setData] = useState<Payment[]>([])

  useEffect(() => {

    fetch("/api/admin/payment/route")

      .then(res => res.json())

      .then(setData)

  }, [])

  const columns: ColumnDef<Payment>[] = [

    {

      header: "ID",

      accessorKey: "payment\_id",

    },

    {

      header: "Transaksi",

      accessorKey: "transaction\_id",

    },

    {

      header: "Ref",

      accessorKey: "reference\_number",

    },

    {

      header: "Nominal",

      accessorKey: "payment\_amount",

      cell: ({ row }) => `Rp ${String(row.getValue("payment\_amount"))}`,

    },

    {

      header: "Biaya Layanan",

      accessorKey: "service\_fee",

      cell: ({ row }) => `Rp ${String(row.getValue("service\_fee"))}`,

    },

    {

      header: "Total",

      accessorKey: "total\_payment",

      cell: ({ row }) => `Rp ${String(row.getValue("total\_payment"))}`,

    },

    {

      header: "Status",

      accessorKey: "payment\_status",

      cell: ({ row }) => {

        const status = String(row.getValue("payment\_status"))

        const color =

          status === "success" ? "default" :

            status === "pending" ? "outline" :

              "destructive"

        return <Badge variant={color}>{status}</Badge>

      },

    },

    {

      header: "Waktu",

      accessorKey: "payment\_time",

      cell: ({ row }) =>

        new Date(String(row.getValue("payment\_time"))).toLocaleString(),

    },

  ]

  return <DataTable columns={columns} data={data} />

}

- src/components/admin/transaction/TransactionColumns.tsx

'use client'

import { ColumnDef } from "@tanstack/react-table"

import { Badge } from "@/components/ui/badge"

import { Transaction } from "./TransactionTable"

import { TransactionStatusSelect } from "./TransactionStatusSelect"

import { TransactionDetailDialog } from "./TransactionDetailDialog"

export const columns = (

  fetchData: (status?: string) => void,

  currentFilter: string

): ColumnDef<Transaction>[] => [

    { header: "ID", accessorKey: "transaction\_id" },

    { header: "Kode", accessorKey: "transaction\_code" },

    { header: "Nama Pengguna", accessorKey: "user.name" },

    { header: "User ID", accessorKey: "user\_id" },

    { header: "Spot ID", accessorKey: "spot\_id" },

    {

      header: "Total",

      accessorKey: "total\_price",

      cell: ({ row }) => `Rp ${Number(row.getValue("total\_price")).toLocaleString()}`

    },

    {

      header: "Status",

      accessorKey: "transaction\_status",

      cell: ({ row }) => (

        <Badge variant={

          row.getValue("transaction\_status") === "pending" ? "outline" :

            row.getValue("transaction\_status") === "paid" ? "default" :

              "destructive"

        }>

          {row.getValue("transaction\_status")}

        </Badge>

      )

    },

    {

      header: "Detail",

      cell: ({ row }) => (

        <TransactionDetailDialog transactionId={Number(row.original.transaction\_id)} />

      )

    },

    {

      header: "Tanggal Kunjungan",

      accessorKey: "visit\_date",

      cell: ({ row }) => new Date(row.getValue("visit\_date")).toLocaleDateString(),

    },

    {

      header: "Waktu Transaksi",

      accessorKey: "transaction\_date", // sesuai field di database dan prisma

      cell: ({ row }) => {

        const date = row.getValue("transaction\_date")

        return date

          ? new Date(date as string).toLocaleString("id-ID", {

            dateStyle: "medium",

            timeStyle: "short",

          })

          : "-"

      },

    },

    {

      header: "Ubah Status",

      cell: ({ row }) => (

        <TransactionStatusSelect

          transactionId={row.original.transaction\_id}

          currentStatus={row.original.transaction\_status}

          onUpdate={() => fetchData(currentFilter)}

        />

      )

    }

  ]

- src/components/admin/transaction/TransactionDetailDialog.tsx

'use client'

import { useEffect, useState } from "react"

import { Dialog, DialogContent, DialogTrigger, DialogTitle } from "@/components/ui/dialog"

import { Button } from "@/components/ui/button"

import Image from "next/image"

type Props = {

  transactionId: number

}

type Detail = {

  detail\_quantity: number

  detail\_subtotal: number

  ticket: {

    ticket\_name: string

    ticket\_price: number

    url\_qr?: string

  }

}

export function TransactionDetailDialog({ transactionId }: Props) {

  const [details, setDetails] = useState<Detail[]>([])

  useEffect(() => {

    fetch(`/api/admin/transaction/detail/${transactionId}`)

      .then(res => res.json())

      .then(setDetails)

  }, [transactionId])

  return (

    <Dialog>

      <DialogTrigger asChild>

        <Button variant="outline" size="sm">Lihat Detail</Button>

      </DialogTrigger>

      <DialogContent>

        <DialogTitle>Detail Tiket Transaksi</DialogTitle>

        <div className="space-y-4">

          {details.map((item, index) => (

            <div key={index} className="border p-3 rounded shadow-sm text-sm">

              <p><strong>Nama Tiket:</strong> {item.ticket.ticket\_name}</p>

              <p><strong>Harga:</strong> Rp {Number(item.ticket.ticket\_price).toLocaleString()}</p>

              <p><strong>Jumlah:</strong> {item.detail\_quantity}</p>

              <p><strong>Subtotal:</strong> Rp {Number(item.detail\_subtotal).toLocaleString()}</p>

              {item.ticket.url\_qr && (

                <div className="mt-2">

                  <p className="mb-1 text-xs text-muted-foreground">QR Code:</p>

                  <Image

                    src={item.ticket.url\_qr || "/fallback-qr.png"}

                    alt={`QR for ${item.ticket.ticket\_name}`}

                    width={128}

                    height={128}

                    className="border rounded"

                  />

                </div>

              )}

            </div>

          ))}

        </div>

      </DialogContent>

    </Dialog>

  )

}

- src/components/admin/transaction/TransactionStatusSelect.tsx

'use client'

import { useState } from "react"

type Props = {

  transactionId: string

  currentStatus: string

  onUpdate: () => void

}

export function TransactionStatusSelect({ transactionId, currentStatus, onUpdate }: Props) {

  const [status, setStatus] = useState(currentStatus)

  const handleChange = async (e: React.ChangeEvent<HTMLSelectElement>) => {

    const newStatus = e.target.value

    setStatus(newStatus)

    await fetch(`/api/admin/transaction/update/${transactionId}`, {

      method: 'PUT',

      headers: {

        'Content-Type': 'application/json',

      },

      body: JSON.stringify({ transaction\_status: newStatus }),

    })

    onUpdate()

  }

  return (

    <select value={status} onChange={handleChange} className="border p-1 rounded text-sm">

      <option value="pending">Pending</option>

      <option value="paid">Paid</option>

      <option value="cancelled">Cancelled</option>

    </select>

  )

}

- src/components/admin/transaction/TransactionTable.tsx

'use client'

import { useEffect, useState } from "react"

import { DataTable } from "@/components/ui/data-table"

import { TransactionToolbar } from "./TransactionToolbar"

import { columns } from "./TransactionColumns"

export type Transaction = {

  transaction\_id: string

  user\_id: string

  spot\_id: string

  transaction\_code: string

  transaction\_status: string

  total\_price: number

  transaction\_time: string

  visit\_date: string

  user?: {

    name: string

  }

}

export function TransactionTable() {

  const [data, setData] = useState<Transaction[]>([])

  const [statusFilter, setStatusFilter] = useState("")

  const fetchData = (status = "") => {

    const url =

      status && status !== "all"

        ? `/api/admin/transaction/route?status=${status}`

        : `/api/admin/transaction/route`

    fetch(url)

      .then(res => res.json())

      .then(setData)

  }

  useEffect(() => {

    fetchData(statusFilter)

  }, [statusFilter])

  return (

    <div className="space-y-4">

      <TransactionToolbar

        statusFilter={statusFilter}

        setStatusFilter={setStatusFilter}

        onRefresh={() => fetchData(statusFilter)}

        data={data}

      />

      <DataTable columns={columns(fetchData, statusFilter)} data={data} />

    </div>

  )

}

- src/components/admin/transaction/TransactionToolbar.tsx

'use client'

import { Button } from "@/components/ui/button"

import {

  Select,

  SelectTrigger,

  SelectValue,

  SelectContent,

  SelectItem

} from "@/components/ui/select"

import { Parser } from "json2csv"

import { Transaction } from "./TransactionTable"

type Props = {

  statusFilter: string

  setStatusFilter: (v: string) => void

  onRefresh: () => void

  data: Transaction[]

}

export function TransactionToolbar({ statusFilter, setStatusFilter, onRefresh, data }: Props) {

  const handleExport = () => {

    const parser = new Parser()

    const csv = parser.parse(data)

    const blob = new Blob([csv], { type: "text/csv;charset=utf-8;" })

    const url = URL.createObjectURL(blob)

    const link = document.createElement("a")

    link.href = url

    link.setAttribute("download", "transaksi.csv")

    link.click()

  }

  return (

    <div className="flex justify-between items-center">

      <Select onValueChange={setStatusFilter} defaultValue={statusFilter}>

        <SelectTrigger className="w-[200px]">

          <SelectValue placeholder="Filter Status" />

        </SelectTrigger>

        <SelectContent>

          <SelectItem value="all">Semua</SelectItem>

          <SelectItem value="pending">Pending</SelectItem>

          <SelectItem value="paid">Paid</SelectItem>

          <SelectItem value="cancelled">Cancelled</SelectItem>

        </SelectContent>

      </Select>

      <div className="flex gap-3">

        <Button variant="outline" onClick={onRefresh}>🔁 Refresh</Button>

        <Button variant="default" onClick={handleExport} disabled={data.length === 0}>

          📦 Export CSV

        </Button>

      </div>

    </div>

  )

}

src/components/user/destination/TicketList.tsx

'use client';

import { Ticket } from "@/types/ticket"; // Pastikan path ini sesuai

import { useState } from "react";

import { MidtransTransactionResult } from "@/types/midtrans"; // Import the type

export default function TicketList({ tickets }: { tickets: Ticket[] }) {

  const [loading, setLoading] = useState<boolean>(false);

  // Fungsi untuk menghandle klik tombol beli tiket

  const handleBuyTicket = async (ticketId: string, ticketPrice: number) => {

    setLoading(true);

    try {

      // Buat transaksi pembayaran di backend

      const order\_id = `ORDER-${Date.now()}`;  // ID transaksi berdasarkan waktu

      const response = await fetch("/api/payment/tokenize", {

        method: "POST",

        headers: {

          "Content-Type": "application/json",

        },

        body: JSON.stringify({ order\_id, amount: ticketPrice }),

      });

      const data = await response.json();

      if (data.token\_id) {

        // Memanggil Midtrans Snap.js untuk memproses pembayaran

        window.snap.pay(data.token\_id, {

          onSuccess: (result: MidtransTransactionResult) => {

            console.log("Payment success:", result);

            // Redirect atau update status setelah pembayaran berhasil

          },

          onPending: (result: MidtransTransactionResult) => {

            console.log("Payment pending:", result);

          },

          onError: (error: Error) => {

            console.error("Payment error:", error);

          },

          onClose: () => {

            console.log("User closed payment popup");

          },

        });

      } else {

        alert("Gagal mengarahkan ke halaman pembayaran");

      }

    } catch (error) {

      console.error("Failed to fetch payment token:", error);

    } finally {

      setLoading(false);

    }

  };

  return (

    <div>

      <h3 className="text-xl font-semibold">Tiket Tersedia</h3>

      <div className="space-y-4">

        {tickets.map((ticket) => (

          <div key={ticket.ticket\_id} className="bg-white p-4 rounded shadow">

            <h4 className="font-bold">{ticket.ticket\_name}</h4>

            <p className="text-sm">{ticket.ticket\_desc || "No description"}</p>

            <p className="text-lg font-semibold">

              Rp {ticket.ticket\_price.toLocaleString('id-ID')} {/\* Format angka dengan toLocaleString \*/}

            </p>

            <button

              className="bg-blue-500 text-white py-2 px-4 rounded mt-2"

              onClick={() => handleBuyTicket(ticket.ticket\_id, ticket.ticket\_price)}

              disabled={loading}

            >

              {loading ? "Processing..." : "Beli Sekarang"}

            </button>

          </div>

        ))}

      </div>

    </div>

  );

}

- src/components/user/NavbarUser.tsx

'use client'

import Link from 'next/link'

import { useEffect, useState } from 'react'

import { useRouter } from 'next/navigation'

export default function NavbarUser() {

  const [open, setOpen] = useState(false)

  const [user, setUser] = useState<{ name: string; email: string } | null>(null)

  const [showConfirm, setShowConfirm] = useState(false)

  const router = useRouter()

  useEffect(() => {

    const loadUser = () => {

      const data = localStorage.getItem('user')

      if (data) {

        setUser(JSON.parse(data))

      } else {

        setUser(null)

      }

    }

    loadUser()

    window.addEventListener('userChanged', loadUser)

    return () => window.removeEventListener('userChanged', loadUser)

  }, [])

  const handleLogout = () => {

    setShowConfirm(false) // tutup modal dulu

    // Tunggu sedikit supaya animasi/modal hilang

    setTimeout(() => {

      localStorage.removeItem('user')

      window.dispatchEvent(new Event('userChanged'))

      router.push('/user/login')

    }, 100) // 100ms sudah cukup

  }

  return (

    <>

      <nav className="bg-white shadow px-6 py-4">

        <div className="flex justify-between items-center">

          <Link href="/user" aria-label="Beranda" className="text-xl font-bold hover:bg-gray-100 px-2 py-1 rounded">

            Baturraden Tourism

          </Link>

          {/\* Menu Desktop \*/}

          <div className="hidden md:flex items-center space-x-4">

            {user ? (

              <>

                <span className="text-sm text-gray-600">Hi, {user.name}</span>

                <button onClick={() => setShowConfirm(true)} className="text-red-600">

                  Logout

                </button>

              </>

            ) : (

              <Link href="/user/login">Login</Link>

            )}

            <Link href="/user/payment">Riwayat</Link>

          </div>

          {/\* Hamburger Button \*/}

          <button onClick={() => setOpen(!open)} className="md:hidden">

            <svg className="w-6 h-6" fill="none" stroke="currentColor" viewBox="0 0 24 24">

              <path

                strokeLinecap="round"

                strokeLinejoin="round"

                strokeWidth={2}

                d={open ? 'M6 18L18 6M6 6l12 12' : 'M4 6h16M4 12h16M4 18h16'}

              />

            </svg>

          </button>

        </div>

        {/\* Mobile Menu \*/}

        {open && (

          <div className="md:hidden mt-2 flex flex-col space-y-2">

            {user ? (

              <>

                <span className="text-sm text-gray-600">Hi, {user.name}</span>

                <button onClick={() => setShowConfirm(true)} className="text-red-600 text-left">

                  Logout

                </button>

              </>

            ) : (

              <Link href="/user/login">Login</Link>

            )}

            <Link href="/user/payment">Riwayat</Link>

          </div>

        )}

      </nav>

      {/\* Modal Konfirmasi Logout \*/}

      {showConfirm && (

        <div className="fixed inset-0 z-50 bg-black bg-opacity-50 flex items-center justify-center">

          <div className="bg-white rounded-lg shadow-lg p-6 max-w-sm w-full">

            <h2 className="text-lg font-semibold mb-4">Konfirmasi Logout</h2>

            <p className="mb-6">Apakah Anda yakin ingin keluar dari akun?</p>

            <div className="flex justify-end space-x-3">

              <button

                onClick={() => setShowConfirm(false)}

                className="px-4 py-2 bg-gray-200 hover:bg-gray-300 rounded"

              >

                Batal

              </button>

              <button

                onClick={handleLogout}

                className="px-4 py-2 bg-red-600 text-white hover:bg-red-700 rounded"

              >

                Ya, Keluar

              </button>

            </div>

          </div>

        </div>

      )}

    </>

  )

}

- src/types/midtrans.d.ts

export interface MidtransTransactionResult {

  transaction\_id: string;

  order\_id: string;

  gross\_amount: string;

  payment\_type: string;

  transaction\_status: string;

  fraud\_status?: string;

  status\_code?: string;

  status\_message?: string;

  merchant\_id?: string;

  [key: string]: unknown;

}

interface MidtransSnap {

  pay: (

    token: string,

    options?: {

      onSuccess?: (result: MidtransTransactionResult) => void;

      onPending?: (result: MidtransTransactionResult) => void;

      onError?: (error: Error) => void;

      onClose?: () => void;

    }

  ) => void;

}

// Ini bagian penting:

declare global {

  interface Window {

    snap: MidtransSnap;

  }

}

- src/types/qrcode.d.ts

declare module 'qrcode' {

  export function toDataURL(data: string, options?: object): Promise<string>;

}

- package.json (copy bagian versi Next.js, qrcode, pdf-lib, midtrans)

{

  "name": "baturraden-apps",

  "version": "0.1.0",

  "private": true,

  "scripts": {

    "dev": "next dev",

    "build": "next build",

    "start": "next start",

    "lint": "next lint"

  },

  "dependencies": {

    "@prisma/client": "^6.6.0",

    "@radix-ui/react-alert-dialog": "^1.1.11",

    "@radix-ui/react-dialog": "^1.1.11",

    "@radix-ui/react-label": "^2.1.4",

    "@radix-ui/react-select": "^2.2.2",

    "@radix-ui/react-slot": "^1.2.0",

    "@radix-ui/react-toast": "^1.2.11",

    "@tanstack/react-table": "^8.21.3",

    "class-variance-authority": "^0.7.1",

    "clsx": "^2.1.1",

    "firebase": "^11.7.1",

    "formidable": "^3.5.4",

    "fs-extra": "^11.3.0",

    "html5-qrcode": "^2.3.8",

    "json2csv": "^6.0.0-alpha.2",

    "lucide-react": "^0.503.0",

    "midtrans-client": "^1.4.2",

    "next": "^14.2.3",

    "pdf-lib": "^1.17.1",

    "qrcode": "^1.5.4",

    "react": "^18.2.0",

    "react-dom": "^18.2.0",

    "recharts": "^2.15.3",

    "tailwind-merge": "^3.2.0",

    "tailwindcss-animate": "^1.0.7"

  },

  "devDependencies": {

    "@types/formidable": "^3.4.5",

    "@types/fs-extra": "^11.0.4",

    "@types/json2csv": "^5.0.7",

    "@types/node": "^20",

    "@types/qrcode": "^1.5.5",

    "@types/react": "^18.0.28",

    "@types/react-dom": "^18.0.11",

    "eslint": "^8",

    "eslint-config-next": "15.0.3",

    "postcss": "^8",

    "tailwindcss": "^3.4.1",

    "typescript": "^5"

  }

}