Sempurna! Aku akan membuatkan panduan super lengkap step-by-step, dari setup Expo React Native, integrasi Web3 (Thirdweb & alternatif), upload ke Livepeer, backend Node.js sederhana, hingga implementasi QR Code verifikasi NFT. Panduan ini akan ditulis dengan gaya bahasa manusia, sangat mudah dimengerti untuk level junior, dan terstruktur rapi. Aku mulai kerjakan sekarang, nanti akan aku kabari setelah selesai!

Panduan Lengkap: Membangun Aplikasi Mobile React Native (Expo) dengan Smart Contract di Jaringan Manta Pacific

1. Setup Environment

- **Install Node.js dan npm**: Pastikan Node.js (LTS) sudah terpasang. NPM biasanya sudah terikut. Pastikan versinya terbaru sesuai kebutuhan React Native dan Hardhat (How to Create and Deploy a Smart Contract with Hardhat | QuickNode Guides).
- **Install Hardhat**: Jalankan `npm install --save-dev hardhat` untuk memasang Hardhat. Hardhat adalah dev environment yang membantu compile, deploy, dan test smart contract (How to Create and Deploy a Smart Contract with Hardhat | QuickNode Guides) (How to Create and Deploy a Smart Contract with Hardhat | QuickNode Guides).
- **Install Expo CLI**: Gunakan `npm install -g expo-cli` atau `npx create-expo-app NamaAplikasi` untuk memulai proyek Expo. Misalnya `npx create-expo-app MyCourseApp`. Expo CLI memudahkan pengembangan React Native. Setelah buat proyek, jalankan `npx expo prebuild` sekali, lalu `npx expo run:ios` atau `npx expo run:android` untuk debug build (Expo Go *tidak* didukung saat pakai thirdweb) (thirdweb React Native SDK).
- **Livepeer Account**: Daftar di Livepeer Studio untuk mengelola video streaming. Di Dashboard Livepeer, buat API Key (Bearer token). Kunci ini digunakan di backend untuk otentikasi ketika upload video ke Livepeer (Authentication - Livepeer Docs). Pastikan **tidak** membocorkan kunci ini pada frontend publik.
- **Node.js+Express (backend) dan IDE**: Sediakan editor kode (mis. VSCode) dan terminal. Siapkan project **backend** terpisah dengan `npm init -y`, lalu pasang `express` (untuk API) dan `multer` (untuk upload file) misalnya melalui `npm install express multer`.

2. Setup Smart Contract Deployment

- **Susun Kontrak dan Hardhat Config**: Simpan kode smart contract `CourseFactory`, `CourseLicense`, `CertificateManager`, `ProgressTracker` di folder `contracts/`. Buat file konfigurasi Hardhat (`hardhat.config.js`) dengan menambahkan jaringan Manta Pacific Sepolia Testnet, misalnya:

```
module.exports = {
  solidity: "0.8.x",
  networks: {
    mantaTestnet: {
      url: "https://pacific-rpc.sepolia-testnet.manta.network/http",
      chainId: 3441006,
      accounts: [process.env.DEPLOYER_PRIVATE_KEY],
    },
```

```
  },
};
```

Di sini `chainId: 3441006` dan RPC URL diambil dari dokumentasi Manta Pacific ([Network Information | Manta Network Technical Resources](https://docs.manta.network/docs/manta-pacific/Build on Manta/Network Information#:~:text=RPC%20URLhttps%3A%2F%2Fpacific,Chain%20ID%203441006)). Currency di jaringan ini ETH (gas tetap pakai ETH).

- **Tuliskan Script Deploy**: Contoh `scripts/deploy.js`:

```
async function main() {
  const [deployer] = await ethers.getSigners();
  console.log("Deploying contracts with account:", deployer.address);

  const CourseFactory = await ethers.getContractFactory("CourseFactory");
  const courseFactory = await CourseFactory.deploy();
  await courseFactory.deployed();
  console.log("CourseFactory deployed at:", courseFactory.address);

  // Lakukan hal serupa untuk CourseLicense, CertificateManager, ProgressTracker
  const CourseLicense = await ethers.getContractFactory("CourseLicense");
  const courseLicense = await CourseLicense.deploy(/* constructor args */);
  await courseLicense.deployed();
  console.log("CourseLicense deployed at:", courseLicense.address);

  // ... dan seterusnya
}

main().catch((error) => {
  console.error(error);
  process.exit(1);
});
```

Pada contoh di atas kita memanggil `ethers.getContractFactory("NamaKontrak")` dan `.deploy()` untuk men-deploy setiap kontrak (How to Create and Deploy a Smart Contract with Hardhat | QuickNode Guides).

- **Jalankan Deploy**: Setelah membuat skrip, deploy ke testnet dengan perintah:

```
npx hardhat run scripts/deploy.js --network mantaTestnet
```

Jika berhasil, Hardhat akan mencetak alamat kontrak yang ter-deploy (How to Create and Deploy a Smart Contract with Hardhat | QuickNode Guides). Simpan alamat ini (ABI, dll.) agar frontend dapat menggunakannya. Anda bisa memverifikasi di Manta Pacific Explorer menggunakan alamat tersebut.

3. Setup Backend Server (Node.js + Express)

- **Inisialisasi Project**: Di folder backend, jalankan `npm init -y`. Pasang Express dan Multer:

```
npm install express multer
```

- **Buat Server Express**: Contoh `index.js`:

```
const express = require('express');
const multer = require('multer');
const app = express();
const upload = multer({ dest: "uploads/" });

app.use(express.json());

// Endpoint untuk daftar courses (misal mengambil data dari blockchain atau DB)
app.get('/courses', async (req, res) => {
  // Ambil data course dari smart contract atau database
  res.json(/* list of courses */);
});

// Endpoint untuk kirim progress (misal pemenandaan selesai section)
app.post('/progress', async (req, res) => {
  // Terima data progress, simpan ke DB atau blockchain
  res.json({ status: "OK" });
});

// Endpoint untuk upload video course
app.post('/upload-video', upload.single('video'), async (req, res) => {
  const file = req.file; // file video yang diunggah
  const { name } = req.body;
  // Kirim file ke Livepeer (mungkin menggunakan Livepeer JS SDK atau API)
  // Contoh menggunakan fetch:
  const videoUrl = /* URL video dari file / storage sementara */;
  const lpResponse = await fetch(
    "https://livepeer.studio/api/asset/import?url=" +
encodeURIComponent(videoUrl),
    {
      method: 'POST',
      headers: {
        Authorization: `Bearer ${process.env.LIVEPEER_API_KEY}`,
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ name, input: { url: videoUrl } }),
    }
  );
  const data = await lpResponse.json();
  res.json({ asset: data });
});

const port = 3000;
app.listen(port, () => {
  console.log(`Server running on http://localhost:${port}`);
});
```

Kode di atas menggunakan **Multer** untuk menerima upload file (video) (Building a File Upload REST API with Node.js and Express - DEV Community). Endpoint `app.listen` akan memulai server

(contoh log server) (Building a File Upload REST API with Node.js and Express - DEV Community).

- **Integrasi Livepeer**: Di route upload di atas, setelah file diterima, kirimkan video ke Livepeer dengan API key (`Bearer API_KEY` dalam header) (Authentication - Livepeer Docs). Bisa menggunakan `fetch` atau library `livepeer` JS. Hasilnya berupa asset ID/playback ID yang dapat disimpan (misal di DB) untuk digunakan di frontend. Penting: simpan `LIVEPEER_API_KEY` di environment backend agar tidak bocor ke publik.

4. Setup Frontend React Native

Wallet Login

- **Pasang Thirdweb SDK**: Instal dengan `npm install @thirdweb/react @thirdweb-dev/react-native-adapter` .
- **Konfigurasi ThirdwebProvider**: Di root (misal `App.js`), wrap aplikasi dengan ThirdwebProvider, misalnya:

```
import { ThirdwebProvider, ChainId } from "@thirdweb/react";
// ...
export default function App() {
  return (
    <ThirdwebProvider desiredChainId={3441006}>
      {/* Komponen selanjutnya */}
    </ThirdwebProvider>
  );
}
```

Di sini `desiredChainId={3441006}` menunjuk ke Manta Pacific Sepolia Testnet ([Network Information | Manta Network Technical Resources](https://docs.manta.network/docs/manta-pacific/Build on Manta/Network Information#:~:text=RPC%20URLhttps%3A%2F%2Fpacific,Chain%20ID%203441006)).

- **Connect Wallet Button**: Gunakan komponen `ConnectButton` dari thirdweb untuk memudahkan user login via wallet pilihan. Contohnya:

```
import { ConnectButton } from "@thirdweb/react-native";
// ...
function LoginScreen() {
  return (
    <View>
      <Text>Mari hubungkan wallet Anda</Text>
      <ConnectButton client={client} chain={defineChain(3441006)} />
    </View>
  );
}
```

Tombol ini otomatis membuka modal pilihan wallet (termasuk in-app wallet, WalletConnect, dll) (Build Web3 Mobile Apps with React Native and thirdweb SDK). In-App Wallet thirdweb memungkinkan pengguna pemula membuat wallet dengan email atau login sosial tanpa tinggalkan aplikasi (Build Web3 Mobile Apps with React Native and thirdweb SDK). Sebagai fallback, jika thirdweb tidak tersedia, Anda dapat mengintegrasikan **WalletConnect** secara manual.

Fetch & Tampilkan Daftar Courses

- **Hubungkan ke CourseFactory**: Gunakan `getContract` dan hook `useReadContract` untuk membaca data on-chain. Contoh:

```
import { getContract, useReadContract } from "@thirdweb/react";

const factoryContract = getContract({
  client,
  chain: defineChain(3441006),
  address: COURSE_FACTORY_ADDRESS,
  abi: CourseFactoryABI,
});

const { data: courses } = useReadContract({
  contract: factoryContract,
  method: "getAllCourses",
  params: [],
});

// Tampilkan courses dalam list
<FlatList
  data={courses}
  renderItem={({ item }) => (
    <Text>{item.title} - Price: {item.price} ETH</Text>
  )}
/>
```

Kode di atas mengambil instansi kontrak `CourseFactory`, lalu memanggil method (misalnya `getAllCourses()`) untuk daftar course. Anda bisa menyesuaikan nama method sesuai smart contract yang ada (Build Web3 Mobile Apps with React Native and thirdweb SDK) (Build Web3 Mobile Apps with React Native and thirdweb SDK).

Mint License NFT (Pembelian Akses)

- **Hubungkan ke CourseLicense**: Untuk memproses pembelian, sambungkan kontrak `CourseLicense`. Misalnya:

```
import { useSendAndConfirmTransaction, prepareContractCall } from
"@thirdweb/react";

const licenseContract = getContract({
  client,
  chain: defineChain(3441006),
  address: COURSE_LICENSE_ADDRESS,
  abi: CourseLicenseABI,
});

const { sendTransaction: sendLicense } = useSendAndConfirmTransaction();
const buyLicense = async (courseId) => {
  const tx = prepareContractCall({
    contract: licenseContract,
    method: "mintLicense",
```

```
    params: [courseId],
  });
  sendLicense(tx);
};

// Contoh tombol beli:
<Button title="Beli Lisensi" onPress={() => buyLicense(selectedCourseId)} />
```

Cara ini memanggil fungsi `mintLicense(courseId)` pada kontrak smart contract untuk mencetak NFT lisensi (Build Web3 Mobile Apps with React Native and thirdweb SDK). Pastikan user memiliki cukup ETH (atau gas) di wallet (atau gunakan account abstraction untuk gasless jika disupport thirdweb).

View Sections

- **Ambil Bagian (Sections)**: Setelah user beli lisensi, bisa ambil data section dari kontrak. Misalnya fungsi `getSections(courseId)`. Contoh:

```
const { data: sections } = useReadContract({
  contract: factoryContract,
  method: "getSections",
  params: [selectedCourseId],
});

// Render sections:
sections.map((sec, idx) => (
  <Text key={idx}>{sec.title}</Text>
));
```

Sesuaikan nama fungsi `getSections` dengan yang ada di smart contract Anda (Build Web3 Mobile Apps with React Native and thirdweb SDK).

Track Progress & Complete Section

- **Hubungkan ke ProgressTracker**: Untuk menandai kemajuan, panggil kontrak `ProgressTracker`:

```
const progressContract = getContract({
  client,
  chain: defineChain(3441006),
  address: PROGRESS_TRACKER_ADDRESS,
  abi: ProgressTrackerABI,
});

const { sendTransaction: sendProgress } = useSendAndConfirmTransaction();
const completeSection = async (courseId, sectionId) => {
  const tx = prepareContractCall({
    contract: progressContract,
    method: "completeSection",
    params: [courseId, sectionId],
  });
  sendProgress(tx);
};
```

```
// Contoh tombol selesaikan:
<Button title="Selesaikan Section" onPress={() =>
completeSection(selectedCourseId, secId)} />
```

Metode ini mencatat bahwa user telah menyelesaikan suatu section (Build Web3 Mobile Apps with React Native and thirdweb SDK). Anda dapat menunjukkan progress ini di UI (misalnya persentase selesai, badge, dsb).

Mint Certificate NFT Setelah Course Selesai

- **Hubungkan ke CertificateManager**: Setelah semua section selesai, user bisa mendapat sertifikat. Lakukan mint melalui kontrak `CertificateManager`:

```
const certContract = getContract({
  client,
  chain: defineChain(3441006),
  address: CERTIFICATE_MANAGER_ADDRESS,
  abi: CertificateManagerABI,
});

const { sendTransaction: sendCert } = useSendAndConfirmTransaction();
const mintCertificate = async (courseId) => {
  const tx = prepareContractCall({
    contract: certContract,
    method: "mintCertificate",
    params: [courseId],
  });
  sendCert(tx);
};

<Button title="Mint Sertifikat" onPress={() => mintCertificate(selectedCourseId)}
/>
```

Fungsi `mintCertificate(courseId)` akan menghasilkan NFT sertifikat untuk user (Build Web3 Mobile Apps with React Native and thirdweb SDK). Anda bisa menyimpan `tokenId` atau hash sertifikat ini untuk validasi selanjutnya.

Generate QR Code untuk Sertifikat

(Create a QR Code Generator App using React-Native | GeeksforGeeks)Gunakan paket **react-native-qrcode-svg** untuk menghasilkan QR Code di aplikasi. Pasang dengan `npm install react-native-qrcode-svg react-native-svg`. Contohnya:

```
import QRCode from 'react-native-qrcode-svg';

function CertificateScreen({ certificateData }) {
  return (
    <View style={{ alignItems: 'center', marginTop: 20 }}>
      <QRCode value={certificateData} size={200} />
      <Text>Scan untuk verifikasi sertifikat</Text>
    </View>
```

```
    );
}
```

Pada kode di atas, `certificateData` bisa berupa hash atau URL metadata NFT. QR Code akan menampilkan data tersebut dan bisa dipindai untuk verifikasi. Contoh UI di atas menunjukkan kotak input dan tombol "Generate QR Code" (Create a QR Code Generator App using React-Native | GeeksforGeeks) (Create a QR Code Generator App using React-Native | GeeksforGeeks).

5. Deployment ke Manta Pacific Testnet & Testing

- **Deploy Kontrak ke Manta**: Jika ingin gunakan mainnet, atur `networks.mantaMain` di Hardhat dengan RPC `https://pacific-rpc.manta.network/http` dan `chainId: 169` (createx/hardhat.config.ts at main · pcaversaccio/createx · GitHub) ([Network Information | Manta Network Technical Resources](https://docs.manta.network/docs/manta-pacific/Build on Manta/Network Information#:~:text=RPC%20URLhttps%3A%2F%2Fpacific,Chain%20ID%203441006)). Untuk testnet, gunakan `mantaTestnet` seperti di atas. Jalankan perintah deploy Hardhat sesuai network target.
- **Verifikasi Kontrak**: Salin alamat kontrak yang ter-deploy dan cek di Manta Pacific Explorer atau socialscan untuk melihat transaksi deploy dan ABI.
- **Uji Aplikasi**: Pastikan proses “mint” NFT, pembacaan data, dan interaksi blockchain berjalan. Gunakan test tokens (ETH di testnet) dan pantau transaksi melalui explorer. Tindakan seperti beli kursus, menyelesaikan section, dan mint sertifikat harus memicu transaksi di chain Manta Pacific. Jika ada bug, cek log di Hardhat atau di konsol browser.

6. Bonus: UX Tips untuk Pemula

- **Gunakan In-App Wallet**: Integrasi in-app wallet Thirdweb sangat membantu pemula karena mereka bisa membuat wallet via email/social tanpa pusing install eksternal (Build Web3 Mobile Apps with React Native and thirdweb SDK). Misalnya, Anda bisa menyediakan opsi *Login dengan Email* atau *Metamask/WalletConnect* sekaligus.
- **Sederhanakan Antarmuka**: Gunakan bahasa sehari-hari (bahasa Indonesia non-jargon) untuk instruksi. Contohnya, ganti “kunci privat” dengan “kata kunci rahasia” jika perlu, atau tampilkan instruksi langkah demi langkah. Pastikan tombol dan label jelas (misal “Beli Kursus”, “Mulai Belajar”, “Selesaikan Section”).
- **Berikan Feedback Visual**: Saat aplikasi melakukan transaksi blockchain, tampilkan spinner atau status “Memproses transaksi…”. Setelah transaksi selesai, konfirmasi dengan pesan berhasil (mis. “Lisensi berhasil dicetak”) atau gagal (“Transaksi dibatalkan”). Hal ini membantu pengguna paham apa yang terjadi.
- **Sediakan Bantuan/Panduan Singkat**: Sebagai pemula Web3, sertakan sedikit tutorial atau tooltips. Misalnya, sebelum user connect wallet, tampilkan info singkat: “Kami menggunakan wallet Anda untuk menyimpan lisensi NFT sebagai bukti kepemilikan kursus” atau “Sertifikat akan muncul sebagai QR Code setelah kursus selesai”.
- **Fallback Konektivitas**: Selain thirdweb, Anda bisa sediakan opsi manual via WalletConnect SDK agar pengguna bisa connect lewat MetaMask mobile atau TrustWallet jika mereka terbiasa dengan itu.
- **Keamanan & Jelasnya Data**: Pastikan pengguna tahu bahwa NFT lisensi dan sertifikat adalah milik mereka sendiri. Tampilkan alamat penerima dan link ke explorer setelah mint. Ini menumbuhkan

kepercayaan karena mereka bisa verifikasi langsung di blockchain.

- **Performance dan Optimasi**: Karena video kursus diupload ke Livepeer, tampilkan thumbnail atau loading kecil untuk video. Jangan memaksa download video melalui blockchain. Simpan metadata kursus (judul, deskripsi, durasi) di backend atau storage terpisah agar frontend cepat memuat list kursus.

Dengan mengikuti langkah-langkah di atas dan memanfaatkan SDK Thirdweb serta Livepeer, Anda dapat membangun aplikasi e-learning mobile dengan blockchain (NFT lisensi dan sertifikat) yang ramah pemula di jaringan Manta Pacific. Selamat mencoba dan terus eksplorasi Web3!

Sumber Referensi: Panduan dan dokumentasi resmi Thirdweb (React Native SDK) (thirdweb React Native SDK) (Build Web3 Mobile Apps with React Native and thirdweb SDK), dokumentasi Manta Pacific (EVM Compatible | Manta Network Technical Resources) ([Network Information | Manta Network Technical Resources](https://docs.manta.network/docs/manta-pacific/Build on Manta/Network Information#:~:text=RPC%20URLhttps%3A%2F%2Fpacific,Chain%20ID%203441006)), dan tutorial Hardhat/Node.js (How to Create and Deploy a Smart Contract with Hardhat | QuickNode Guides) (Building a File Upload REST API with Node.js and Express - DEV Community). Gambar aplikasi QR Code diambil dari contoh *GeeksforGeeks* (Create a QR Code Generator App using React-Native | GeeksforGeeks) (Create a QR Code Generator App using React-Native | GeeksforGeeks).

Terima kasih jawabannya! Aku akan buatkan tutorial React Native Expo full lengkap, mulai dari login Wallet (Thirdweb & WalletConnect fallback), fetching course dari blockchain, minting license NFT, tracking progress, issuing certificate NFT, generate QR code verifikasi sertifikat, hingga membuat halaman scan QR Code untuk validasi. Aku juga akan kasih penjelasan kapan perlu backend (cache) dan kapan tidak, biar pemula seperti kamu bisa memahami arsitektur Web3 lebih dalam. Aku mulai kerjakan sekarang dan akan kabari setelah siap ya!

Persiapan Environment

Sebelum mulai, pastikan **Node.js** terinstal (versi LTS terbaru) dan **npm/yarn** sudah tersedia. Install juga **Expo CLI** secara global:

```
npm install -g expo-cli
```

Selanjutnya, buat akun di Thirdweb Dashboard dan **dapatkan Client ID** untuk React Native. Banyak komponen Thirdweb memerlukan `clientId` agar bisa tersambung ke RPC free mereka (thirdweb React Native SDK). Anda dapat membuat `clientId` baru di Dashboard Thirdweb sebelum melanjutkan.

Catatan Penting: Gunakan selalu versi terbaru library Thirdweb yang disarankan untuk React Native. Pastikan juga SDK Chain (misal Ethereum) terinstall jika Anda menargetkan jaringan spesifik.

1. Setup Expo React Native

Buat projek baru dengan Expo:

```
expo init nama-proyek --template blank
cd nama-proyek
```

Setelah itu, install package yang dibutuhkan Thirdweb dan peer-dependencies yang disebutkan oleh Thirdweb:

```
expo install thirdweb @thirdweb-dev/react-native-adapter
expo install react-native-get-random-values @react-native-community/netinfo expo-application @react-native-async-storage/async-storage expo-web-browser expo-linking react-native-aes-gcm-crypto react-native-quick-crypto amazon-cognito-identity-js @coinbase/wallet-mobile-sdk react-native-mmkv react-native-svg @walletconnect/react-native-compat react-native-passkey
```

Perintah di atas mengikuti anjuran dokumentasi Thirdweb RN (thirdweb React Native SDK). Jika Anda menggunakan React Native versi ≥0.60, jalankan juga `pod install` di folder ios.

Karena Thirdweb membutuhkan dukungan named exports modern, kita juga perlu **menyesuaikan** `metro.config.js`. Jalankan:

```
npx expo customize metro.config.js
```

lalu ubah file `metro.config.js` Anda seperti ini:

```
const { getDefaultConfig } = require("expo/metro-config");
const config = getDefaultConfig(__dirname);
config.resolver.unstable_enablePackageExports = true;
config.resolver.unstable_conditionNames = [
  "react-native",
  "browser",
  "require",
];
module.exports = config;
```

Potongan di atas memastikan Metro (bundler React Native) dapat mengenali export named pada package Thirdweb (thirdweb React Native SDK).

Terakhir, di `App.js` atau `App.tsx`, **import polyfill Thirdweb** sebelum import lain:

```
// file: App.tsx (atau App.js)
import "@thirdweb-dev/react-native-adapter"; // Polyfill wajib untuk Thirdweb RN ([thirdweb React Native SDK](https://portal.thirdweb.com/react-native/v5/getting-started#:~:text=%2F%2F%20file%3A%20App))
import React from "react";
import { Text, View } from "react-native";
// ... import lainnya
export default function App() {
  return (
    <View style={{ flex: 1 }}>
      <Text>App Thirdweb Starter</Text>
    </View>
  );
}
```

Import `@thirdweb-dev/react-native-adapter` sebaiknya paling atas, sesuai instruksi Thirdweb (thirdweb React Native SDK).

2. Integrasi Thirdweb SDK & WalletConnect

Sekarang kita integrasikan Thirdweb ke dalam aplikasi. Bungkus komponen utama dengan `ThirdwebProvider` dan atur koneksi wallet. Contohnya:

```
import "@thirdweb-dev/react-native-adapter";
import React from "react";
import { ThirdwebProvider, walletConnect, ConnectWallet } from "@thirdweb-dev/react-native";
import { Ethereum } from "@thirdweb-dev/chains";

export default function App() {
  return (
    <ThirdwebProvider
      clientId="YOUR_CLIENT_ID"          // Ganti dengan Client ID Anda dari Dashboard ([thirdweb docs](https://portal.thirdweb.com/changelog/walletconnect-in-our-react-native-sdk#:~:text=))
      activeChain={Ethereum}             // Rantai yang aktif (misal Ethereum)
      supportedWallets={[walletConnect()]} // Aktifkan WalletConnect sebagai opsi wallet ([thirdweb docs](https://portal.thirdweb.com/changelog/walletconnect-in-our-react-native-sdk#:~:text=))
    >
      <ConnectWallet />                 {/* Tombol Connect Wallet bawaan Thirdweb RN ([How to Fully Customize the ConnectWallet Button in the React Native SDK](https://blog.thirdweb.com/guides/reactnative-sdk-full-theme-customization/#:~:text=Import%20the%20SDK%27s%20%27ConnectWallet%27%20component)) */}
    </ThirdwebProvider>
  );
}
```

Pada kode di atas, kita **menggunakan** `ConnectWallet` dari Thirdweb RN sebagai komponen UI untuk menghubungkan dompet pengguna (How to Fully Customize the ConnectWallet Button in the React Native SDK). Opsi `supportedWallets={[walletConnect()]}` memastikan pengguna tanpa wallet in-app (misal MetaMask di perangkat) tetap bisa connect lewat WalletConnect (thirdweb docs). Ketika tombol tersebut diklik, akan muncul modal pilihan wallet (MetaMask, Coinbase Wallet, dll) untuk login.

Tips: Anda dapat menyesuaikan style dan judul button via props `theme` atau `buttonTitle` sesuai dokumentasi Thirdweb RN (How to Fully Customize the ConnectWallet Button in the React Native SDK), agar UI sesuai kebutuhan. Pastikan juga user menyetujui koneksi wallet (pop-up) sebelum lanjut ke langkah berikutnya.

3. Membaca Data Course dari Blockchain

Untuk menampilkan daftar course dari kontrak **CourseFactory**, kita memanggil fungsi `getCourses()` (atau fungsi serupa) melalui Thirdweb. Contoh:

```
import { getContract, useReadContract } from "@thirdweb-dev/react-native";
import { Ethereum } from "@thirdweb-dev/chains";
```

```
// Inisialisasi kontrak CourseFactory
const courseFactoryAddress = "0x...CourseFactory...";
const courseFactoryContract = getContract({
  address: courseFactoryAddress,
  chain: Ethereum,
});

// Mengambil data courses (misal array Course) dari blockchain
const { data: courses, isLoading, error } = useReadContract({
  contract: courseFactoryContract,
  method: "getCourses",
});
```

Di sini kita gunakan hook `useReadContract` yang diperkenalkan di Thirdweb SDK v5. Hook ini menggantikan hook-hok lama dan berguna untuk **membaca state kontrak** di React/VN (thirdweb React SDK). Hasil `courses` biasanya berupa array objek atau alamat course. Anda dapat menampilkan `courses` di UI setelah data tersedia (`!isLoading`).

Catatan: Pastikan nama fungsi kontrak (`getCourses`) sesuai ABI kontrak. Jika fungsi berbeda (misal `allCourses()`), sesuaikan `method` . Anda juga bisa memberikan `args` array jika fungsi membutuhkan parameter. Hook `useReadContract` akan otomatis refresh data jika kontrak diupdate (thirdweb React SDK).

4. Minting License dan Pengecekan Status

Setelah user melakukan proses pembelian (misal checkout), kita harus **mint NFT License** dengan memanggil kontrak `CourseLicense` . Misalnya kontrak punya fungsi `mintLicense(address _to, uint256 _courseId)` . Contoh kode:

```
import { useSendAndConfirmTransaction } from "@thirdweb-dev/react-native";

// Inisialisasi kontrak CourseLicense
const licenseAddress = "0x...CourseLicense...";
const licenseContract = getContract({ address: licenseAddress, chain: Ethereum });

const { mutateAsync: mintLicense, isLoading: minting } =
useSendAndConfirmTransaction();

// Fungsi untuk mint license
const handleMintLicense = async (userAddress, courseId) => {
  try {
    const tx = await mintLicense({
      contract: licenseContract,
      functionName: "mintLicense",
      args: [userAddress, courseId],
    });
    console.log("License minted!", tx);
  } catch (error) {
    console.error("Mint gagal:", error);
  }
};
```

Kita menggunakan hook `useSendAndConfirmTransaction` (atau `useSendTransaction`) untuk mengeksekusi transaksi tulis ke kontrak (thirdweb React SDK). Setelah user sign transaksi di wallet, NFT license akan dibuat dan dikirim ke address user.

Untuk **memeriksa status license** (apakah user sudah memiliki license), Anda dapat menggunakan `useReadContract` lagi, misalnya panggil fungsi `balanceOf(userAddress)` pada kontrak ERC721 atau fungsi khusus `hasLicense(user, courseId)`. Contoh sederhana dengan `balanceOf`:

```
const { data: balance } = useReadContract({
  contract: licenseContract,
  method: "balanceOf",
  args: [userAddress],
});
// Jika balance > 0 berarti user memiliki minimal 1 license NFT
```

Dengan demikian, Anda bisa menampilkan pesan “License diperoleh” jika `balance > 0`.

5. Menyelesaikan Section dan Tracking Progress

Saat user menyelesaikan setiap section di course, panggil fungsi **ProgressTracker** di kontrak untuk update progress. Misalnya kontrak punya fungsi `completeSection(address _user, uint256 _courseId, uint256 _sectionIndex)`. Contoh:

```
const progressAddress = "0x...ProgressTracker...";
const progressContract = getContract({ address: progressAddress, chain: Ethereum });

const { mutateAsync: completeSection } = useSendAndConfirmTransaction();

// Misal user klik tombol selesai section
const handleCompleteSection = async (userAddress, courseId, sectionIndex) => {
  try {
    await completeSection({
      contract: progressContract,
      functionName: "completeSection",
      args: [userAddress, courseId, sectionIndex],
    });
    console.log("Section selesai dicatat");
  } catch (error) {
    console.error("Gagal update progress:", error);
  }
};
```

Setelah itu, Anda bisa **membaca progress** user untuk ditampilkan (misal berapa persen selesai) dengan fungsi seperti `getProgress(user, courseId)`. Contoh:

```
const { data: progress } = useReadContract({
  contract: progressContract,
  method: "getProgress",
  args: [userAddress, courseId],
});
// Tampilkan nilai progress (misal persen atau section terakhir)
```

Dengan cara ini, UI bisa memperlihatkan bar/indikator progress belajar user berdasarkan data on-chain.

6. Mengeluarkan Sertifikat dan Menampilkan Metadata

Setelah user **menyelesaikan seluruh course**, kita mint sertifikat NFT melalui kontrak **CertificateManager**. Misalnya fungsi kontrak `mintCertificate(address to, uint256 courseId)`. Contoh:

```
const certAddress = "0x...CertificateManager...";
const certContract = getContract({ address: certAddress, chain: Ethereum });

const { mutateAsync: mintCert } = useSendAndConfirmTransaction();

const handleMintCertificate = async (userAddress, courseId) => {
  try {
    await mintCert({
      contract: certContract,
      functionName: "mintCertificate",
      args: [userAddress, courseId],
    });
    console.log("Certificate NFT minted");
  } catch (error) {
    console.error("Gagal mint sertifikat:", error);
  }
};
```

Setelah sertifikat dicetak di blockchain, kita dapat menampilkan metadata-nya di aplikasi. Salah satu cara mudah adalah menggunakan hook `useNFT` untuk mengambil data NFT dan komponen `ThirdwebNftMedia` (atau elemen `<Image />` biasa) untuk render. Contohnya (dengan asumsi NFT pertama tokenId=0):

```
import { useContract, useNFT } from "@thirdweb-dev/react";

// Dapatkan objek kontrak menggunakan hook
const { contract: certNftContract } = useContract(certAddress);
const { data: certificate } = useNFT(certNftContract, 0);

// Di render:
return (
  <View>
    {certificate && (
      <ThirdwebNftMedia metadata={certificate.metadata} style={{ width: 200, height:
200 }} />
    )}
  </View>
);
```

Cara di atas mengikuti panduan Thirdweb: gunakan `useNFT` untuk memuat metadata NFT, lalu tampilkan (misal dengan `ThirdwebNftMedia` atau tag gambar) (How to Render NFT Metadata In a React App). Metadata bisa berupa nama, tanggal selesai, dan link gambar, tergantung implementasi kontrak.

7. Generate QR Code dari Data Sertifikat

Setiap sertifikat yang diterbitkan bisa kita ubah menjadi QR Code untuk kemudahan share/scan. Kita gunakan library **react-native-qrcode-svg**. Install dengan:

```
yarn add react-native-svg react-native-qrcode-svg
```

Kemudian dalam kode React Native, generate QR Code dengan komponen `QRCode`:

```
import QRCode from "react-native-qrcode-svg";

// Misal kita ingin QR berisi string JSON atau URL sertifikat
const certificateData = JSON.stringify({ certId: 123, owner: userAddress });

return (
  <View>
    <QRCode value={certificateData} size={200} />
  </View>
);
```

Contoh di atas diambil dari dokumentasi library: cukup beri prop `value` berupa string (misal URL atau JSON) untuk konten QR (react-native-qrcode-svg - npm). Ketika di-render, komponen ini menampilkan gambar QR yang dapat discan. Anda juga dapat menambahkan logo kecil di tengah QR atau meng-custom warna melalui props yang tersedia (react-native-qrcode-svg - npm).

8. Halaman Scan QR Code dan Validasi Sertifikat

Buat halaman (screen) khusus yang menggunakan kamera untuk memindai QR Code sertifikat. Gunakan modul **Expo Camera / BarCodeScanner**. Contoh dengan `expo-barcode-scanner`:

```
import React, { useState, useEffect } from "react";
import { View, Text, Button } from "react-native";
import { BarCodeScanner } from "expo-barcode-scanner";

export default function ScanScreen() {
  const [hasPermission, setHasPermission] = useState(null);
  const [scanned, setScanned] = useState(false);
  const [result, setResult] = useState(null);

  useEffect(() => {
    (async () => {
      const { status } = await BarCodeScanner.requestPermissionsAsync();
      setHasPermission(status === "granted");
    })();
  }, []);

  const handleBarCodeScanned = ({ data }) => {
    setScanned(true);
    setResult(data); // data is string value from QR
  };

  if (hasPermission === null) {
```

```
    return <Text>Meminta izin kamera ...</Text>;
  }
  if (hasPermission === false) {
    return <Text>Izin kamera ditolak.</Text>;
  }

  return (
    <View style={{ flex: 1 }}>
      {!scanned && (
        <BarCodeScanner
          onBarCodeScanned={handleBarCodeScanned}
          style={{ flex: 1 }}
        />
      )}
      {scanned && (
        <>
          <Text>Data QR: {result}</Text>
          <Button title="Scan Lagi" onPress={() => { setScanned(false);
setResult(null); }} />
        </>
      )}
    </View>
  );
}
```

Contoh di atas menunjukkan cara menggunakan `BarCodeScanner` untuk memindai QR (BarCodeScanner - Expo Documentation). Setelah QR terbaca, `data` ditangkap di `handleBarCodeScanned`. Misalnya QR berisi `{ certId: 123 }`.

Langkah selanjutnya adalah **validasi sertifikat** melalui blockchain. Anda dapat parsing `result` untuk mengambil `certId`, lalu panggil fungsi kontrak (misal `isCertificateValid(tokenId)`) menggunakan `useReadContract` atau `useContractRead`. Contohnya:

```
import { useReadContract } from "@thirdweb-dev/react-native";

const certContract = getContract({ address: certAddress, chain: Ethereum });
const tokenId = parseResultToId(result); // fungsi parse JSON dari result
const { data: isValid } = useReadContract({
  contract: certContract,
  method: "isCertificateValid",
  args: [tokenId],
});
```

Jika `isValid` true, tampilkan “Sertifikat valid” ke user, sebaliknya “Tidak valid” jika false. (Nama fungsi `isCertificateValid` hanyalah contoh; sesuaikan dengan ABI kontrak Anda.)

Catatan: Pastikan untuk menangani hasil parse QR dengan baik. Juga, pastikan kontrak CertificateManager memiliki metode pengecekan (verifikasi) sertifikat. Jika tidak, Anda dapat menggunakan efek samping atau backend sederhana untuk memverifikasi dengan membaca data on-chain.

9. Tips Penggunaan Backend dan Optimasi

Secara teknis, semua data (courses, progress, sertifikat) bisa langsung di-fetch dari blockchain. Namun panggilan RPC sering kali **berbiaya gas (untuk tulis)** dan **berpotensi lambat** untuk pembacaan berulang. Berikut beberapa opsi backend/caching:

- **Tanpa backend:** Aplikasi langsung memanggil kontrak setiap saat. Ini sederhana dan sepenuhnya *trustless*, namun bisa terasa lambat saat banyak data dibaca berulang. Kelebihannya: data pasti update terbaru dari chain.
- **Dengan backend/caching:** Anda bisa menyimpan sementara hasil query (misal daftar course, progress user, metadata sertifikat) di server Anda sendiri atau menggunakan solusi seperti TheGraph/Indexing service. Dengan cache, aplikasi tidak perlu query ke node setiap kali, sehingga data bisa ditampilkan jauh lebih cepat (Unlocking the Power of Caching: Accelerate Your Web3 Products). Misalnya, cache daftar course di database dan update sekali-sekali, sehingga list course disajikan seketika tanpa delay RPC.

Secara umum, **backend disarankan** bila aplikasi Anda melayani banyak pengguna atau data yang besar. Caching mengurangi beban `RPC calls` yang berulang (Unlocking the Power of Caching: Accelerate Your Web3 Products). Namun, backend menambah kompleksitas: Anda perlu sinkronisasi data on-chain dengan cache. Pastikan untuk mengatur *cache invalidation* (misal ketika ada course baru diciptakan).

Sebagai best practice: gunakan caching untuk data yang jarang berubah (contoh: daftar course, atau history sertifikat) agar performa aplikasi baik. Untuk data yang sangat sensitif/update real-time (misal saldo atau nonce), panggil langsung smart contract. Dengan arsitektur hybrid ini, aplikasi React Native Anda tetap responsif dan memanfaatkan keamanan blockchain secara optimal.

Sumber Referensi: Dokumentasi Thirdweb v5, blog Thirdweb, dan dokumentasi Expo telah digunakan sebagai acuan dalam tutorial ini (thirdweb React Native SDK) (thirdweb docs) (thirdweb React SDK) (react-native-qrcode-svg - npm) (BarCodeScanner - Expo Documentation) (Unlocking the Power of Caching: Accelerate Your Web3 Products). Ini memastikan informasi terbaru dan praktik yang disarankan dipakai dalam pembuatan aplikasi.

Mantap! Aku akan mulai membuat tutorial super lengkap berdasarkan permintaan kamu.

Tutorial ini akan dibagi menjadi beberapa bagian besar:

1. Setup Lingkungan Developer (Tools Wajib)
2. Pembuatan dan Deployment Smart Contract (Course License, Certificate Manager, Progress Tracker)
3. Upload Video ke Livepeer dan Simpan Metadata di IPFS
4. Setup Project Mobile App dengan React Native + Expo + Thirdweb SDK
5. Integrasi Frontend Mobile ke Smart Contract (Mint NFT, Cek License, Dapatkan Data Course, Request Certificate)
6. Desain UI/UX Basic untuk Aplikasi (Bonus: Layout + Wireframe)
7. Testing End-to-End (Dari deploy sampai user mint NFT dan akses course)

Aku akan segera kembali dengan panduan super duper lengkap ini untukmu, pastikan siap membaca dan mengimplementasikan semuanya 🚀 🔥

Panduan Lengkap Membangun Aplikasi Web3 Mobile (Android) dengan React Native, Expo, dan Thirdweb SDK

Tutorial ini akan memandu Anda **langkah demi langkah** membangun aplikasi mobile Web3 di Android menggunakan React Native, Expo, dan Thirdweb. Kita akan mencakup persiapan lingkungan, deployment smart contract, penyimpanan video dan metadata, sampai ke integrasi fitur-fitur utama di aplikasi. Bahasa yang digunakan mudah dimengerti dan cocok untuk developer Web3 pemula.

1. Persiapan Lingkungan Development

Sebelum mulai coding, siapkan dulu lingkungan kerja Anda:

- **Node.js**: Install versi LTS terbaru dari nodejs.org. Versi 16 atau 18 sudah cukup.
- **Hardhat**: Framework pengembangan smart contract. Pasang dengan:

  ```
  npm install --save-dev hardhat
  ```

- **Expo CLI**: Untuk proyek React Native dengan Expo. Pasang secara global:

  ```
  npm install -g expo-cli
  ```

- **Visual Studio Code (VSCode)**: IDE yang direkomendasikan, dengan ekstensi linting JavaScript/TypeScript dan Solidity.
- **Metamask**: Wallet kripto untuk testing, install ekstensi di browser Anda. Siapkan akun dan ambil private key (untuk deploy).
- **Thirdweb SDK dan CLI**: Thirdweb mempermudah integrasi Web3. Instal CLI (opsional) dan library:

  ```
  npm install -g @thirdweb-dev/cli
  npm install @thirdweb-dev/react-native
  ```

  Anda juga bisa langsung membuat proyek React Native dengan Thirdweb:

  ```
  npx thirdweb create app --react-native
  ```

  Cara ini membuatkan starter kit Expo + Thirdweb dengan cepat (thirdweb React Native SDK). Jika tidak menggunakan CLI, Anda dapat membuat projek Expo biasa lalu menambahkan Thirdweb SDK di dalamnya.

Setelah semua diinstall, siapkan juga file **.env** di folder proyek Hardhat, misalnya:

```
MANTA_RPC_URL=https://rpc.mantapacific-testnet.com
DEPLOYER_PRIVATE_KEY=<private_key_metamask>
```

(Ganti `MANTA_RPC_URL` dan `chainId` sesuai dokumentasi Manta Pacific Testnet yang tersedia.)

2. Deployment Smart Contracts ke Manta Pacific Testnet

Asumsikan kode smart contract sudah tersedia: `CourseFactory.sol`, `CourseLicense.sol`, `ProgressTracker.sol`, `CertificateManager.sol`. Berikut langkah deploy-nya:

1. **Konfigurasi Hardhat**. Buat `hardhat.config.js` di root proyek:

```
require("@nomiclabs/hardhat-ethers");
require("dotenv").config();

module.exports = {
  solidity: "0.8.4",
  networks: {
    manta: {
      url: process.env.MANTA_RPC_URL,
      accounts: [process.env.DEPLOYER_PRIVATE_KEY],
      chainId:  /* masukkan chainId Manta Pacific Testnet, misal 71393 */,
    },
    // (Opsional) tambahkan testnet lain jika perlu
  },
};
```

Masukkan `MANTA_RPC_URL`, `DEPLOYER_PRIVATE_KEY`, dan `chainId` yang sesuai testnet Manta.

2. **Buat skrip deployment**. Di folder `scripts/`, misalnya `deploy.js`:

```
const hre = require("hardhat");

async function main() {
  // Deploy CourseFactory
  const Factory = await hre.ethers.getContractFactory("CourseFactory");
  const factory = await Factory.deploy();
  await factory.deployed();
  console.log("CourseFactory deployed at:", factory.address);

  // Deploy CourseLicense
  const License = await hre.ethers.getContractFactory("CourseLicense");
  const license = await License.deploy();
  await license.deployed();
  console.log("CourseLicense deployed at:", license.address);

  // Deploy ProgressTracker
  const Tracker = await hre.ethers.getContractFactory("ProgressTracker");
  const tracker = await Tracker.deploy();
  await tracker.deployed();
  console.log("ProgressTracker deployed at:", tracker.address);

  // Deploy CertificateManager
  const Cert = await hre.ethers.getContractFactory("CertificateManager");
  const cert = await Cert.deploy();
  await cert.deployed();
  console.log("CertificateManager deployed at:", cert.address);
}

main()
  .then(() => process.exit(0))
  .catch((error) => {
    console.error(error);
    process.exit(1);
  });
```

Jalankan perintah:

```
npx hardhat run --network manta scripts/deploy.js
```

Hardhat akan mengompilasi kontrak dan mengirimkannya ke Manta Testnet, lalu mencetak alamat kontrak di konsol. Catat alamat-alamat ini untuk digunakan di aplikasi mobile nanti.

Tips: Pastikan Anda memiliki saldo testnet MANTA untuk biaya gas. Anda bisa memperoleh token uji (faucet) dari sumber Manta Pacific Testnet. Jika ada masalah jaringan, cek dokumentasi Manta atau diskusi komunitasnya.

3. Upload Video Course ke Livepeer

Video course perlu disimpan terdesentralisasi. Kita akan menggunakan **Livepeer** sebagai penyimpanan video. Berikut langkah umumnya:

1. **Dapatkan API Key Livepeer**. Daftar di Livepeer Studio dan buat API Key (token bearer).
2. **Gunakan SDK Livepeer**. Contoh kode upload video dengan JavaScript/Node.js:

```
import { Livepeer } from "livepeer";
import { Type } from "livepeer/models/components";

const livepeer = new Livepeer({
  apiKey: "<LIVEPEER_API_KEY>",
});

async function uploadVideo() {
  const result = await livepeer.asset.create({
    name: "course-video.mp4",
    staticMp4: true,
    // Kebijakan pemutaran (contoh: webhook untuk notifikasi)
    playbackPolicy: {
      type: Type.Webhook,
      webhookId: "1bde4o2i6xycudoy", // ganti ID webhook jika ada
      webhookContext: { "courseId": "1234" },
      refreshInterval: 600,
    },
    // Profil transkode (resolusi, bitrate, dll)
    profiles: [
      { name: "720p", bitrate: 3000000, fps: 30, height: 720, width: 1280 },
      // Tambah profil lain jika perlu
    ],
  });
  console.log("Livepeer asset:", result);
  // Dapatkan URL video
  if (result.status.static) {
    console.log("Video siap diputar di URL:", result.downloadUrl ||
result.assetId);
  }
}

uploadVideo().catch(console.error);
```

Kode di atas menggunakan SDK Livepeer untuk membuat asset video. **staticMp4: true** akan menghasilkan file MP4 yang siap diputar. Setelah selesai, output `result` berisi informasi tentang asset, termasuk URL playback atau `assetId`. Anda dapat memanggil `result.downloadUrl` atau `result.playbackUrl` (tergantung versi API) untuk mendapatkan link video yang bisa digunakan di aplikasi. (livepeer-js/docs/sdks/asset/README.md at main · livepeer/livepeer-js · GitHub) (livepeer-js/docs/sdks/asset/README.md at main · livepeer/livepeer-js · GitHub).

> **Catatan:** Jika jaringan lambat, pertimbangkan menggunakan *resumable upload* via TUS sesuai dokumentasi Livepeer. Di atas kami menggunakan metode simple; untuk skenario nyata, pelajari opsi lengkap di dokumentasi Livepeer.

3. **Simpan URL video**. Setelah video berhasil diunggah, simpan URL atau ID video. Misalnya, `videoUrl = result.downloadUrl`. Ini akan digunakan di metadata kursus.

4. Upload Metadata Course ke IPFS

Setelah video siap, kita perlu menyimpan metadata kursus (judul, deskripsi, thumbnail, URL video, dll) ke IPFS. Kita akan menggunakan **Thirdweb Storage** untuk ini.

1. **Siapkan metadata JSON**. Contoh struktur:

```
{
  "title": "Belajar Web3 dengan React Native",
  "description": "Kursus lengkap membangun aplikasi Web3 Mobile menggunakan React Native, Expo, dan Thirdweb SDK.",
  "thumbnail": "ipfs://QmThumbnailHash",
  "videoUrl": "ipfs://QmLivepeerVideoHash"
}
```

Gantilah `thumbnail` dengan CID gambar thumbnail (jika ada) dan `videoUrl` dengan CID video dari langkah sebelumnya. Thumbnail bisa diupload ke IPFS juga (bisa via Thirdweb atau layanan pinning IPFS).

2. **Upload dengan Thirdweb CLI** (opsional). Cara mudah tanpa coding:

```
npx thirdweb@latest upload path/to/metadata.json
```

CLI ini akan mengunggah berkas dan menampilkan **IPFS URI** (misal `ipfs://Qm...`) di konsol.

3. **Upload dengan Thirdweb SDK (kode)**. Jika Anda ingin programmatically, contoh menggunakan Thirdweb TypeScript/JS:

```
import { ThirdwebSDK } from "@thirdweb-dev/sdk";

const sdk = new ThirdwebSDK("mumbai"); // (jika ada jaringan untuk IPFS saja bisa pakai mana saja)
const metadata = {
  title: "Belajar Web3 dengan React Native",
  description: "Kursus lengkap...",
  thumbnail: "ipfs://<CID_thumbnail>",
  videoUrl: "ipfs://<CID_video>"
};
const uploadResult = await sdk.storage.upload({
```

```
  files: [
    {
      name: "course-metadata.json",
      data: metadata,
    },
  ],
});
console.log("Metadata uploaded to IPFS:", uploadResult);
// uploadResult berisi array URI, misal ["ipfs://Qm..."]
```

Contoh di atas menggunakan fungsi `upload` dari SDK thirdweb/storage (thirdweb TypeScript SDK). URI `ipfs://...` yang didapat merupakan link metadata kursus yang dapat dibagikan. Simpan URI ini (CID) di kontrak atau di database agar aplikasi mobile dapat membacanya.

Dengan metadata di IPFS, selanjutnya aplikasinya tinggal mengambil URI ini untuk menampilkan info kursus.

5. Setup Proyek React Native (Expo)

Selanjutnya, buat proyek mobile dengan React Native + Expo:

1. **Inisialisasi Proyek Expo**. Di terminal:

```
expo init Web3CourseApp
```

   Pilih template *blank* (dengan JavaScript atau TypeScript sesuai preferensi).
2. **Pasang Thirdweb React Native**:

```
cd Web3CourseApp
npm install @thirdweb-dev/react-native
```

3. **Edit `App.js` (atau `App.tsx`)**. Buat struktur dasar:

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';
import { ThirdwebProvider } from '@thirdweb-dev/react-native';

// ID client Thirdweb (dapat di dashboard Thirdweb)
const clientId = "YOUR_THIRDWEB_CLIENT_ID";

export default function App() {
  return (
    <ThirdwebProvider
      clientId={clientId}
      activeChain="manta">  {/* Ganti dengan nama atau chain ID Manta Pacific */}
      <View style={styles.container}>
        <Text>Halo, Web3 Mobile App!</Text>
      </View>
    </ThirdwebProvider>
  );
}
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

Jangan lupa ganti `clientId` dengan Client ID yang Anda dapat dari dashboard Thirdweb (thirdweb React Native SDK). Atur juga `activeChain` ke Manta Pacific (jika didukung) atau ke chain EVM testnet yang compatible.

4. **Jalankan Aplikasi**:

```
expo start
```

Buka di emulator Android atau perangkat fisik. Saat ini tampilan hanya teks “Halo, Web3 Mobile App!” sebagai pembuka.

6. Integrasi Thirdweb SDK untuk Web3 Wallet Connect

Kini kita tambahkan kemampuan pengguna untuk **menghubungkan wallet** (MetaMask, WalletConnect, dll) menggunakan Thirdweb. Cara termudah adalah menggunakan komponen *Connect* dari Thirdweb.

1. **Membuat klien Thirdweb**. Di `App.js`, import dan buat client:

```
import { createThirdwebClient } from "thirdweb";
import { ConnectEmbed, ThirdwebProvider } from "@thirdweb-dev/react-native";

const client = createThirdwebClient({
  clientId: clientId,
  chain: { slug: "manta" } // atau sesuaikan dengan network Thirdweb
});
```

2. **Gunakan komponen ConnectEmbed**. Ini membuat tombol connect secara inline (thirdweb React Native SDK):

```
export default function App() {
  return (
    <ThirdwebProvider client={client}>
      <View style={styles.container}>
        {/* Tombol Connect Wallet */}
        <ConnectEmbed client={client} />
        {/* Konten aplikasi lainnya */}
      </View>
    </ThirdwebProvider>
  );
}
```

Kode di atas menampilkan tombol "Connect Wallet". Ketika diklik, pengguna dapat memilih wallet (MetaMask, WalletConnect, dll). Setelah terhubung, Thirdweb menyimpan status koneksi, dan kita

bisa memanfaatkan hook seperti `useAddress`, `useContract`, dll. Contoh penggunaan `ConnectEmbed` ini diambil dari dokumentasi Thirdweb (thirdweb React Native SDK).

3. **Hak Akses DApp**. Pastikan DApp Anda didaftarkan di Thirdweb Dashboard agar `clientId` valid.

Dengan ini, aplikasi mobile sudah dapat meminta koneksi wallet user. Selanjutnya, kita akan mengakses kontrak melalui Thirdweb hooks.

7. Fitur Utama di Mobile App

Berikut fitur-fitur inti aplikasi yang harus diimplementasikan. Kita akan menggunakan Thirdweb React hooks (`useContract`, `useContractRead`, `useContractWrite`) untuk memanggil smart contract.

7.1. Menampilkan Daftar Course dari Kontrak

Misalnya kontrak `CourseFactory` memiliki fungsi `getAllCourses()`. Caranya:

```
import { useContract, useContractRead } from '@thirdweb-dev/react';

// Di dalam komponen...
const courseFactoryAddress = "0x..."; // alamat CourseFactory hasil deploy
const { contract: courseFactory } = useContract(courseFactoryAddress);
const { data: courses, isLoading } = useContractRead(courseFactory, "getAllCourses");
```

- Jika `courses` berisi array objek course, Anda bisa render dengan komponen seperti `FlatList`.
- Tampilkan minimal `title` dan `price` atau atribut penting lain.
- Contoh tampilan sederhana:

```
{isLoading ? (
  <Text>Loading courses...</Text>
) : (
  courses.map((c, idx) => (
    <View key={idx} style={styles.courseItem}>
      <Text style={styles.title}>{c.title}</Text>
      <Text>Harga: {c.price.toString()} ETH</Text>
    </View>
  ))
)}
```

- Sesuaikan field `c.title`, `c.price` dengan return value aktual kontrak.

7.2. Mint License NFT untuk Akses Course

Setelah user memilih course, kita sediakan tombol **Beli/Mint License** yang memanggil kontrak `CourseLicense` atau fungsi pembelian di `CourseFactory`.

Misalkan fungsi kontrak bernama `mintLicense(courseId) payable`. Cara panggil dengan Thirdweb:

```
import { useContractWrite } from '@thirdweb-dev/react';

// Di dalam komponen detail course:
const licenseContractAddress = "0x..."; // alamat CourseLicense
```

```
const { contract: licenseContract } = useContract(licenseContractAddress);
const { mutateAsync: mintLicense } = useContractWrite(licenseContract,
"mintLicense");

// Di event handler:
async function onBuyPress(courseId, price) {
  try {
    await mintLicense({ args: [courseId], value: price });
    alert("License minted! Anda sekarang dapat mengakses course.");
  } catch (err) {
    console.error(err);
    alert("Gagal mint license");
  }
}
```

- Parameter `value: price` mengirim ETH sesuai harga (dalam satuan Wei).
- Harga dan `courseId` didapat dari data course.
- Setelah transaksi berhasil, user mendapatkan NFT license.

7.3. Cek Validitas License NFT

Pastikan user memang memiliki license sebelum mengakses course. Misalnya kontrak `CourseLicense` punya fungsi `balanceOf(user, courseId)` atau mapping pengecekan. Anda bisa memanggilnya dengan `useContractRead`:

```
const { data: hasLicense } = useContractRead(licenseContract, "hasLicense", [address,
courseId]);
if (!hasLicense) {
  // Tampilkan pesan "Anda belum punya license"
}
```

Atau cukup cek kepemilikan NFT: jika kontrak adalah ERC-1155, gunakan `balanceOf`; jika ERC-721, periksa pemilik token.

7.4. Lihat Detail Course dan Sections

Saat user memilih course (dan sudah punya license), tampilkan detail kursus:

- Ambil metadata kursus dari IPFS (URI yang sudah diupload sebelumnya). Misal:

```
import { MediaRenderer, useStorageDownload } from "@thirdweb-dev/react";

const { data: metadata } = useStorageDownload(ipfsUri);
// metadata.title, metadata.description, metadata.thumbnail, metadata.videoUrl
```

- Tampilkan judul dan deskripsi.
- Untuk video, gunakan `MediaRenderer` dari Thirdweb untuk men-stream dari IPFS/Livepeer:

```
<MediaRenderer src={metadata.videoUrl} />
```

Ini akan render video yang mendukung sumber IPFS maupun URL biasa (Upload Files to IPFS | thirdweb Storage).

- Jika ada *sections*, misalnya daftar link video atau teks materi, tampilkan dalam list.

7.5. Menandai Progress Course (Complete Section)

Kontrak `ProgressTracker` memungkinkan mencatat seberapa jauh user telah belajar. Misal fungsi `completeSection(courseId, sectionId)`. Contoh panggilan:

```
const { contract: progressContract } = useContract("0x..."); // alamat ProgressTracker
const { mutateAsync: completeSection } = useContractWrite(progressContract, "completeSection");

async function onCompleteSection(courseId, sectionId) {
  await completeSection({ args: [courseId, sectionId] });
  alert("Bagian selesai!");
}
```

Terapkan pada button di setiap section; simpan state dan update UI agar bagian tersebut tercoret atau ditandai selesai. Anda juga bisa membaca progress dengan `useContractRead` jika kontrak menyimpan data.

7.6. Mint Sertifikat Setelah Menyelesaikan Course

Setelah semua section selesai, user bisa mint sertifikat (kontrak `CertificateManager`). Misalnya fungsi `mintCertificate(courseId)`:

```
const { contract: certContract } = useContract("0x..."); // alamat CertificateManager
const { mutateAsync: mintCertificate } = useContractWrite(certContract, "mintCertificate");

async function onMintCertificate(courseId) {
  await mintCertificate({ args: [courseId] });
  alert("Sertifikat berhasil di-mint! 🎉");
}
```

Tombol ini hanya muncul jika semua bagian course sudah selesai, sesuai logika aplikasi atau smart contract.

7.7. Pembayaran dan Split Fee

Ketika user membeli license NFT, transaksi ETH yang dikirim bisa otomatis terbagi antara **creator** dan **platform**. Biasanya logika ini ada di smart contract `CourseFactory`. Misal:

```
uint256 creatorShare = price * 90 / 100;
uint256 platformShare = price - creatorShare;
payable(course.creator).transfer(creatorShare);
payable(owner()).transfer(platformShare);
```

Sebagai aplikasi, kita cukup **memastikan nilai** `msg.value` **sesuai**. Contoh pada `onBuyPress` di atas kita sudah mengirim `value: price`. Selanjutnya smart contract akan menangani pembagian tersebut.

8. Implementasi Payment Split ke Creator dan Platform

Pada langkah pembelian (mint License), ETH yang dikirim melalui `mintLicense` akan diproses oleh kontrak. Misalnya `CourseFactory` menyimpan alamat creator untuk setiap course. Saat mint, kontrak bisa membagi pembayaran seperti contoh di atas. Tidak ada kode tambahan di aplikasi, cukup memanggil fungsi kontrak dengan nilai (`value`) yang sudah mencakup total harga.

Jika Anda mengembangkan kontrak sendiri, pastikan menulis fungsi **splitting** tersebut. Jika kontrak sudah tersedia, pastikan membaca dokumentasi atau kode sumbernya untuk mengetahui parameter yang benar. Contoh di frontend:

```
await mintLicense({ args: [courseId], value: ethers.utils.parseEther("0.1") });
```

Kode ini mengirim 0.1 ETH yang kemudian kontrak bagi ke creator dan platform secara otomatis.

9. Desain UI/UX Dasar

Untuk UI/UX sederhana, Anda bisa mempertimbangkan wireframe berikut (sebagai contoh):

- **Beranda**: Daftar kursus (thumbnail + judul + tombol Beli).
- **Halaman Course Detail**: Judul, deskripsi, video (gunakan `<MediaRenderer />` (Upload Files to IPFS | thirdweb Storage)), daftar section dengan checkbox/tombol "Complete".
- **Menu Wallet/User**: Tombol Connect Wallet (di header) dan status koneksi (alamat wallet).
- **Profile/Certificate**: Opsi lihat sertifikat yang sudah di-mint.

Misalnya, menggunakan komponen React Native seperti `View`, `Text`, `Button`, dan `FlatList`. Contoh pemakaian `MediaRenderer` untuk menampilkan video IPFS:

```
import { MediaRenderer } from "@thirdweb-dev/react-native";

<MediaRenderer src={metadata.videoUrl} style={{ width: 300, height: 200 }} />
```

Komponen ini otomatis menangani pemutaran media (video, gambar, audio) dari IPFS (Upload Files to IPFS | thirdweb Storage). Untuk thumbnail biasa, gunakan `Image` standar React Native.

Untuk wireframe kasar, bayangkan layout satu kolom: header wallet di atas, lalu daftar kursus. Tampilan detail kursus bisa bertingkat: judul, video, tombol mark selesai. Anda dapat menggunakan library UI sederhana seperti React Native Paper atau built-in styling.

Contoh kode penampilan daftar kursus:

```
<FlatList
  data={courses}
  keyExtractor={(item) => item.id.toString()}
  renderItem={({ item }) => (
    <TouchableOpacity onPress={() => goToDetail(item.id)}>
      <View style={styles.courseCard}>
        <Text style={styles.courseTitle}>{item.title}</Text>
```

```
          <Text style={styles.coursePrice}>Harga: {item.price} ETH</Text>
        </View>
      </TouchableOpacity>
    )}
  />
```

Setiap `TouchableOpacity` bisa mengarahkan ke halaman detail kursus dengan `react-navigation` atau navigasi lainnya.

10. Testing End-to-End (Deploy → Mint → Akses Course → Selesaikan → Mint Certificate)

Setelah semuanya diimplementasikan, lakukan pengujian menyeluruh:

1. **Deploy Kontrak**: Pastikan smart contract sudah terdeploy di testnet (Manta Pacific) dan alamatnya benar di aplikasi.
2. **Persiapkan Data**: Upload video ke Livepeer, upload metadata ke IPFS. Catat `metadataURI`.
3. **Run App di Perangkat**: Jalankan aplikasi expo di emulator/perangkat Android.
4. **Connect Wallet**: Klik tombol *Connect Wallet*, hubungkan ke akun testnet Anda di Metamask. Pastikan saldo cukup.
5. **Beli/Mint License**: Pilih salah satu course, tekan *Buy License*. Periksa MetaMask dan konfirmasi transaksi. Setelah selesai, NFT license harus muncul di wallet.
6. **Cek Akses Course**: Masuk ke halaman detail course, harusnya bisa melihat video dan materi karena sudah ada license. Jika belum, periksa logika validasi license.
7. **Mark Progress**: Tandai semua section sebagai selesai. Pastikan fungsi `completeSection` memanggil kontrak dan tidak error.
8. **Mint Certificate**: Setelah progress 100%, tombol *Mint Certificate* muncul. Tekan dan konfirmasi transaksi. NFT sertifikat seharusnya muncul (dan mungkin tampil di aplikasi atau wallet).
9. **Pembagian Pembayaran**: Cek contract owner (platform) dan creator wallet, pastikan dana yang diterima sesuai pembagian (90/10 atau sebagaimana diatur).
10. **Uji Kegagalan**: Cobalah kasus-kasus negatif, misal mencoba akses course tanpa license, atau mint certificate sebelum selesai. Aplikasi seharusnya menolak dan menampilkan pesan error.

Jika semua berhasil, fitur inti sudah teruji. Dokumentasikan langkah dan hasilnya agar mudah diperbaiki jika ada masalah.

Ringkasan: Dengan mengikuti langkah-langkah di atas, Anda akan membuat aplikasi mobile Web3 di Android (React Native + Expo) yang terhubung ke blockchain Manta Pacific, menggunakan Thirdweb SDK untuk integrasi Web3 (wallet connect, storage IPFS), serta Livepeer untuk menyimpan dan memutar video kursus. Tutorial ini menggunakan pendekatan *practical* dengan potongan kode lengkap untuk setiap bagian. Langkah-langkahnya disusun agar mudah diikuti developer junior Web3. Semoga sukses!

Sumber Referensi: Contoh penggunaan Thirdweb SDK dan Livepeer diambil dari dokumentasi resmi (thirdweb React Native SDK) (livepeer-js/docs/sdks/asset/README.md at main · livepeer/livepeer-js · GitHub) (thirdweb TypeScript SDK) (Upload Files to IPFS | thirdweb Storage) (thirdweb React Native SDK). Data di atas bersifat ilustrasi dan perlu disesuaikan dengan lingkungan Anda.

Mantap, sekarang aku akan mulai membuatkan panduan super lengkap untukmu! 🚀

Aku akan buatkan tutorial full dari awal:

- Setup backend (Livepeer untuk upload video, Thirdweb IPFS untuk upload thumbnail & sertifikat dinamis)
- Setup frontend React Native dengan Expo + Thirdweb SDK
- Menjelaskan interaksi penuh dengan smart contract yang sudah kamu deploy
- Menjelaskan pembuatan gambar sertifikat dinamis berdasarkan data blockchain
- Menyambungkan semua alur, step-by-step, dengan bahasa super mudah dimengerti untuk level pemula

Aku akan langsung mulai dan akan update kamu ketika panduannya sudah siap ya!

Panduan Lengkap: Membangun Platform Edukasi Web3 (Blockchain & NFT) dengan React Native, Livepeer, Thirdweb

Panduan ini menjelaskan langkah demi langkah cara membuat platform edukasi berbasis blockchain dan NFT dengan teknologi React Native (Expo), Livepeer, dan Thirdweb. Kami akan membahas mulai dari *setup* lingkungan pengembangan, membuat proyek Expo, menghubungkan dompet (wallet), hingga mengunggah video dengan Livepeer, menyimpan data ke IPFS, interaksi penuh dengan smart contract Eduverse (di jaringan Manta Pacific Testnet), serta pembuatan sertifikat NFT dinamis. Penjelasan disertai contoh kode yang lengkap dan analogi sederhana agar mudah dipahami. Setiap bagian diberi referensi ke dokumentasi resmi untuk memperkuat konsep.

---

1. Persiapan Lingkungan Kerja

Sebelum memulai, pastikan Anda telah menyiapkan beberapa hal berikut:

- **Node.js & npm/yarn:** Pastikan sudah terpasang Node.js terbaru. NPM (atau `yarn`) dibutuhkan untuk mengelola paket dan menjalankan Expo.
- **Expo CLI:** Pasang Expo secara global dengan `npm install -g expo-cli` (untuk membuat dan menjalankan proyek React Native) (Getting Started - Livepeer Docs) (Getting Started - Livepeer Docs).
- **Akun Thirdweb:** Daftar akun di thirdweb.com untuk mendapat *Client ID* (API key) untuk integrasi SDK Thirdweb. Client ID ini diperlukan untuk `ThirdwebProvider` di aplikasi React Native (Upload Files to IPFS | thirdweb Storage) (How to Fully Customize the ConnectWallet Button in the React Native SDK).
- **Akun Livepeer:** Daftar akun di livepeer.studio untuk mendapat *API Key* Livepeer (Studio) yang digunakan untuk mengunggah video. Ini diperlukan untuk membuat client Livepeer (Getting Started - Livepeer Docs) (Getting Started - Livepeer Docs).
- **Dompet Crypto & Testnet Manta:** Siapkan dompet crypto (misalnya MetaMask atau In-App Wallet Thirdweb) dan tambahkan jaringan **Manta Pacific Sepolia Testnet**. Rincian chain: Chain ID **3441006**, RPC `https://3441006.rpc.thirdweb.com` (Manta Pacific Sepolia Testnet: RPC and Chain Settings). Anda dapat mendapat token testnet (ETH) dari faucet Manta Pacific Sepolia agar bisa membayar gas di jaringan ini.

> **Analogi:** Anggap blockchain seperti buku besar (ledger) publik, dan NFT seperti sertifikat digital unik yang tercatat di buku besar itu. Thirdweb dan Livepeer adalah “alat bantu” (toolkit) yang mempermudah kita menulis dan membaca buku besar tersebut, serta menyimpan berkas (video, gambar) ke IPFS.

---

2. Membuat Proyek React Native (Expo)

2.1 Inisialisasi Proyek

Buat proyek baru Expo dengan template blank:

```
expo init EduverseApp
```

Pilih opsi **blank (TypeScript)** atau **blank (JavaScript)**. Setelah selesai, masuk ke direktori proyek:

```
cd EduverseApp
```

2.2 Instalasi Dependensi

Pasang dependensi utama:

- **Thirdweb SDK:** Untuk integrasi Web3, gunakan `@thirdweb-dev/react-native`. Juga pasang adapter jika diperlukan (tergantung dokumentasi Thirdweb versi terbaru). Contoh:

  ```
  npm install @thirdweb-dev/react-native
  ```

  atau

  ```
  yarn add @thirdweb-dev/react-native
  ```

- **Livepeer React Native:** Install paket Livepeer untuk React Native beserta peer-dependencies:

  ```
  expo install @livepeer/react-native expo-av react-native-svg
  ```

  Ini sesuai dokumentasi Livepeer, yang menyebutkan `@livepeer/react-native`, `expo-av`, dan `react-native-svg` untuk pemrosesan video (Getting Started - Livepeer Docs).
- **Thirdweb Storage (Node, jika diperlukan):** Jika Anda membuat sertifikat gambar secara dinamis di sisi Node atau server, pasang `@thirdweb-dev/storage`:

  ```
  npm install @thirdweb-dev/storage
  ```

- **Canvas (Node):** Untuk membuat gambar sertifikat di server, Anda bisa pasang `canvas` untuk Node.js (jika tidak menggunakan Expo). Contoh:

```
npm install canvas
```

(Pada tutorial ini akan kami tampilkan contoh Node Canvas, tetapi pengembang bisa menggunakan metode lain seperti `fabric.js` atau library khusus gambar jika di mobile.)

Pastikan juga editor Anda siap (VS Code, WebStorm, dsb).

2.3 Konfigurasi Lingkungan

Buat file `.env` di root proyek untuk menyimpan *Client ID Thirdweb* dan *Livepeer API Key*. Contoh `.env`:

```
EXPO_PUBLIC_THIRDWEB_CLIENT_ID=YOUR_THIRDWEB_CLIENT_ID
LIVEPEER_API_KEY=YOUR_LIVEPEER_API_KEY
```

Lalu, install package `expo-constants` untuk membaca env di Expo (atau gunakan `process.env` sesuai setup). Jangan lupa menambahkan `babel-plugin-inline-dotenv` atau sejenis jika menggunakan React Native dengan Metro (tergantung kebutuhan).

---

3. Koneksi Wallet dengan Thirdweb

3.1 Konfigurasi ThirdwebProvider

Di `App.js` atau `App.tsx`, bungkus aplikasi dengan `ThirdwebProvider` sehingga semua komponen dapat terhubung ke Thirdweb:

```
import React from 'react';
import { ThirdwebProvider, ChainId } from '@thirdweb-dev/react-native';

// Ambil Client ID dari .env
const clientId = Constants.manifest.extra.expoPublicThirdwebClientId;

export default function App() {
  return (
    <ThirdwebProvider
      clientId={clientId}
      // Atur chain ke Manta Pacific Sepolia Testnet
      activeChain={ChainId.MantaPacificSepolia}
    >
      {/* Konten aplikasi */}
    </ThirdwebProvider>
  );
}
```

Dalam contoh di atas, `ChainId.MantaPacificSepolia` mengacu pada nilai *3441006*, sesuai chain ID testnet Manta (Manta Pacific Sepolia Testnet: RPC and Chain Settings). Thirdweb secara otomatis menggunakan RPC yang sesuai (lihat RPC di chain list Thirdweb).

Jika `ChainId` tersebut belum tersedia, Anda bisa juga menggunakan string atau angka langsung: `activeChain={3441006}`. Yang penting, pastikan jaringan sudah ditambahkan di dompet pengguna dan di ThirdwebProvider.

3.2 Tombol “Connect Wallet”

Agar pengguna dapat menghubungkan dompet, kita gunakan komponen `<ConnectWallet />` dari Thirdweb. Contoh penggunaan sederhana:

```
import React from 'react';
import { View } from 'react-native';
import { ConnectWallet } from '@thirdweb-dev/react-native';

export default function WalletConnectScreen() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <ConnectWallet buttonTitle="Hubungkan Wallet" />
    </View>
  );
}
```

Kode di atas akan menampilkan tombol **"Hubungkan Wallet"**. Saat ditekan, akan muncul modal pilihan wallet (MetaMask Mobile, WalletConnect, atau bahkan *In-App Wallet* Thirdweb). Komponen ini sudah menangani proses otentikasi ke Thirdweb (How to Fully Customize the ConnectWallet Button in the React Native SDK) (How to Fully Customize the ConnectWallet Button in the React Native SDK).

> **Referensi:** Contoh import dan penggunaan `ConnectWallet` dapat dilihat di dokumentasi Thirdweb React Native (How to Fully Customize the ConnectWallet Button in the React Native SDK). Komponen ini membuat integrasi wallet sangat mudah: pengguna cukup klik, lalu koneksi blockchain siap.

---

4. Mengunggah Video Course dengan Livepeer

4.1 Konfigurasi Livepeer Client

Livepeer menyediakan *React SDK* untuk mengunggah dan memutar video. Ikuti langkah berikut:

1. **Buat Livepeer Client:** Di file `App.js` (sama seperti ThirdwebProvider), buat client Livepeer dengan `createReactClient`. Contoh:

```
import { LivepeerConfig, createReactClient, studioProvider } from
'@livepeer/react-native';

const livepeerClient = createReactClient({
  provider: studioProvider({ apiKey: Constants.manifest.extra.livepeerApiKey }),
});

export default function App() {
  return (
    <LivepeerConfig client={livepeerClient}>
```

```
      {/* Konten aplikasi */}
    </LivepeerConfig>
  );
}
```

- `studioProvider({ apiKey })` adalah cara memberi tahu Livepeer untuk menggunakan akun Livepeer Studio Anda. Masukkan API Key yang Anda dapatkan (Getting Started - Livepeer Docs).
- Bungkus `<LivepeerConfig>` mengisolasi context Livepeer agar hook di komponen manapun bisa digunakan (Getting Started - Livepeer Docs) (Getting Started - Livepeer Docs).

2. **Siapkan antarmuka unggah:** Anda bisa membuat komponen untuk memilih file video (misalnya menggunakan `expo-document-picker` atau modul khusus kamera). Setelah mendapatkan file video (misalnya objek `File` atau `uri`), kita akan panggil hook `useCreateAsset`.
3. **Hook `useCreateAsset`:** Ini adalah React Hook Livepeer untuk mengunggah (upload) video. Contoh penggunaan:

```
import React, { useState } from 'react';
import { Button, Text } from 'react-native';
import { useCreateAsset } from '@livepeer/react-native';

function UploadVideo() {
  const [file, setFile] = useState<File | null>(null);
  const { mutate: createAsset, status, progress, error } = useCreateAsset(
    file
      ? {
          sources: [
            {
              name: file.name,
              file: file,
              storage: {
                ipfs: true, // simpan hasil transcoding ke IPFS
                metadata: {
                  name: "Video Course",
                  description: "Video lesson for the course",
                },
              },
            },
          ],
        }
      : null
  );

  const handlePickVideo = async () => {
    // ... logika memilih file dari device (misalnya using expo-document-picker)
    // setelah itu: setFile(pilihanFile);
  };

  const handleUpload = () => {
    createAsset?.(); // mulai upload
  };

  return (
```

```
      <>
        <Button title="Pilih Video" onPress={handlePickVideo} />
        <Button title="Unggah Video ke Livepeer" onPress={handleUpload} />
        <Text>Status: {status}</Text>
        <Text>Progress: {progress.progress || 0}%</Text>
        {error && <Text>Error: {error.message}</Text>}
      </>
    );
  }
```

Penjelasan:

- Ketika `file` sudah dipilih, kita panggil hook `useCreateAsset({ sources: [...] })` (useCreateAsset - Livepeer Docs). Hook ini menyediakan fungsi `createAsset` yang akan mulai proses unggah video.
- Proses unggah menggunakan protokol **TUS (resumable upload)**, sehingga unggahan video besar lebih andal (useCreateAsset - Livepeer Docs).
- Pilihan `ipfs: true` pada `storage` memastikan video hasil transcoding akan di-*pin* ke IPFS.
- Anda bisa melihat `status`, `progress`, dan `error` untuk memantau proses. Saat selesai, Livepeer akan menyediakan *assetId* dan *playbackId* yang bisa dipakai untuk memutar video nanti.

> **Referensi:** Contoh di atas diadaptasi dari dokumentasi Livepeer React Native. Hook `useCreateAsset` secara otomatis menangani unggahan (menggunakan TUS) dan memantau status hingga selesai (useCreateAsset - Livepeer Docs) (useCreateAsset - Livepeer Docs).

4. **Memutar Video:** Setelah video terunggah, Anda bisa menggunakan komponen `<Player>` dari Livepeer atau `<Video>` React Native untuk memutar video tersebut. Gunakan `playbackId` yang diperoleh dari hasil `createAsset`. (Panduan pemutaran dapat dilihat di dokumentasi Livepeer, namun di luar cakupan tutorial ini.)

---

5. Mengunggah Thumbnail dan Sertifikat ke IPFS (Thirdweb Storage)

Selain video, kita perlu mengunggah gambar **thumbnail kursus** dan **sertifikat NFT** ke IPFS. Thirdweb menyediakan layanan *Storage* terdesentralisasi yang terintegrasi dengan IPFS. Caranya:

5.1 Hook `useStorageUpload` (React Native)

Anda bisa menggunakan hook `useStorageUpload` dari Thirdweb untuk mengunggah file dalam aplikasi React Native/Expo. Contoh:

```
import React from 'react';
import { Button } from 'react-native';
import { useStorageUpload } from '@thirdweb-dev/react-native';

function UploadImages() {
  const { mutateAsync: upload } = useStorageUpload();

  const handleUploadFiles = async () => {
    // contoh: kita asumsikan 'files' adalah array File atau Buffer gambar
```

```
    const imageFiles = [ /* File atau Blob gambar */ ];
    try {
      const uris = await upload({ data: imageFiles });
      console.log("IPFS URIs:", uris);
      // uris[0] misalnya "ipfs://Qm.../0"
    } catch (err) {
      console.error(err);
    }
  };

  return <Button title="Upload ke IPFS" onPress={handleUploadFiles} />;
}
```

- `useStorageUpload` mengembalikan fungsi `upload({ data })` yang menerima array file/data yang ingin diunggah ke IPFS (Upload Files to IPFS | thirdweb Storage).
- Hasilnya adalah array URI IPFS (misal `"ipfs://QmXYZ"`). Anda bisa menyimpan URI ini sebagai `imageURI` dalam metadata NFT nantinya.
- Dengan cara ini, thumbnail kursus dan gambar sertifikat dapat dikirim ke IPFS dan dijadikan sumber immutable untuk NFT.

> **Referensi:** Dokumentasi Thirdweb Storage menunjukkan penggunaan hook `useStorageUpload` di React/React Native untuk unggah file ke IPFS (Upload Files to IPFS | thirdweb Storage). Contoh di atas memakai `mutateAsync` untuk operasi asinkron.

5.2 Storage SDK (Node)

Sebagai alternatif (misalnya jika pembuatan sertifikat dilakukan di server/komputer), Thirdweb menyediakan **Storage SDK** untuk Node.js. Contoh penggunaan:

```
import { ThirdwebStorage } from "@thirdweb-dev/storage";
import { readFileSync } from "fs";

async function uploadImageToIPFS(path) {
  const storage = new ThirdwebStorage();
  const file = readFileSync(path); // baca file gambar
  const uri = await storage.upload(file);
  console.log("Uploaded to IPFS:", uri);
  return uri; // e.g. "ipfs://Qm..."
}
```

Dokumentasi menyebutkan contoh serupa untuk mengunggah file (@thirdweb-dev/storage - npm). Metode ini berguna jika Anda membuat sertifikat dinamis secara otomatis di backend.

---

6. Integrasi Smart Contract Eduverse dengan Thirdweb

Misalkan kontrak pintar **Eduverse Platform** sudah dideploy di Manta Pacific Testnet dan menyediakan fungsi seperti `mintLicense`, `hasAccess`, `mintCertificate`, dsb. Kita akan menggunakan Thirdweb SDK untuk berinteraksi penuh dengan kontrak tersebut.

6.1 Mendapatkan Objek Kontrak

Pertama, buat referensi kontrak dengan alamat yang diberikan. Misalnya alamat kontrak `0xEduVerse...`. Contoh penggunaan Thirdweb:

```
import React from 'react';
import { useContract } from '@thirdweb-dev/react-native';

export default function ContractInteraction() {
  // Ganti dengan alamat kontrak Eduverse Platform Anda
  const contractAddress = "0xEDUVERSE_CONTRACT_ADDRESS";
  const { contract } = useContract(contractAddress);

  // ... gunakan 'contract' untuk baca/tulis
  return null;
}
```

Atau menggunakan `getContract` jika belum ada `useContract`:

```
import { getContract, useReadContract, useSendAndConfirmTransaction } from
'@thirdweb-dev/react-native';

async function example() {
  const contract = await getContract({
    client: thirdwebClient,           // Thirdweb client dari ThirdwebProvider
    chain: ChainId.MantaPacificSepolia,
    address: contractAddress,
  });
}
```

Dengan ini, kita memiliki objek kontrak yang siap dipanggil fungsi-fungsinya.

6.2 Baca Data Kontrak (NFT License, Access)

Untuk memeriksa akses atau data di kontrak, gunakan hook pembacaan. Contoh: memeriksa apakah pengguna sudah punya lisensi (license) untuk suatu kursus:

```
import { useContractRead } from '@thirdweb-dev/react-native';

function CheckAccess({ contract, userAddress, courseId }) {
  const { data: hasLicense, isLoading } = useContractRead(contract, "hasLicense",
[userAddress, courseId]);

  if (isLoading) return <Text>Memeriksa akses...</Text>;
  return (
    <Text>
      {hasLicense
        ? "Anda sudah memiliki license untuk kursus ini."
        : "Anda belum memiliki license."}
    </Text>
  );
}
```

Di sini `useContractRead(contract, "hasLicense", [...])` memanggil fungsi view `hasLicense(address,uint256) returns (bool)` pada kontrak. Contoh ini mirip prinsip pada dokumentasi Thirdweb, meski mereka mencontohkan ERC-20 `totalSupply` (Build Web3 Mobile Apps with React Native on Mode).

> **Referensi:** Cara memanggil fungsi baca (read) pada kontrak Thirdweb dapat dengan `useReadContract` atau `useContractRead`, seperti pada tutorial Thirdweb (Build Web3 Mobile Apps with React Native on Mode).

6.3 Menulis Transaksi (Mint License & Mint Sertifikat)

Untuk menulis transaksi (misalnya mint NFT), gunakan hook `useSendAndConfirmTransaction` atau `useContractWrite`. Contoh mint license NFT:

```
import { useSendAndConfirmTransaction } from '@thirdweb-dev/react-native';

function MintLicenseButton({ contract, toAddress, courseId }) {
  const { sendMutation: sendTransaction } = useSendAndConfirmTransaction();

  const mintLicense = async () => {
    // Siapkan transaksi mint (asumsi kontrak punya fungsi
mintLicense(address,uint256))
    const txn = await contract.prepare("mintLicense", [toAddress, courseId]);
    sendTransaction(txn);
  };

  return <Button title="Mint License NFT" onPress={mintLicense} />;
}
```

Dalam kode di atas:

- `prepare("mintLicense", [...])` adalah asumsi contoh fungsi pada kontrak Eduverse. Sesuaikan dengan ABI kontrak Anda.
- Setelah `sendTransaction(txn)`, Thirdweb SDK akan meminta tanda tangan (signature) dan mengirim transaksi ke blockchain.

Tutorial Thirdweb memberi contoh serupa menggunakan `useSendAndConfirmTransaction`, `prepareContractCall`, atau helper `mintTo` untuk standar ERC-721 (Build Web3 Mobile Apps with React Native on Mode) (Build Web3 Mobile Apps with React Native on Mode). Prinsipnya sama: siapkan panggilan kontrak lalu kirim.

Setelah transaksi terkonfirmasi, pengguna akan menerima NFT License di dompet mereka.

> **Referensi:** Dokumentasi Thirdweb contoh minting NFT dengan `useSendAndConfirmTransaction`, `prepareContractCall` atau `mintTo` (Build Web3 Mobile Apps with React Native on Mode) (Build Web3 Mobile Apps with React Native on Mode).

---

7. Membuat dan Minting Sertifikat NFT Dinamis

Bagian akhir adalah membuat sertifikat digital (NFT) untuk setiap siswa yang menyelesaikan kursus. Prosesnya:

1. Buat gambar sertifikat dengan detail dinamis (nama siswa, nama kursus, tanggal).
2. Unggah gambar tersebut ke IPFS (gunakan Thirdweb Storage seperti langkah 5).
3. Mint NFT sertifikat di kontrak dengan `tokenURI` yang mengarah ke gambar IPFS.

7.1 Menciptakan Gambar Sertifikat

Kita dapat membuat sertifikat dengan teknik canvas. Misal di Node.js:

```
import { createCanvas, loadImage } from 'canvas';

// Data dinamis
const studentName = "Budi Santoso";
const courseName = "Belajar Blockchain";
const issueDate = "01 Januari 2025";

// Ukuran sertifikat (contoh: 600x400 px)
const width = 600;
const height = 400;

// Buat kanvas
const canvas = createCanvas(width, height);
const ctx = canvas.getContext('2d');

// Gambar latar putih
ctx.fillStyle = '#FFFFFF';
ctx.fillRect(0, 0, width, height);

// Tambahkan teks
ctx.fillStyle = '#000000';
ctx.font = 'bold 24px Arial';
ctx.fillText(`Sertifikat Penyelesaian`, 150, 50);
ctx.font = '20px Arial';
ctx.fillText(`Diberikan kepada: ${studentName}`, 50, 150);
ctx.fillText(`Untuk kursus: ${courseName}`, 50, 200);
ctx.fillText(`Tanggal: ${issueDate}`, 50, 250);

// Simpan sebagai buffer PNG
const buffer = canvas.toBuffer('image/png');
// (Anda bisa menyimpan ke file: fs.writeFileSync('cert.png', buffer))
```

Analoginya, `canvas` bertindak seperti kertas gambar digital, dan `fillText` menuliskan teks di atasnya (Drawing text - Web APIs | MDN). Referensi MDN menjelaskan metode `fillText(text, x, y)` sebagai cara menggambar teks pada kanvas (Drawing text - Web APIs | MDN) (Drawing text - Web APIs | MDN). Kode di atas membuat sertifikat sederhana.

7.2 Unggah Sertifikat ke IPFS

Setelah gambar siap (misal di `buffer`), unggah ke IPFS:

```
const storage = new ThirdwebStorage({ clientId: YOUR_CLIENT_ID }); // jika
menggunakan Thirdweb Storage
const cid = await storage.upload(buffer);
console.log("Gambar di-IPFS dengan URI:", cid);
```

Atau jika Anda menjalankan di React Native, ubah `buffer` menjadi objek File/Blob dan gunakan hook `useStorageUpload`. Hasilnya `cid` berupa `ipfs://...`.

7.3 Mint NFT Sertifikat

Dengan URI IPFS gambar sertifikat (misal `ipfs://Qm.../0`), kita mint sertifikat NFT:

```
// Di komponen React Native
const { sendMutation: sendTransaction } = useSendAndConfirmTransaction();

const mintCertificateNFT = async () => {
  const transaction = await contract.prepare("mintCertificate", [studentAddress,
certificateCid]);
  sendTransaction(transaction);
};
```

Asumsinya kontrak Eduverse memiliki fungsi `mintCertificate(address,string)` yang menerima alamat penerima dan URI metadata (atau langsung URI gambar). Anda dapat menyesuaikan dengan fungsi di kontrak.

Setelah transaksi ini, siswa akan mendapatkan NFT sertifikat di dompetnya, yang merujuk ke gambar sertifikat di IPFS yang berisi nama dan detail kursus mereka.

> **Referensi:** Teknik menggambar teks pada kanvas bisa dilihat di MDN: `ctx.fillText("Hello world", x, y)` sebagai contoh (Drawing text - Web APIs | MDN). Sedangkan contoh penggunaan Storage SDK Thirdweb untuk upload file tercantum di dokumentasi mereka (@thirdweb-dev/storage - npm).

---

8. Ringkasan Alur Kerja

Berikut ringkasan langkah utama dalam proyek ini, secara urut:

1. **Siapkan Lingkungan:** Install Node.js, Expo CLI, pasang dependensi Thirdweb dan Livepeer (Getting Started - Livepeer Docs) (@thirdweb-dev/storage - npm). Daftarkan akun Thirdweb & Livepeer untuk API keys. Tambahkan Manta Pacific Sepolia ke dompet.
2. **Buat Proyek Expo:** `expo init`, lalu import `ThirdwebProvider` dan `LivepeerConfig` di `App.js`. Konfigurasi ActiveChain ke Manta Pacific Sepolia (Manta Pacific Sepolia Testnet: RPC and Chain Settings) (Getting Started - Livepeer Docs).
3. **Koneksikan Wallet:** Gunakan `<ConnectWallet />` (atau `<ConnectButton />`) dari Thirdweb agar pengguna dapat log in dengan dompet mereka (How to Fully Customize the ConnectWallet Button in the React Native SDK).
4. **Upload Video Course:** Gunakan hook `useCreateAsset` dari Livepeer untuk mengunggah video kursus. Pastikan video di-*pin* ke IPFS jika perlu (useCreateAsset - Livepeer Docs) (useCreateAsset

- Livepeer Docs).

5. **Upload Gambar ke IPFS:** Pakai hook `useStorageUpload` (atau `ThirdwebStorage` di server) untuk mengunggah thumbnail kursus dan sertifikat ke IPFS (Upload Files to IPFS | thirdweb Storage) (@thirdweb-dev/storage - npm).
6. **Interaksi Smart Contract:** Buat objek kontrak Thirdweb dengan `useContract` atau `getContract`, lalu panggil fungsi baca/tulis. Contoh: `useContractRead(contract, "hasLicense", [...])` atau `useSendAndConfirmTransaction` untuk mint NFT (Build Web3 Mobile Apps with React Native on Mode) (Build Web3 Mobile Apps with React Native on Mode).
7. **Mint License NFT:** Ketika pengguna membeli kursus, panggil fungsi kontrak untuk mint NFT lisensi (license). Tanda tangani transaksi via wallet.
8. **Generate Sertifikat Dinamis:** Buat gambar sertifikat di server dengan canvas (menggunakan nama siswa, kursus, tanggal) (Drawing text - Web APIs | MDN) (Drawing text - Web APIs | MDN).
9. **Mint Sertifikat NFT:** Unggah gambar sertifikat ke IPFS, lalu panggil kontrak untuk mint NFT sertifikat ke alamat siswa.

Setiap langkah di atas kami jelaskan lengkap agar pengembang pemula bisa mengikuti dan memahami prosesnya. Semoga tutorial ini bermanfaat dalam membangun platform edukasi berbasis blockchain dan NFT yang modern! 🎓🚀

Sumber Referensi: Dokumentasi dan blog Thirdweb (SDK React Native, Storage) (Upload Files to IPFS | thirdweb Storage) (Build Web3 Mobile Apps with React Native on Mode); Dokumentasi Livepeer (React Native SDK, upload) (Getting Started - Livepeer Docs) (useCreateAsset - Livepeer Docs); MDN Web Docs (Canvas API) (Drawing text - Web APIs | MDN) (Drawing text - Web APIs | MDN). Semua informasi di atas diambil dari sumber resmi untuk memastikan keakuratan.