```
/*
 AdEarn – Refactored MVP with AI Ad Refresh + Full Admin UI
 Status: MVP / Demo (AI mocked – production ready structure)
*/

const express = require('express');
const sqlite3 = require('sqlite3').verbose();
const bcrypt = require('bcrypt');
const session = require('express-session');
const multer = require('multer');
const nodemailer = require('nodemailer');

const app = express();
const db = new sqlite3.Database('./adearn.db');

// ============================
// MIDDLEWARE
// ============================
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(session({ secret: 'adearn_secret', resave: false, saveUninitialized: true }));
app.use('/uploads', express.static('uploads'));

const upload = multer({ dest: 'uploads/' });

// ============================
// DATABASE SETUP
// ============================

db.serialize(() => {
  db.run(`CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT,
    email TEXT UNIQUE,
    password TEXT,
    phone TEXT,
    balance INTEGER DEFAULT 0,
    role TEXT DEFAULT 'user',
    referral_code TEXT
  )`);

  db.run(`CREATE TABLE IF NOT EXISTS ads_feed (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT,
    video_url TEXT,
    payout INTEGER DEFAULT 10,
    source TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
  )`);

  db.run(`CREATE TABLE IF NOT EXISTS submissions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    ad_id INTEGER,
    proof_file TEXT UNIQUE,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
  )`);

  db.run(`CREATE TABLE IF NOT EXISTS transactions (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    type TEXT,
    amount INTEGER,
    reference TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
  )`);
});

// ============================
// AUTH
// ============================

function referralCode() {
  return Math.random().toString(36).substring(2, 8).toUpperCase();
}

app.post('/register', async (req, res) => {
  const { name, email, password, phone } = req.body;
  const hash = await bcrypt.hash(password, 10);
  db.run(
    'INSERT INTO users (name,email,password,phone,referral_code) VALUES (?,?,?,?,?)',
    [name, email, hash, phone, referralCode()],
    err => err ? res.send('Account exists') : res.redirect('/login')
  );
});

app.post('/login', (req, res) => {
  const { email, password } = req.body;
  db.get('SELECT * FROM users WHERE email=?', [email], async (e, u) => {
    if (!u || !(await bcrypt.compare(password, u.password))) return res.send('Invalid login');
    req.session.userId = u.id;
    req.session.role = u.role;
    res.redirect('/dashboard');
  });
});
```

```
// ============================
// AI AD REFRESH (MOCK ENGINE)
// ============================

function aiGenerateAds() {
  const aiAds = [
    { title: 'AI Travel Inspiration – Maldives', video:
'https://cdn.coverr.co/videos/coverr-tropical-island-8553/1080p.mp4', source: 'AI' },
    { title: 'AI Fashion Trends Worldwide', video:
'https://cdn.coverr.co/videos/coverr-fashion-walk-9174/1080p.mp4', source: 'AI' },
    { title: 'AI Motivation – Success Daily', video:
'https://cdn.coverr.co/videos/coverr-success-mindset-1602/1080p.mp4', source: 'AI' }
  ];

  aiAds.forEach(ad => {
    db.run('INSERT INTO ads_feed (title,video_url,source) VALUES (?,?,?)', [ad.title,
ad.video, ad.source]);
  });
}

// Refresh ads every 24 hours (mock AI)
setInterval(aiGenerateAds, 24 * 60 * 60 * 1000);
aiGenerateAds();

// ============================
// USER ADS FEED
// ============================

app.get('/ads-feed', (req, res) => {
  if (!req.session.userId) return res.redirect('/login');
  db.all('SELECT * FROM ads_feed ORDER BY created_at DESC LIMIT 20', (e, ads) => {
    res.send(ads.map(a => `
      <h3>${a.title}</h3>
      <video width="320" controls src="${a.video_url}"></video>
      <form method="POST" action="/submit-ad-proof" enctype="multipart/form-data">
        <input type="hidden" name="ad_id" value="${a.id}" />
        <input type="file" name="proof" required />
        <button>Submit Screenshot</button>
      </form><hr/>`).join(''));
  });
});

// ============================
// ANTI-FAKE SCREENSHOT
// ============================

app.post('/submit-ad-proof', upload.single('proof'), (req, res) => {
  if (!req.session.userId) return res.send('Login required');
  if (!req.file) return res.send('Screenshot required');

  const allowed = ['image/png', 'image/jpeg'];
  if (!allowed.includes(req.file.mimetype)) return res.send('Invalid screenshot');

  db.run(
    'INSERT INTO submissions (user_id,ad_id,proof_file) VALUES (?,?,?)',
    [req.session.userId, req.body.ad_id, req.file.filename],
    err => {
      if (err) return res.send('Duplicate proof detected');
      db.run('UPDATE users SET balance = balance + 10 WHERE id=?', [req.session.userId]);
      db.run('INSERT INTO transactions (user_id,type,amount,reference) VALUES (?,?,?,?)',
        [req.session.userId, 'ad-view', 10, 'AI Ad']);
      res.send('Proof approved. KES 10 credited');
    }
  );
});

// ============================
// DASHBOARD + MARQUEE
// ============================

app.get('/dashboard', (req, res) => {
  if (!req.session.userId) return res.redirect('/login');
  const marquee = ['+254 7638***** withdrew KES 10,120', '+256 7012***** withdrew UGX
385,000', '+255 7123***** withdrew TZS 240,000'];
  res.send(`
    <marquee>${marquee.join(' • ')}</marquee>
    <h2>User Dashboard</h2>
    <a href="/ads-feed">View Ads</a><br/>
    <a href="/withdraw">Withdraw</a><br/>
    ${req.session.role === 'admin' ? '<a href="/admin">Admin Panel</a>' : ''}
  `);
});

// ============================
// WITHDRAW + EMAIL
// ============================

const transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: { user: 'earnad67@gmail.com', pass: 'APP_PASSWORD' }
});

app.post('/withdraw', (req, res) => {
  db.get('SELECT email FROM users WHERE id=?', [req.session.userId], (e, u) => {
    transporter.sendMail({
      from: 'Ad Earn <earnad67@gmail.com>',
      to: u.email,
      subject: 'Withdrawal Under Review',
      text: 'The withdrawal request is under review and funds will be received after approval.

Kind regards
Ad earn team'
    });
    res.send('Withdrawal request submitted');
  });
});

// ============================
// FULL ADMIN UI
// ============================

app.get('/admin', (req, res) => {
  if (req.session.role !== 'admin') return res.send('Access denied');
  res.send(`
    <h2>Admin Dashboard</h2>
    <ul>
      <li><a href="/admin/users">Users</a></li>
      <li><a href="/admin/submissions">Ad Proofs</a></li>
      <li><a href="/admin/transactions">Transactions</a></li>
      <li><a href="/admin/ads">Ads Feed</a></li>
    </ul>
  `);
});

app.get('/admin/users', (req, res) => {
  if (req.session.role !== 'admin') return res.send('Denied');
  db.all('SELECT id,email,balance,role FROM users', (e, rows) => res.json(rows));
});

app.get('/admin/submissions', (req, res) => {
  if (req.session.role !== 'admin') return res.send('Denied');
  db.all('SELECT * FROM submissions ORDER BY created_at DESC', (e, rows) =>
res.json(rows));
});

app.get('/admin/transactions', (req, res) => {
  if (req.session.role !== 'admin') return res.send('Denied');
  db.all('SELECT * FROM transactions ORDER BY created_at DESC', (e, rows) =>
res.json(rows));
});

app.get('/admin/ads', (req, res) => {
  if (req.session.role !== 'admin') return res.send('Denied');
  db.all('SELECT * FROM ads_feed ORDER BY created_at DESC', (e, rows) => res.json(rows));
});

// ==============================
// START SERVER
// ==============================

app.listen(3000, () => console.log('AdEarn running at http://localhost:3000'));
```