Ride-Share App: Login, Verification & Admin Dashboard Plan

Step 1: Account Creation & Verification

1. User registers with phone/email.
2. Save user in DB with status = 'pending_verification'.
3. Generate and send OTP via SMS or email.
4. User inputs OTP to verify.
5. If OTP valid, mark user as 'verified = true' and return JWT token.
6. Store token on frontend (AsyncStorage/localStorage).
7. Use token for all future API calls.

Step 2: Backend Auth API Endpoints

Rider:

- POST /api/auth/register
- POST /api/auth/verify-otp
- POST /api/auth/login

Driver:

- POST /api/auth/driver/register
- POST /api/auth/driver/verify-otp
- POST /api/auth/driver/login

Step 3: Database Tables

drivers:

- id, name, phone/email, password_hash, is_verified, created_at

riders:

- Same as drivers

otp_verifications:

- id, user_id, role, otp_code, expires_at

Ride-Share App: Login, Verification & Admin Dashboard Plan

Step 4: Login Flow

1. User enters phone/email + password/OTP.
2. Backend checks credentials and 'verified' status.
3. If valid, return JWT token and navigate to dashboard.
4. Else return 401 Unauthorized or appropriate error.

Step 5: Admin Dashboard

1. Create route: /admin-dashboard
2. Accessible only to users with role = 'admin'
3. Sections:
   - Drivers: list, status, ratings
   - Riders: list, payment history
   - Live map (optional)
   - Rides summary: earnings, top drivers
   - Safety logs & verification status

Example APIs:

- GET /api/admin/drivers
- GET /api/admin/riders
- GET /api/admin/rides
- GET /api/admin/safety-logs
- GET /api/admin/stats

Step 6: Frontend Integration

Mobile App (React Native):

- Use AuthContext for login state
- Secure token storage in AsyncStorage
- Navigate based on verification

Ride-Share App: Login, Verification & Admin Dashboard Plan

Web (React Admin):

- Route: /admin-dashboard
- Show stats, charts, user lists
- Protect routes with role-based token auth

Step 7: Developer To-Do List

- Implement OTP generation/expiry
- Create 'otp_verifications' table
- Add 'is_verified' to user tables
- Use JWT + role for secure APIs
- Build admin dashboard UI
- Add token-auth middleware
- Protect admin routes