RIDE N SHARE — 60-DAY AMBITIOUS BUILD ROADMAP

Solo build roadmap — Phong Lu + Claude Code

Start Date: April 6, 2026

Launch Target: June 4, 2026

Builder: Phong Lu (solo developer + Claude Code)

PHASE 1: FOUNDATION (Days 1–10)

Week 1: Legal + Setup (Days 1–7)

Day 1–2: Incorporate + Dev Setup

- ■ Reserve name "Ride N Share Inc." at bcregistry.gov.bc.ca ($31.50) — ■ BC Registry closed for Easter, call when open
- ■ Prepare Articles of Incorporation
- ■ File incorporation online ($350.21)
- ■ Apply for Business Number with CRA (free, online)
- ■ Set up business bank account — KOHO (no fees, 1% cashback)
- ■ ~~Install Homebrew 5.1.4~~
- ■ ~~Install Node.js v25.9.0~~
- ■ ~~Install npm 11.12.1~~
- ■ ~~Install VS Code~~
- ■ ~~Install Expo CLI 6.3.12~~
- ■ ~~Install GitHub CLI 2.89.0~~
- ■ ~~Create Expo account (sorrisolu)~~
- ■ ~~Install Expo Go on iPhone + logged in~~
- ■ ~~Create GitHub repo (private) — github.com/sorrisolu/ride-n-share~~
- ■ ~~Push existing code to GitHub~~
- ■ ~~Update logo to Kindness Keystone PNG across all pages~~
- ■ ~~Write grant inquiry letter (.docx) for Gwaii Trust~~
- ■ ~~Write 99-word project summary~~

- ■ Run existing website locally (`python3 -m http.server 3456`)
- ■ Run existing backend locally (`cd backend && npm install && npm run dev`)
- ■ Run existing app locally (`cd app && npm install && npx expo start`)

Day 2 (continued): Community Car Rental API

- ■ ~~Design rental_listings and rental_bookings database tables~~
- ■ ~~Create tiered fee structure (19%/26%/33% based on vehicle age)~~
- ■ ~~Build full rental API: list, browse, book, cancel, my-listings, my-bookings~~
- ■ ~~Overlap detection for double-booking prevention~~
- ■ ~~24-hour cancellation with full refund policy~~
- ■ ~~Guest 10% service fee calculation~~
- ■ ~~Fee comparison endpoint (vs Turo, traditional rentals)~~
- ■ ~~Test all rental endpoints end-to-end~~
- ■ ~~Update website pages with car rental feature (index, how-it-works, for-drivers)~~
- ■ ~~Update MASTER_PLAN.md with car rental section~~

Day 3: Database & Auth (moved up from Day 4–5)

Day 3–4: Database & Auth

- ■ ~~Set up PostgreSQL (free cloud: Supabase)~~
- ■ ~~Create database schema (users, rides, vehicles, payments, disputes, rentals — 10 tables)~~
- ■ Set up Redis (local + free cloud: Upstash)
- ■ ~~Wire up auth routes to real database (bcrypt + JWT)~~
- ■ ~~Test register → login → JWT flow end-to-end~~
- ■ Test on phone via Expo Go

Day 6–7: Maps & Location

- ■ Get Mapbox API key (free tier: 50,000 loads/month)
- ■ Integrate Mapbox into HomeScreen.js (replace placeholder MapView)
- ■ Get user's live GPS location
- ■ Display current location on map
- ■ Add destination search/autocomplete
- ■ Test location on physical device (GPS requires real device)

Week 2: Core Ride Flow (Days 8–14)

Day 8–9: Rider Flow

- ■ Ride request screen: pick destination → confirm → submit
- ■ Save ride request to database (status: "requested")
- ■ Real-time ride status updates (Socket.io)
- ■ Ride states: requested → accepted → driver_arriving → in_progress → completed
- ■ Show driver info after match (name, photo, vehicle, rating)
- ■ Display ETA and route on map

Day 10–11: Driver Flow

- ■ Driver mode toggle (rider ↔ driver)
- ■ Driver dashboard: go online/offline
- ■ See nearby ride requests on map
- ■ Accept ride → navigate to pickup
- ■ Start ride → navigate to destination
- ■ Complete ride → trigger payment/rating
- ■ Driver earnings tracker

Day 12–13: Ride Matching

- ■ Matching algorithm: nearest available driver wins
- ■ Redis-backed real-time driver location tracking
- ■ Push notifications (Expo Notifications) for new ride requests
- ■ Timeout: if no driver accepts in 60 seconds, notify rider
- ■ Cancel ride (rider or driver) with appropriate status updates

Day 14: Ratings & History

- ■ Post-ride rating screen (1–5 stars + optional comment)
- ■ Both rider and driver rate each other
- ■ Ride history screen (past rides with date, route, fare, rating)
- ■ Driver average rating calculation + display on profile
- ■ **END OF WEEK 2 MILESTONE:** Full car ride flow working on phone

---

PHASE 2: PAYMENTS, BOAT RIDES & CAR RENTAL UI (Days 15–28)

Week 3: Payments (Days 15–21)

Day 15–16: Stripe Integration

- ■ Connect existing Stripe account
- ■ Add `STRIPE_PUBLISHABLE_KEY` and `STRIPE_SECRET_KEY` to `.env`
- ■ Create PaymentIntent on ride completion
- ■ In-app payment screen (card input via Stripe SDK)
- ■ Stripe webhook → mark ride as "paid"
- ■ Fee calculation: 17% from driver earnings for rideshare/boats; lower of $3 or 3% for skill share
- ■ Test with Stripe test mode cards

Day 17–18: Crypto Payments

- ■ Create NOWPayments account
- ■ Integrate NOWPayments API (BTC, ETH, XRP)
- ■ Crypto payment option on payment screen
- ■ Generate payment address/QR code for rider
- ■ IPN webhook → confirm crypto payment received
- ■ Handle price volatility (lock rate for 15 minutes)

Day 19: Cash Payments

- ■ Cash option on payment screen
- ■ Driver confirms cash received (in-app button)
- ■ Platform fee still tracked (paid via weekly settlement)
- ■ Cash ride marked as "paid_cash" in database

Day 20–21: Payment Polish

- ■ Payment history screen (all payment methods)
- ■ Driver payout tracking (Stripe Connect or manual)
- ■ Receipt generation (email via SendGrid)
- ■ Fee transparency: show rider + driver fee breakdown before confirming ride
- ■ **END OF WEEK 3 MILESTONE:** All payment methods working

Week 4: Boat Rides + Scheduled Rides (Days 22–28)

Day 22–23: Boat Ride Flow

- ■ Car/Boat toggle on home screen

- ■ Captain registration flow (vessel info, license, insurance upload)
- ■ Boat ride request: select coastal points (dropdown or map)
- ■ Captain-specific matching (boats near departure point)
- ■ Boat ride pricing (distance-based or captain-set)
- ■ Weather check: flag rough conditions from Environment Canada

Day 24–25: Scheduled Rides

- ■ Schedule ride screen: date picker (1 hour to 7 days ahead)
- ■ Scheduled ride stored in database with future timestamp
- ■ Driver/captain can browse and claim scheduled rides
- ■ Push notification reminders (30 min before pickup)
- ■ Cancel/reschedule flow

Day 26: Car Rental App Screens

- ■ Rental browse screen: search by location, date, vehicle type
- ■ Rental listing detail screen: photos, rates, host info, reviews
- ■ Rental booking flow: select dates, see price breakdown, confirm
- ■ Host listing creation screen: add photos, set rates, availability
- ■ Host dashboard: my listings, incoming bookings, earnings
- ■ Renter dashboard: my bookings, active rentals, history
- ■ Wire rental screens to backend API (`/api/rentals/`)

Day 27–28: Document Upload & Verification

- ■ AWS S3 bucket setup for document storage
- ■ Driver document upload screen (photo, license, insurance)
- ■ Captain document upload screen (vessel license, marine insurance)
- ■ Admin verification queue (approve/reject documents)
- ■ Verified badge on driver/captain profiles
- ■ Block unverified drivers from going online

Day 28: Safety Features

- ■ Emergency button → calls 911 + sends SMS with GPS to emergency contact
- ■ Share ride details via SMS (link to live tracking)
- ■ In-app messaging (rider ↔ driver, Socket.io)
- ■ Report user flow
- ■ **END OF WEEK 4 MILESTONE:** Car + boat rides, car rental UI, all payments, scheduling, safety

PHASE 3: BLOCKCHAIN & SKILL SHARE PAY OPTION (Days 29–38)

Week 5: Smart Contract Deployment (Days 29–35)

Day 29–30: Smart Contract Setup

- ■ Install Hardhat (`npm install --save-dev hardhat`)
- ■ Configure Hardhat for Arbitrum One
- ■ Write deployment script for SkillShareEscrow.sol
- ■ Deploy to Arbitrum Sepolia testnet first
- ■ Test all contract functions on testnet
- ■ Verify contract on Arbiscan

Day 31–32: Backend Bridge

- ■ Connect backend to Arbitrum via ethers.js v6
- ■ `POST /api/contracts/propose` → calls `propose()` on-chain
- ■ `POST /api/contracts/accept` → calls `accept()`
- ■ `POST /api/contracts/sign` → calls `sign()` (both parties)
- ■ `POST /api/contracts/ride-done` → calls `markRideDone()`
- ■ `POST /api/contracts/skill-done` → calls `confirmSkillDone()` → auto-settles
- ■ `POST /api/contracts/dispute` → calls `openDispute()`
- ■ Test full flow on testnet

Day 33–34: Skill Share App Screens

- ■ Wire up SkillShareScreen.js to real backend
- ■ Browse tab: see available skill trade proposals
- ■ Propose tab: create new skill trade (category, description, hours, partner)
- ■ My Trades tab: active trades with status tracking
- ■ Skill categories: 12 categories with emoji (electrical, plumbing, carpentry, etc.)
- ■ Contract status display (proposed → accepted → signed → active → completed)
- ■ Dispute flow in-app

Day 35: Deploy to Mainnet

- ■ Deploy SkillShareEscrow.sol to Arbitrum One mainnet
- ■ Update `SKILL_SHARE_CONTRACT_ADDRESS` in `.env`

- ■ Fund platform wallet with small ETH for gas (~$5 worth)
- ■ End-to-end test on mainnet (propose → complete)
- ■ **END OF WEEK 5 MILESTONE:** Skill share fully working on Arbitrum mainnet

---

Week 5.5: Charity System (Days 36–38)

Day 36: Donation Website Tab

- ■ Wire up `donate.html` to real Stripe account
- ■ Donation tiers ($10 / $25 / $50 / custom amount)
- ■ Stripe PaymentIntent for direct donations
- ■ Donation confirmation + thank you page
- ■ Donation counter on website (total donated)

Day 37: Auto-Donation System

- ■ Monthly cron job calculates platform profits
- ■ Auto-calculates 10% of monthly revenue
- ■ Stripe Connect transfer to Islamic Relief's account
- ■ Admin dashboard: donation history, current recipient, rotation schedule
- ■ Transparency: public stats endpoint

Day 38: Charity Polish

- ■ Charity section on app (Impact tab)
- ■ Show current recipient, total donated, next rotation
- ■ Allow users to vote on next charity (future feature flag)
- ■ Wire CarbonTrackerScreen to real ride data
- ■ **MILESTONE:** Charity auto-donation + direct donation working

---

PHASE 4: POLISH & LANGUAGE (Days 39–48)

Week 6: Haida Language + i18n (Days 39–42)

Day 39–40: Language System

- ■ Finalize i18n system in app (react-i18next or custom)
- ■ English + French translations complete
- ■ Language switcher in app settings
- ■ Website language toggle (already built, needs real translations)
- ■ All UI strings externalized (no hardcoded text)

Day 41–42: Haida Language

- ■ Contact local Haida language speakers/knowledge keepers
- ■ Request translations for key UI strings (~50–100 phrases)
- ■ Skidegate dialect (X■aayda Kil) translations
- ■ Masset dialect (X■aad Kil) translations
- ■ Community review of translations before publishing
- ■ Note: use placeholder text if translations not ready — add later

---

Week 7: Admin Dashboard + Polish (Days 43–48)

Day 43–44: Admin Dashboard

- ■ Admin web panel (simple React or static HTML + API calls)
- ■ User management (view, approve, suspend)
- ■ Ride monitoring (active rides, completed, disputes)
- ■ Payment dashboard (revenue, fees, payouts)
- ■ Document verification queue
- ■ Dispute resolution interface
- ■ Charity donation tracker

Day 45–46: App Polish

- ■ Onboarding screens (3-slide intro for new users)
- ■ Profile editing (photo, name, phone, emergency contact)
- ■ Notification settings
- ■ Dark mode support (optional)
- ■ Loading states, error handling, empty states
- ■ Offline handling (show cached data, queue requests)

Day 47–48: Website Polish

- ■ Upload website to Bluehost

- ■ Test all pages on mobile + desktop
- ■ Wire waitlist form to real database
- ■ SEO: meta tags, Open Graph, sitemap
- ■ Google Analytics setup
- ■ **END OF WEEK 7 MILESTONE:** Everything built and polished

---

PHASE 5: TESTING & LAUNCH (Days 49–60)

Week 8: Testing (Days 49–54)

Day 49–50: Internal Testing

- ■ Test every screen on iOS (Expo Go)
- ■ Test every screen on Android (emulator or device)
- ■ Test all payment flows (Stripe test mode, crypto testnet)
- ■ Test skill share contract flow end-to-end
- ■ Test boat ride booking
- ■ Test scheduled rides
- ■ Test emergency features
- ■ Fix all critical bugs

Day 51–52: Beta Testers

- ■ Recruit 10 beta testers (friends, family, community members)
- ■ Distribute via Expo Go (no app store needed yet)
- ■ Create feedback form (Google Forms)
- ■ Monitor for crashes (Expo error reporting)
- ■ Collect and prioritize feedback

Day 53–54: Bug Fixes + Performance

- ■ Fix bugs reported by beta testers
- ■ Optimize slow screens
- ■ Test with poor internet (common on Haida Gwaii)
- ■ Reduce app bundle size
- ■ Backend load testing (simulate 50 concurrent users)

Week 9: Pre-Launch (Days 55–58)

Day 55: App Store Preparation

- ■ Apple Developer Account ($99/year)
- ■ Google Play Developer Account ($25 one-time)
- ■ App screenshots (6.5" iPhone, 5.5" iPhone, Android)
- ■ App Store description + keywords
- ■ Privacy policy page on website
- ■ Terms of service page on website

Day 56–57: Deployment

- ■ Deploy backend to production (Railway/Render/DigitalOcean)
- ■ Set up production PostgreSQL + Redis
- ■ Configure production environment variables
- ■ Set up Stripe webhooks for production URL
- ■ Deploy smart contract to Arbitrum mainnet (if not done in Day 35)
- ■ SSL certificate on backend
- ■ Submit iOS app to App Store review
- ■ Submit Android app to Google Play review

Day 58: Marketing Prep

- ■ Print 200 flyers for local distribution
- ■ Create social media accounts (Instagram, TikTok, Facebook)
- ■ Draft launch announcement posts
- ■ Contact local media (Haida Gwaii Observer, CFNR Radio)
- ■ Prepare ferry terminal signage
- ■ Email waitlist subscribers (from website)

Week 9+: SOFT LAUNCH (Days 59–60)

Day 59: Founding Drivers

- ■ Recruit 5–10 founding drivers personally
- ■ Walk each driver through signup process

■ Verify all driver documents

■ Test first real ride with a founding driver

■ Buy each driver coffee ■

Day 60: PUBLIC LAUNCH

■ Announce on social media

■ Distribute flyers at ferry terminal, Co-op, gas stations

■ Post in Haida Gwaii community Facebook groups

■ First real paid rides

■ Monitor everything (server, payments, crashes)

■ Celebrate ■

POST-LAUNCH (Days 61+)

Month 2: Iterate

- Collect user feedback daily
- Fix bugs immediately
- Add most-requested features
- Grow driver pool to 20+
- Submit grant applications with real usage data

Month 3: Expand

- Expand to Moresby Island
- Add ferry schedule integration
- Add offline mode for poor-signal areas
- Multi-stop routes
- Partner with tourism operators

Month 4–6: Scale

- Expand to other remote BC communities
- Apply for additional grants with proven traction
- Consider community ownership model
- Plan national expansion

DAILY ROUTINE (RECOMMENDED)

| Time | Activity |

|------|----------|

| 8:00 AM | Review roadmap — what's today's task? |

| 8:30–12:00 | Code (focused, no distractions) |

| 12:00–1:00 | Lunch + walk |

| 1:00–5:00 | Code + test on phone |

| 5:00–5:30 | Push to GitHub, write notes on what you built |

| 5:30–6:00 | Plan tomorrow's task |

Rule: Ship something every single day. Even if it's small. Progress > perfection.

60 days. One feature at a time. You've got this.