Project Proposal:
My Car Concierge – MVP Stabilization & Launch Readiness

Replit / Netlify

Prepared For: Jordan Zaneti

Table of Contents

Executive Summary

My Car Concierge is a customer-first marketplace platform that reverses the traditional automotive services model by allowing service providers to compete for customer jobs through a transparent bidding process. The platform enables car owners and fleet operators to list vehicles, define maintenance or custom service needs, and select providers based on competitive bids, trust indicators, and pricing.

The product is already functionally advanced and close to launch, with core user flows implemented and initial pilot providers identified. The immediate need is not new feature expansion, but **stability, quality assurance, performance reliability, and store-readiness** to support a successful public launch on iOS and Android.

This proposal focuses on delivering a **stable, production-ready MVP**, resolving known glitches, validating workflows end-to-end, and ensuring the platform can operate reliably with real users and payments.

Problem Statement

Car owners—especially rideshare drivers and fleet operators—waste significant time and money searching for trustworthy automotive service providers, comparing quotes, and managing recurring maintenance. The existing ecosystem incentivizes opaque pricing, bait-and-switch practices, and poor customer experience.

While My Car Concierge successfully addresses this problem conceptually, the current implementation faces:

- Intermittent UI and loading issues (server “spazzing” behavior)
- Inconsistent behavior across hosting environments (Replit vs Netlify)
- Limited QA coverage across real-world user journeys

- Risk of App Store rejection due to stability, performance, or edge-case failures

Without stabilization and structured QA, these issues could undermine user trust at launch.

Goals and Objectives

Primary Goals

- Deliver a stable, fully functional MVP suitable for App Store and Play Store submission
- Ensure all critical user journeys work reliably end-to-end
- Reduce operational and reputational risk at launch

Secondary Goals

- Prepare the platform for early provider onboarding and pilot usage
- Establish a technical baseline that supports future feature growth
- Improve performance, reliability, and maintainability without a full rebuild

User Journey

Member (Car Owner / Fleet User)

1. Registers and logs into the platform
2. Adds one or more vehicles (year, make, model)
3. Uses system-recommended or custom service schedules
4. Creates service listings (one-time or recurring)
5. Receives competitive bids from local providers
6. Reviews bids, chats if needed, and selects a provider
7. Pays through Stripe escrow
8. Approves or rejects additional charges within a 4-hour window

9. Confirms job completion and release of funds

Service Provider

1. Registers as a provider business
2. Purchases bid credit packages
3. Browse eligible service listings
4. Submits bids on relevant jobs
5. Wins bids and fulfills services
6. Receives payment upon job completion
7. Optionally opts into employee background check visibility

Platform (System)

- Manages escrow payments
- Enforces bidding lifecycle rules
- Sends notifications for approvals, bid wins, and changes
- Tracks referral credits and founder commissions

Solution Option

MVP Stabilization, QA, and Store Readiness

- No redesign
- No feature creep
- No architectural overhaul

A focused engagement to:

- Perform in-depth QA
- Fix identified bugs and performance issues
- Stabilize hosting and API behavior
- Prepare the app for successful App Store & Play Store submission

Timeline and Hour Estimates

Phase	(Hours)	Key Activities
Phase 1: Platform Audit & QA Planning	20	- Codebase and hosting review (Replit / Netlify) - Environment consistency checks - Critical path identification - QA checklist and test-case definition
Phase 2: In-Depth QA & Bug Identification	30	- Member portal testing - Provider portal testing - Bidding lifecycle validation - Stripe escrow flow testing - Notification & approval flows - Cross-browser and responsive testing
Phase 3: Bug Fixes & Stability Improvements	35	- UI/UX rendering issues - Loading and state-management bugs - Hosting and deployment inconsistencies - Performance optimizations - Error handling and edge cases
Phase 4: Store Readiness & Submission Support	15	- Final regression testing - Build verification - App Store & Play Store compliance checks - Submission support and fixes (if required)
Total Estimated Effort	**100**	

Technical Roadmap of the Solution

Architecture

- Existing frontend and backend retained
- Replit / Netlify hosting stabilized
- Stripe escrow integration maintained
- Modular fixes without refactor risk

Data & Payments

- Secure Stripe escrow flows
- Transaction lifecycle validation
- Bid credit accounting verification

Integrations

- Stripe (payments & escrow)
- Optional Checker background check integration (deferred / sandbox-ready)

QA & Reliability

- Functional QA
- Regression testing
- Performance and loading validation
- App Store readiness checks

Our Assumptions

- Core features shown in the demo remain unchanged
- No new major features are added during this phase
- Founder provides access to repositories, hosting, and Stripe sandbox
- Checker integration is **optional and non-blocking** for MVP
- Emergency roadside assistance (AAA-style) is out of MVP scope

Recommendations to the Client

1. **Launch with stability, not complexity**
   Early trust matters more than feature depth.

2. **Defer advanced edge cases**
   Emergency diagnostics and roadside workflows should be post-launch.

3. **Use early providers as feedback partners**
   Pilot shops will expose real-world gaps faster than speculation.

4. **Plan a Phase-2 roadmap post-launch**
   Background checks, automation, and scaling should follow traction.