From Intent to Delivery

The Vertical Horizon Platform

Upload any RFP, PRD, or statement of work. Vertical Horizon performs complete requirements analysis, structured task decomposition, calibrated estimation, and generates a swarm package AI agent teams can execute autonomously.

Kindred Table — Supper Club Marketplace

Development Proposal

---

1. Executive Summary

This proposal outlines the complete technical implementation of Kindred Table, a two-sided marketplace connecting supper club hosts with guests in Austin, Texas. The solution will deliver a fully functional web platform enabling hosts to publish ticketed dinners, guests to discover and book experiences, and platform administrators to manage operations through a comprehensive console.

The implementation encompasses all requirements specified in the Preliminary Requirements and System Specification v0.2, including host onboarding with Stripe Connect integration, dinner listing management, guest discovery and booking with real-time inventory control, payment processing with configurable fee structures, review systems, notification infrastructure, and founder administration tools.

Total Investment: $163,305
Timeline: 8 weeks (4 two-week sprints)
Deliverable: Production-ready marketplace platform with comprehensive testing and documentation

---

2. Understanding of Requirements

Kindred Table addresses the need for a trusted marketplace that facilitates intimate dining experiences while managing the complexity of:

- **Host verification and approval** through a structured application process with quality controls
- **Real-time seat inventory management** preventing overbooking under concurrent transactions
- **Money movement** through Stripe Connect with destination charges, application fees, and host payouts
- **Configurable business rules** including cancellation policies (time-based refund percentages), platform fee rates, and promotional codes
- **Trust and safety** mechanisms through double-blind reviews, host reliability tracking, and address revelation protocols
- **Operational transparency** via milestone event tracking, exportable audit logs, and reconciled financial ledgers

The platform targets demonstration deployment with specified production-grade elements (payment security via PCI-compliant Stripe Elements, webhook reconciliation, idempotent event handling) while acknowledging deferred production requirements (food safety compliance depth, sales tax automation, disaster recovery protocols, formal dispute resolution) as documented in the M0 audit findings.

---

3. Requirements Traceability Matrix

The following matrix demonstrates complete coverage of all RFP requirements across the proposed solution:

Epic	Feature	RFP Reference
Host Onboarding & Account Management	Host Application Registration	§4 REQ-H1.1: Application with profile, venue description, sample menu, photos entering APPLIED status
	Host Application Approval Workflow	§4 REQ-H1.1: Founder approval with checklist (complete profile, ≥3 photos, sample menu, Stripe onboarding, phone/video screen, safety flags)
	Stripe Connect Onboarding Integration	§4 REQ-H1.2: Required completion before publishing (ERR_PAYOUT_NOT_READY); §10 OQ-2: Express chosen for fastest onboarding
	Co-Host Support	§4 REQ-H1.3: One co-host per dinner with display + access, payout to primary host only
	Host Reliability Tracking	§4 REQ-H2.3: Host-reliability mark visible to founder for repeat cancellation review
Dinner Listing Management	Create Dinner Listing	§4 REQ-H2.1: Title, narrative, menu, dietary tags (VEGETARIAN, VEGAN, GF, DAIRY_FREE, NUT_FREE, NONE), date/time, address, seat count (2–20), price
	Listing Lifecycle State Machine	§4 REQ-H2.2: DRAFT → PUBLISHED → SOLD_OUT ⇄ PUBLISHED → COMPLETED → CLOSED with illegal transition rejection
	Host Dinner Cancellation with Refunds	§4 REQ-H2.3: Full refunds via X1, guest notifications via X2, host-reliability marking
	Host Dashboard	§4 REQ-H2.4: Per-dinner seat sales, guest list (first name + dietary tags), expected payout in cents, review summary
Guest Discovery & Browse	Dinner Discovery Interface	§5 REQ-G1.1: Search and filters (date range, price range, dietary tags, neighborhood zones); sold-out dinners visible marked SOLD_OUT
	Soonest-First Default Ordering	§5 REQ-G1.2: Default ordering soonest-first; §10 OQ-6: Advanced ranking deferred
	Neighborhood Zone Display	§5 REQ-G2.4: Exact address revealed only to CONFIRMED guests at T-24h; neighborhood zone shown publicly
Guest Booking & Reservations	Seat Booking with Inventory Control	§5 REQ-G2.1: Book 1–4 seats with overbooking prevention under concurrency (AT-4: no oversell)
	Booking Lifecycle State Machine	§5 REQ-G2.2: PENDING_PAYMENT → CONFIRMED → ATTENDED/NO_SHOW; cancellation paths; 10-minute payment expiry
	Cancellation Policy Engine	§5 REQ-G2.3: Refund rules >72h=100%, 24–72h=50%, <24h=0%, host cancellation=100%; versioned configuration; historic bookings use original terms
	Address Revelation at T-24h	§5 REQ-G2.4: Exact address to CONFIRMED guests only at T-24h; §11 AT-5: No earlier path exists
	Booking Notifications	§5 REQ-G2.5: Confirmations, reminders (T-48h, T-4h with address), cancellation notices via X2
	Guest Identity with Passwordless Auth	§10 OQ-1: Email + SMS verification, passwordless (magic link/OTP), 30-day rolling sessions
Review & Rating System	Double-Blind Review Submission	§5 REQ-G3.1: Post-COMPLETED guests review dinner (1–5 + text); hosts review guests; double-blind publish (when both submit or T+7 days)
	Host Aggregate Rating Display	§5 REQ-G3.2: Aggregate rating on listings after ≥3 reviews; founder suppression with audit trail
Payment Processing & Money Movement	Booking Fee Calculation	§6 REQ-B1.1: subtotal = seat_price_cents × qty; platform_fee = round_half_even(subtotal × fee_rate); host_net = subtotal − platform_fee; §3 C-1: Integer cents only
	Host Payout Ledger View	§6 REQ-B1.2: Per-host derived ledger (gross, fee, net, Stripe transfer reference, status) reconciling to X1 events (AT-1/AT-8); view of Stripe truth
	Refund Processing via Stripe	§6 REQ-B1.3: Policy engine computation through X1, idempotent updates by Stripe event id
	Stripe Connect Destination Charges	§7 REQ-X1.1: Connect Standard/Express onboarding, destination charges with application fees, refunds, webhook consumption (payment_intent.succeeded, charge.refunded, payout.paid, account.updated); idempotent by event id
	Webhook Reconciliation	§7 REQ-X1.3: Scheduled reconciliation against Stripe API; missed webhooks converge within 1 hour (AT-9)
	PCI Compliance via Stripe Elements	§7 REQ-X1.2: Stripe handles tax documentation (1099-K) and PCI compliance; card data never touches platform
Promotional Codes & Discounts	Promo Code Management	§6 REQ-B1.4: Flat or percentage discounts, founder-created, usage-capped; discount reduces subtotal before fee; schema: 4–16 alphanumeric, unique; 5–50% or $5–$50 flat; single use per guest; cap 1–500; expiry required; no stacking
Notification & Messaging	Notification Provider Integration	§7 REQ-X2.1: Email/SMS via provider adapter with templated messages (confirmation, reminders, cancellations, review prompts), delivery tracking, SMS opt-in/out compliance
Founder Admin Console	Host Application Review & Approval	§8 REQ-P1.1: Console host application review/approval with §4 REQ-H1.1 approval checklist
	Listing Takedown	§8 REQ-P1.1: Console listing takedown capability
	Booking Lookup & Refund Exceptions	§8 REQ-P1.1: Booking lookup with refund-exception execution; reason codes: HOST_FAULT, GUEST_HARDSHIP, SAFETY, ERROR, OTHER (requires note)
	Review Moderation	§8 REQ-P1.1: Review suppression; §5 REQ-G3.2: Suppress pending dispute with audit trail
	Promo Code Management Console	§8 REQ-P1.1: Console promo management
	Policy & Fee Configuration	§8 REQ-P1.1: Versioned policy/fee configuration
	Neighborhood Zone Configuration	§5 REQ-G1.1: Console-editable zone list (Downtown, East Austin, South Congress, Hyde Park, Mueller, Zilker/Barton Hills, North Loop, Other)
	Host Approval Checklist Configuration	§4 REQ-H1.1: Console-editable approval checklist configuration
Business Intelligence & Milestones	Milestone Event Tracking	§8 REQ-P2.1: First-class events (FIRST_HOST_ACTIVATED, FIRST_DINNER_PUBLISHED, FIRST_BOOKING_CONFIRMED, FIRST_REVENUE_PROCESSED); REQ-P2.2: Immutable facts derived from X1/booking records, independently verifiable
	Founder Dashboard Metrics	§8 REQ-P2.1: Running GMV, fee revenue, bookings, repeat-guest rate to founder dashboard
	Exportable Event Log	§8 REQ-P2.1: Exportable event log with timestamps and source references
Public Landing & Waitlist	Waitlist Capture	§8 REQ-P3.1: Public landing presenting concept, capturing waitlist signups (both sides)
	Launch Toggle	§8 REQ-P3.1: Convert to live discovery at launch toggle (console)
Audit & Compliance	Admin Action Audit Trail	§8 REQ-P4.1: Append-only recording of console actions, policy/fee changes, refund exceptions, suppressions with actor and timestamp
	Review Suppression Audit Trail	§5 REQ-G3.2: Suppression audit trail with dispute context
User Experience & Design	Responsive Web Layout System	§1.3: Responsive web (mobile-primary); no native apps v0.1
	Accessibility Compliance (WCAG 2.1 AA)	Standard web accessibility requirements
	Loading States & Skeleton Screens	Standard UX patterns for async operations
	Error State Handling	Error codes throughout requirements (ERR_PAYOUT_NOT_READY, ERR_ILLEGAL_TRANSITION, etc.)
	Empty State Designs	Standard UX patterns for null states
	Form Validation with Inline Feedback	Standard form interaction patterns
	Toast Notification System	Standard user feedback mechanism

---

4. Proposed Solution Overview

4.1 Technical Architecture

The solution implements a modern web application architecture with the following technical approach:

Frontend Layer:

- Responsive web interface optimized for mobile-first usage
- Component-based UI architecture with reusable design system
- Real-time inventory updates for booking availability
- Progressive enhancement for accessibility (WCAG 2.1 AA compliance)
- Optimistic UI updates with server reconciliation

Backend Layer:

- RESTful API design with resource-oriented endpoints
- State machine implementations for listing and booking lifecycles
- Idempotent operation handling for external service integration
- Integer-based monetary calculations (cents throughout, no floating point)
- Versioned configuration system for business rules (cancellation policies, fee rates, approval checklists)

Data Layer:

- Relational database schema with referential integrity
- Append-only audit tables for compliance tracking
- Derived view pattern for financial ledger (source of truth: Stripe)
- Optimistic locking for concurrent seat inventory operations

Integration Layer:

- Stripe Connect integration (Express accounts for fast host onboarding)
- Destination charge pattern with application fees
- Webhook consumption with idempotent event processing
- Scheduled reconciliation for webhook outage tolerance
- Email/SMS provider adapter pattern with delivery tracking

4.2 Key Technical Capabilities

Concurrency Control:

- Pessimistic locking on seat inventory prevents overselling under concurrent last-seat purchases (AT-4)
- Database-level constraints enforce seat count boundaries
- Payment timeout mechanism (10 minutes) automatically releases uncommitted inventory

Financial Integrity:

- All monetary values stored as integer cents (i64)
- Half-even rounding for fee calculations prevents cumulative drift
- Per-booking snapshot of fee rate and cancellation policy version enables historic accuracy
- Ledger view reconciles exactly to Stripe events with independent verification path

State Management:

- Explicit state machines for listings and bookings with illegal transition rejection
- Immutable milestone events derived from transactional records
- Address revelation controlled by booking state + time threshold (CONFIRMED + T-24h only)

Operational Transparency:

- Every founder action logged with actor, timestamp, reason codes
- Business milestone events (first host, first dinner, first revenue) captured as first-class facts
- Exportable event log with source references to Stripe records
- Review suppression requires dispute context and creates audit trail

---

5. Implementation Approach

5.1 Methodology

The implementation follows an agile delivery model with two-week sprints organized into functional phases. Each phase delivers vertically integrated features with complete testing coverage.

Development Principles:

- Test-driven development for business logic components
- API-first design enabling future mobile client development
- Configuration-over-code for business rules (fees, policies, zones, checklists)
- Idempotent operations for external service integration
- Defensive validation at API boundaries

Quality Gates:

- Automated test execution on every commit
- Code review requirement for all changes
- Performance testing against concurrency thresholds (AT-4: concurrent last-seat purchase)
- Integration testing against Stripe test mode
- Accessibility audit for all user-facing interfaces

5.2 Risk Mitigation

Payment Integration Risk:

- Early Stripe sandbox integration in Phase 2
- Webhook replay testing for idempotency verification
- Reconciliation logic implemented before production deployment
- Test coverage for all refund scenarios and edge cases

Concurrency Risk:

- Database transaction isolation testing
- Load testing for seat inventory contention scenarios
- Fallback error handling for lock timeout conditions

Operational Risk:

- Comprehensive admin console for founder self-service
- Exportable audit logs for independent verification
- Milestone event tracking validates platform health
- Documented runbooks for common operational tasks

---

6. Timeline and Phases

Phase 1: Infrastructure & Environment Setup

Duration: Days 1-2
Effort: 40.3 hours
Investment: $6,045

Establishes development, staging, and production environments with continuous integration/deployment pipelines, monitoring infrastructure, and database provisioning.

Phase 2: Foundation & Core Platform

Duration: Week 1-2 (Sprint 1)
Effort: 159.5 hours
Investment: $23,925

Implements foundational systems including:

- Database schema for hosts, dinners, users, bookings
- Authentication system (passwordless email/SMS OTP)
- Host application registration and APPLIED state
- Basic dinner listing creation (DRAFT state)
- Stripe Connect Express onboarding integration
- Admin console framework with host approval workflow
- Initial API endpoints and security middleware

Milestone: First host can register and submit application

Phase 3: Booking & Communication Core

Duration: Week 3-4 (Sprint 2)
Effort: 253.5 hours
Investment: $38,025

Delivers core marketplace functionality:

- Guest discovery interface with filters (date, price, dietary, zone)
- Real-time seat booking with inventory control and overbooking prevention
- Booking state machine (PENDING_PAYMENT → CONFIRMED)
- Payment processing via Stripe destination charges
- Fee calculation engine with configurable rates
- Cancellation policy engine with time-based refund logic
- Notification provider integration (confirmation, reminder, cancellation emails/SMS)
- Host dashboard showing seat sales, guest lists, expected payouts
- Webhook consumption for Stripe events with idempotent processing

Milestones:

- First dinner can be published (PUBLISHED state)
- First guest can complete booking with payment
- First revenue processing event captured

Phase 4: Reviews, Trust & Discovery Enhancement

Duration: Week 5 (Sprint 3, Part 1)
Effort: 121 hours
Investment: $18,150

Builds trust and safety mechanisms:

- Double-blind review system (guest reviews dinner, host reviews guest)
- Review publication logic (both submitted or T+7 days)
- Host aggregate rating calculation and display (≥3 reviews threshold)
- Review suppression with audit trail
- Address revelation at T-24h for CONFIRMED bookings only
- Host reliability tracking for cancellation patterns
- Enhanced discovery with sold-out dinner visibility
- Neighborhood zone configuration in admin console

Milestone: First reviews published double-blind

Phase 5: Admin Tools & Platform Management

Duration: Week 6 (Sprint 3, Part 2)
Effort: 171 hours
Investment: $25,650

Completes operational tooling:

- Host approval checklist configuration
- Listing takedown capability
- Booking lookup with refund exception processing (reason-coded)
- Promotional code management (creation, validation, usage tracking)
- Policy and fee configuration with versioning
- Payout ledger view reconciling to Stripe events
- Business intelligence dashboard (GMV, fee revenue, bookings, repeat rate)
- Exportable event log with timestamps and source references
- Audit trail for all admin actions

Milestone: Founder can operate platform independently

Phase 6: Advanced Features & Optimization

Duration: Week 7 (Sprint 4, Part 1)
Effort: 101.5 hours
Investment: $15,225

Refines user experience and operational efficiency:

- Co-host support (display and access, payout to primary)
- Promotional code application in booking flow
- Enhanced host dashboard with review summaries
- Webhook reconciliation scheduler (missed webhook convergence <1 hour)
- Public landing page with waitlist capture (both sides)
- Launch toggle for waitlist-to-live conversion
- Loading states, skeleton screens, empty states
- Toast notification system
- Form validation with inline feedback

Milestone: Platform UI polished for demonstration launch

Phase 7: Quality Assurance & Testing

Duration: Week 7 (Sprint 4, Part 2)
Effort: 161.3 hours
Investment: $24,195

Comprehensive testing across all requirements:

- Automated test suite execution (unit, integration, end-to-end)
- Concurrency testing (AT-4: no oversell on last-seat contention)
- Payment flow testing (all refund scenarios, webhook replay)
- State machine validation (illegal transition rejection)
- Address revelation verification (AT-5: no early access path)
- Financial reconciliation testing (AT-1/AT-8: ledger matches Stripe)
- Accessibility audit (WCAG 2.1 AA)
- Performance testing against specification thresholds
- Security review (PCI compliance via Stripe Elements, auth boundaries)

Phase 8: Training & Documentation

Duration: Week 8, Days 1-2
Effort: 40.3 hours
Investment: $6,045

Operational enablement:

- Founder console usage guide
- Host onboarding documentation
- Guest user guides
- Admin runbooks (refund processing, review moderation, policy updates)
- API documentation for future integrations
- Database schema documentation
- Incident response procedures

Phase 9: Production Readiness & Deployment

Duration: Week 8, Days 3-5
Effort: 40.3 hours
Investment: $6,045

Production deployment and stabilization:

- Production environment configuration
- Stripe production account connection and validation
- Monitoring and alerting configuration
- Backup and recovery verification
- Production deployment execution
- Post-deployment smoke testing
- Handoff and knowledge transfer

Final Milestone: Platform live in production with founder-approved first host

---

7. Investment Summary

Phase	Effort (Hours)	Investment
Infrastructure & Environment Setup	40.3	$6,045
Foundation & Core Platform	159.5	$23,925
Booking & Communication Core	253.5	$38,025
Reviews, Trust & Discovery Enhancement	121.0	$18,150
Admin Tools & Platform Management	171.0	$25,650
Advanced Features & Optimization	101.5	$15,225
Quality Assurance & Testing	161.3	$24,195
Training & Documentation	40.3	$6,045
Production Readiness & Deployment	40.3	$6,045
Total	**1,088.7**	**$163,305**

Payment Milestones

- **25% ($40,826.25)** — Upon contract execution and Phase 1 completion
- **25% ($40,826.25)** — Completion of Phase 2 (Foundation & Core Platform)
- **25% ($40,826.25)** — Completion of Phase 3 (Booking & Communication Core)
- **25% ($40,826.25)** — Final acceptance after Phase 9 (Production deployment)

All pricing is fixed for the defined scope. Change requests will be evaluated and priced separately.

---

8. Proposed Team Structure

The implementation will be executed by a dedicated team:

- Product Manager (1)
- Tech Lead / Solution Architect (1)
- Senior Backend Engineers (2)
- Senior Frontend Engineer (1)
- Full-Stack Engineer (1)
- QA Engineer (1)
- DevOps Engineer (1)
- UX/UI Designer (1)

---

9. Quality Assurance Process

9.1 Testing Strategy

Unit Testing:

- Business logic components (fee calculation, policy engine, state machines)
- Data validation and constraint enforcement
- Utility functions and helpers
- Target: >90% code coverage on business logic

Integration Testing:

- API endpoint contracts
- Database transaction integrity
- Stripe integration flows (onboarding, payments, refunds, webhooks)
- Notification delivery
- State machine transitions with database persistence

End-to-End Testing:

- Complete user journeys (host onboarding → dinner creation → guest booking → completion → review)
- Cancellation flows (guest-initiated and host-initiated)
- Refund processing with policy variations
- Admin workflows (approval, takedown, refund exceptions)
- Concurrent booking scenarios

Performance Testing:

- Concurrent last-seat purchase (AT-4 validation)
- Database query optimization under load
- API response time thresholds
- Webhook processing latency

Security Testing:

- Authentication boundary enforcement
- Authorization checks (host can only edit own dinners, guests see appropriate data)
- PCI compliance verification (card data never touches server)
- SQL injection prevention
- CSRF protection
- Address revelation access control (AT-5 validation)

Accessibility Testing:

- WCAG 2.1 AA compliance verification
- Keyboard navigation
- Screen reader compatibility
- Color contrast validation
- Form label association

9.2 Acceptance Testing

Each phase includes acceptance criteria mapped to RFP requirements. Acceptance gates include:

1. **Functional Completeness:** All features in phase deliver specified behavior
2. **Test Coverage:** Automated tests pass with coverage thresholds met
3. **Integration Validation:** External services (Stripe, notifications) function correctly
4. **Performance Validation:** Meets concurrency and latency requirements
5. **Security Review:** No critical vulnerabilities, authorization correctly enforced
6. **Accessibility Audit:** WCAG 2.1 AA compliance verified

The client will have access to staging environment for validation before each payment milestone.

9.3 Defect Management

- Critical defects (data loss, payment failures, security vulnerabilities): 4-hour response, immediate remediation
- High-priority defects (broken user flows, incorrect calculations): 24-hour response, resolution within sprint
- Medium/low-priority defects: Triaged and scheduled in subsequent sprints

---

10. Next Steps

Immediate Actions

1. **Contract Execution:** Review and execute development agreement with payment milestone schedule
2. **Kickoff Meeting:** Schedule project kickoff (week of [DATE]) to review implementation plan, establish communication protocols, and confirm access requirements
3. **Access Provisioning:** Provide access to any existing resources (domain registration, hosting accounts, Stripe account)
4. **Stakeholder Identification:** Confirm founder availability for approval workflow reviews and acceptance testing

Week 1 Deliverables

- Development, staging, and production environments provisioned
- CI/CD pipeline operational
- Initial database schema deployed
- Project tracking dashboard accessible to client
- Sprint 1 planning session completed

Communication Plan

- **Daily:** Asynchronous updates via project tracking system
- **Weekly:** Sprint planning and review meetings (60 minutes)
- **Bi-weekly:** Stakeholder demo of completed features (30 minutes)
- **Ad-hoc:** Slack channel for urgent issues and questions

Decision Points

The following decisions will be required during implementation:

- **Week 1:** Email/SMS provider selection (OQ-3: options will be presented with cost/feature trade-offs)
- **Week 2:** Domain configuration and SSL certificate provisioning
- **Week 3:** Review and approve notification message templates
- **Week 5:** Review and approve initial promotional code offer for launch
- **Week 7:** Approve production deployment schedule

Appendix A: Definitions and Acronyms

- **AT-X:** Acceptance Test requirement number X
- **C-X:** Constraint number X
- **GMV:** Gross Merchandise Value (total of all bookings before fees)
- **OQ-X:** Open Question number X (resolved during M0 audit)
- **PCI:** Payment Card Industry (data security standard)
- **REQ-XX.X:** Requirement identifier from specification
- **T-Xh:** Time threshold (e.g., T-24h = 24 hours before dinner start)
- **WCAG:** Web Content Accessibility Guidelines
- **X1:** External seam 1 (Stripe integration)
- **X2:** External seam 2 (Notification provider)

Appendix B: Assumptions

This proposal assumes:

1. Kindred Table will provide Stripe account (or authorize creation of new Stripe Connect platform account)
2. Domain name will be provided or registered during Phase 1
3. Founder availability for host approval workflow testing and admin console validation
4. No integration with existing systems (greenfield implementation)
5. Demonstration deployment scope as defined in RFP (production-grade findings DEMO-WAIVED: food safety depth, sales tax automation, disaster recovery protocols, formal dispute resolution process)
6. Standard business hours support during implementation (extended support post-launch can be scoped separately)

Proposal Valid Through: [DATE + 30 days]

We look forward to partnering with you to bring Kindred Table to life. Please contact us with any questions or to schedule a proposal review meeting.

© 2026 Vertical Horizon. All Rights Reserved.