DELIVERABLE 02 · TECHNICAL EVALUATION

# Technical & Project Evaluation

Technical architecture, integration landscape, module complexity ratings, and 63–87 developer-month effort estimation.

| PREPARED FOR | SPONSOR | BY | VERSION |
|---|---|---|---|
| Resortments | Russ · Owner | Webaroo | v3.0 |

This document is prepared to serve as the technical blueprint for the Resortments Unified Enterprise Platform — translating the audit findings into architectural decisions, integration strategy, complexity ratings, and effort estimates. Our technical approach rests on four core imperatives:

1. **Integration-first architecture.** Clean connectors to QuickBooks, DocuSign, ADP, and RealPage are built into the foundation layer, not bolted on later. Every division module reads from and writes to the same data model.
2. **One source of truth across 8 divisions.** A unified PostgreSQL schema eliminates data silos and enforces referential integrity from Land Division through Accounting Hub.
3. **Mobile-first for field operations.** Zac's Plant teams and Gerardo's Field teams get native iOS/Android apps with offline capability — not a responsive web fallback.
4. **Security and compliance enforced at the platform level.** 5-tier RBAC, 7–10 year data retention, OSHA compliance tracking, and audit logging are framework-level guarantees, not per-module implementations.

These imperatives determine the technology stack, the module sequencing, and the 63–87 developer-month effort estimate that follows.

## Executive Summary

Resortments currently operates 8 independent divisions (Legal Close Land, Material Procurement, Plant Division, Field Division, Management/Leasing, Legal Rollovers, Land Division, Accounting Hub) with zero integrated technology infrastructure. Operations are managed through phone calls, email, and spreadsheets, resulting in data silos, manual handoffs, and lack of operational visibility for leadership.

**Recommendation: Build a unified enterprise platform** with modular division-specific workflows, shared authentication, centralized data management, and role-based access control. This single platform approach—rather than 8 separate applications—will eliminate redundancy, ensure data consistency, reduce integration overhead, and provide Russ with real-time operational visibility across all divisions.

**Scope:** Web and mobile-responsive interfaces with native mobile apps for field teams and management.
**Timeline:** 14 months from Phase 1 kickoff to full deployment, plus post-launch optimization.

# Architecture Evaluation

### Proposed Unified Platform Architecture

The recommended architecture consists of:

1. **Single Web Application** (React/Next.js) - Responsive design for desktop and tablet users - Role-based dashboard with division-specific views - Workflow management, document management, reporting
2. **Native Mobile Apps** (React Native) - iOS and Android with shared codebase - Field operations (material tracking, work orders) - Management access (approvals, dashboards, notifications)
3. **Shared Backend Services** (Node.js or Python) - Unified REST/GraphQL API layer - Business logic separated by domain (divisions/modules) - Scheduled jobs for integrations and data processing
4. **Centralized Database** (PostgreSQL) - Single source of truth for all divisions - Audit logging for compliance and traceability - Multi-tenancy support at data level for future scaling
5. **Shared Authentication & Authorization** (Auth0 or Clerk) - Single sign-on across all modules - Role-based access control (5 tiers) - 2-factor authentication for security
6. **Integration Layer** - API connectors for external systems - Data sync engines (QuickBooks, ADP, DocuSign, etc.) - Webhook handlers for real-time event processing

### Why One Platform Beats 8 Separate Apps

| CRITERION | 8 SEPARATE APPS | UNIFIED PLATFORM |
|---|---|---|
| **Data Consistency** | Data silos, manual sync, conflicts | Single source of truth, real-time data flow |
| **User Experience** | Switch between tools, context loss | Seamless navigation, unified dashboards, cross-division visibility |
| **Integration Complexity** | Integrate 8 apps × 6 external systems = 48 paths | 1 app × 6 external systems = 6 paths, shared connectors |
| **Maintenance Burden** | 8 codebases, 8 deployment pipelines | 1 codebase, unified DevOps, consistent standards |
| **Scalability** | Add app = replicate entire stack | Add module = extend existing platform, reuse infrastructure |
| **Total Cost** | Higher: redundant hosting, licensing, support | Lower: shared infrastructure, single deployment pipeline |
| **Operational Visibility** | Blind spots between divisions | Real-time owner dashboard, cross-division reporting |
| **Security** | 8 separate auth systems = fragmented control | Centralized auth, unified audit logging, consistent policies |

# Tech Stack Recommendation

### Frontend

**Recommended: React/Next.js**

- **Why:** Industry-standard for enterprise web applications with large feature sets

- **Advantages:**
- Vast ecosystem of libraries and UI frameworks (Material-UI, Chakra, etc.)
- Server-side rendering with Next.js for SEO and performance
- Excellent TypeScript support for type safety
- Strong community and abundant talent pool
- Suitable for complex dashboard and workflow UIs
- **Justification:** Resortments requires complex role-based views, multi-step workflows, document management, and real-time dashboards. React/Next.js excels in these areas and provides the flexibility needed as requirements evolve.

### Mobile

**Recommended: React Native**

- **Why:** Shared JavaScript codebase between iOS and Android, faster development cycles
- **Advantages:**
- Write once, run on iOS and Android (70–80% code sharing typical)
- Faster development and iteration compared to native iOS/Android
- Large developer community and library ecosystem
- Hot reloading for rapid development
- **Alternatives Considered:**
- Native iOS/Android: Higher cost, longer timeline, duplicate codebases
- Flutter: Excellent but smaller ecosystem; React Native better for enterprise integrations
- **Justification:** Field teams (Zac's Plant Division, Gerardo's Field Division) and managers need mobile access. React Native allows us to build both iOS and Android apps efficiently without doubling the development effort.

### Backend

**Recommended: Node.js (Express/NestJS) OR Python (Django/FastAPI)**

- **Node.js (Express or NestJS)**
- Advantages: Shared JavaScript with frontend team, excellent async support, proven in enterprise (Netflix, PayPal, Walmart)
- NestJS recommended for larger team or stricter architecture needs (TypeScript-first, opinionated structure)
- Ideal for real-time features (WebSockets for live notifications)
- **Python (FastAPI or Django)**
- Advantages: Rapid development, excellent data science libraries (if analytics needed), strong type hints with Pydantic
- FastAPI recommended for modern async/await, auto-generated API docs
- Larger talent pool for backend roles
- **Recommendation: Node.js with NestJS** for consistency (JavaScript across stack) and real-time capabilities. If team has stronger Python experience, **FastAPI** is a solid alternative.

### Database

**Recommended: PostgreSQL**

- **Why:** Enterprise-grade, open-source, widely adopted relational database
- **Advantages:**

- ACID compliance for financial/audit data (critical for Accounting Hub)
- Strong JSON/JSONB support for flexible document storage
- Excellent full-text search for document management
- Audit logging and triggers for compliance
- Mature and stable, **30+ years of production use**
- Excellent free tooling (pgAdmin, DBeaver)
- Cost-effective compared to proprietary databases
- **Alternatives Considered:**
- MySQL: Simpler but fewer advanced features; PostgreSQL is stricter and safer for regulated data
- NoSQL (MongoDB): Not recommended for this use case; we need ACID guarantees and relational integrity across divisions
- **Justification:** Resortments handles financial transactions, legal documents, and multi-year data retention. PostgreSQL's reliability, ACID guarantees, and strong audit features make it the clear choice.

### Authentication & Authorization

**Recommended: Auth0 or Clerk**

- **Why:** Enterprise-grade identity management as a service, eliminates building custom auth
- **Key Features:**
- Multi-factor authentication (2FA) out of the box
- Single sign-on (SSO) for web and mobile
- Role-based access control (RBAC) with custom attributes
- Audit logging of all authentication events
- OAuth/OpenID Connect for third-party integrations
- **Alternatives Considered:**
- Okta: Enterprise-focused, higher cost, overkill for current scale
- Custom-built auth: Risk of security vulnerabilities, high maintenance burden
- **Recommendation: Auth0** for its balance of features, pricing, and ease of integration. **Clerk** is a newer, developer-friendly alternative with excellent mobile support.
- **Justification:** Resortments requires 5 access tiers (Owner, Division Owner, Staff, Accounting, External) with granular permissions. An identity platform handles this complexity safely without custom code.

### Hosting & Infrastructure

**Recommended: AWS, Azure, or GCP (Cloud-Native Deployment)**

- **AWS (Recommended)**
- Advantages: Largest ecosystem, EC2/ECS for app hosting, RDS for PostgreSQL, S3 for document storage, Auth0 (or AWS Cognito as alternative) integration, proven at scale
- Services: EC2/ECS (compute), RDS (database), S3 (files), CloudFront (CDN), Lambda (serverless jobs)
- **Azure**
- Advantages: Great if Resortments already uses Microsoft stack, strong enterprise support
- Services: App Service, Azure SQL, Blob Storage, Azure AD (identity)
- **GCP**
- Advantages: Strong data analytics and machine learning, good pricing for compute
- Services: Compute Engine/App Engine, Cloud SQL, Cloud Storage, Firebase (real-time)
- **Recommendation: AWS** as primary choice for breadth of services and proven track record with enterprise clients.

- **Infrastructure as Code:** Use Terraform or CloudFormation to version-control infrastructure and ensure reproducibility.
- **Container Strategy:** Containerize backend with Docker, deploy to ECS or EKS for scalability.
- **CDN:** Use CloudFront for web app and document delivery (fast load times for field staff on slow connections).

### Additional Services

| SERVICE | RECOMMENDATION | PURPOSE |
|---|---|---|
| **API Gateway** | AWS API Gateway or Kong | Rate limiting, request routing, documentation |
| **Message Queue** | AWS SQS or RabbitMQ | Asynchronous job processing (document generation, integrations) |
| **File Storage** | AWS S3 with CloudFront | Scalable document and image storage with access control |
| **Logging & Monitoring** | CloudWatch, DataDog, or New Relic | Real-time visibility into system health and errors |
| **CI/CD** | GitHub Actions, GitLab CI, or Jenkins | Automated testing and deployment pipelines |

## Integration Feasibility Analysis

### 1. QuickBooks API

**Feasibility Rating: HIGH**

- **API Availability:** QuickBooks Online has comprehensive REST API and well-maintained SDKs
- **Integration Scope:**
- Chart of Accounts sync
- Invoice/Bill creation (construction/procurement workflows)
- Purchase Order (PO) reconciliation (~2000+ POs annually)
- Expense tracking
- Financial reporting
- **Implementation Effort:** Medium (3–4 weeks)
- **Key Considerations:**
- OAuth 2.0 authentication (standard)
- Rate limiting: **500 requests per minute per company (realm), with a 10 concurrent-request cap per app and 40 req/min for batch operations**
- Webhook support for real-time sync
- **Risks:** Low. Well-documented API, many integration examples available.
- **Recommendation:** Integrate with nightly sync + real-time webhook handlers for critical transactions.

### 2. ADP Workforce Now API

**Feasibility Rating: MEDIUM-HIGH**

- **API Availability:** ADP provides Workforce Now APIs for payroll and HR data

- **Integration Scope:**
- Employee master data (sync from ADP to dashboard)
- Payroll data (review for cost tracking across projects)
- Time tracking integration (field staff clocking in/out)
- Benefits and deductions (informational for accounting)
- **Implementation Effort:** Medium (3–4 weeks)
- **Key Considerations:**
- ADP APIs require OAuth and have detailed permission scopes
- Payroll data is sensitive; strong audit logging required
- Rate limiting is generous (sufficient for hourly syncs)
- **Risks:** Medium. ADP API documentation is good but can be complex; early testing with ADP's sandbox environment critical.
- **Recommendation:** Start with read-only employee and payroll data sync. Time tracking can be custom-built and sync to ADP later.

### 3. DocuSign API

**Feasibility Rating: HIGH**

- **API Availability:** Excellent REST API with comprehensive eSignature, envelope, and template APIs
- **Integration Scope:**
- Digital signatures for legal documents (Land Division closings, Material Procurement contracts)
- Envelope tracking (status of signed documents)
- Template management (pre-built document workflows)
- **Volume Consideration:** ~20 active closings × 500 documents per closing = ~10,000 documents/year (approximately 400–600 DocuSign envelopes/year, since envelopes bundle multiple documents)
- **Implementation Effort:** Medium (3–4 weeks)
- **Key Considerations:**
- Envelope status tracking via webhooks (real-time updates)
- Audit trail stored in DocuSign and synced to platform
- Sender/recipient role mapping to Resortments users
- Document versioning and status workflow
- **Risks:** Low-Medium. DocuSign is mature, but high document volume requires robust error handling and retry logic.
- **Volume Scaling:** Current estimate (10,000 docs/year, ~400–600 envelopes/year) is manageable. At 50,000+ docs/year, may need optimizations (batch processing, webhook optimization).
- **Recommendation:** Implement DocuSign SDK with envelope webhooks; store envelope metadata in PostgreSQL for dashboard and audit tracking.

### 4. RealPage by Thrive (Property Management Software)

**Feasibility: MEDIUM. Risk: CRITICAL (highest risk in package — see Deliverable 05).**

- **API Availability: RealPage exposes REST APIs via the RPX (RealPage Exchange) AppPartner program**; access requires vendor certification (4–8 week process). Confirm Resortments' RealPage contract tier permits RPX partner provisioning; if not, fallback is daily data export with scheduled parsing (adds 2–3 weeks to integration timeline).
- **Integration Scope:**
- Tenant data (for Management/Leasing Division)

- Lease information
- Payment and collection tracking
- Occupancy data
- **Implementation Effort:** High (6–8 weeks) due to uncertain API availability
- **Key Considerations:**
- RealPage's API limitations mean we may need:
  - **Data Export Approach:** Scheduled daily/weekly exports from RealPage to CSV/API, with custom parser
  - **Screen Scraping (Not Recommended):** Fragile and violates RealPage terms of service
  - **Direct Integration (If API Available):** Negotiate directly with RealPage for API access token
- Requires liaison with Dylan (Management/Leasing) and RealPage support team early
- **Risks:** HIGH. API availability is uncertain; integration approach may need to be data export rather than real-time API.
- **Recommendation:**
- **Action Item:** Contact RealPage to determine if API access is available for Resortments' account
- **Fallback Plan:** If no API, implement daily CSV export → parse → sync to Resortments platform
- **Timeline Impact:** If no API, add 2–3 weeks to integration timeline

### 5. Compsys

**Feasibility Rating: LOW-MEDIUM**

- **API Availability:** Limited public API documentation; primarily a legacy system with limited integration options
- **Integration Scope:** Utility and service cost tracking (if used)
- **Implementation Effort:** High (4–6 weeks) due to limited API, may require vendor collaboration
- **Key Considerations:**
- May only support SFTP file exchanges or proprietary integration format
- Requires early investigation with Compsys vendor
- Manual data entry may be more practical than complex integration
- **Risks:** High. Limited API, long vendor response times typical for legacy systems.
- **Recommendation:**
- **Research First:** Confirm Compsys integration requirements early
- **Cost-Benefit:** Evaluate if manual entry into Resortments is more efficient than tight integration
- **Timeline:** Defer to later phase if it becomes a blocker

### 6. City & State Permitting Portals

**Feasibility Rating: LOW**

- **API Availability:** None. City and state portals are non-standardized, fragmented systems with no public APIs.
- **Integration Scope:**
- Track permit application status across 50+ regulatory bodies
- Document submission tracking
- Deadline monitoring
- **Implementation Effort:** Very High (8–12 weeks) due to manual portal navigation across 50+ systems
- **Key Considerations:**

- Each city/state portal has unique UI, login, and submission process
- No practical way to fully automate without vendor collaboration (unlikely)
- RPA (Robotic Process Automation) could automate some portals but is fragile and high-maintenance
- **Risks:** Very High. Portals change frequently; automation breaks easily. Manual updates are more reliable.
- **Recommended Approach:**
- **Permitting Module in Resortments:** Create a "Permits" module where staff manually track application status
- **Link Aggregation:** Provide clickable links to each portal for easy access
- **Status Updates:** Field/Legal staff update status in Resortments after checking portal (weekly or event-based)
- **Alerts:** Resortments calculates deadline warnings based on manual entry
- **Future Improvement:** As Resortments scales, investigate if RPA for top 5-10 frequently-used portals is ROI-positive.

### 7. Design Tools (Autodesk, Bluebeam, Rhino, Lumion)

**Feasibility Rating: LOW** (Not Recommended for Integration)

- **Integration Approach:** File linking, not deep integration
- **Integration Scope:**
- Link to design files in Resortments dashboard
- Embed design renderings in project cards
- Version tracking of design assets
- **Implementation Effort:** Low (1–2 weeks)
- **Key Considerations:**
- These are highly specialized tools; Resortments should not attempt to replicate functionality
- Resortments acts as a document/project hub, linking to external tools
- File permissions and access must be managed outside Resortments
- **Recommended Approach:**
- Store file URLs/S3 links in Resortments database
- Embed design images/renderings in project dashboard
- Provide buttons to open design files in native applications
- Do NOT build design viewing or editing into Resortments
- **Justification:** Design teams will use Autodesk/Rhino/Lumion; Resortments is a coordination platform, not a replacement.

### 8. Hello Data (Data Analytics/BI Platform)

**Feasibility Rating: MEDIUM**

- **API Availability:** Hello Data offers APIs but documentation and availability are vendor-dependent
- **Integration Scope:**
- Push operational metrics to Hello Data for custom dashboards
- Real-time KPI feeds (project status, budget burn, team workload)
- **Implementation Effort:** Medium (3–4 weeks)
- **Key Considerations:**
- Requires early investigation of Hello Data's API and webhook capabilities
- May prefer Resortments to export data vs. pull integration

- **Risks:** Medium. Dependent on vendor response and API availability.
- **Recommendation:**
- **Research First:** Confirm Hello Data's current API offerings
- **Alternative:** Resortments can build custom reports/dashboards; Hello Data integration is a future enhancement if needed
- **Timeline:** Defer to Phase 2 (post-MVP)

### Integration Priority & Phasing

| PRIORITY | INTEGRATION | PHASE | TIMELINE |
|---|---|---|---|
| **P0 (MVP)** | Auth, QuickBooks, DocuSign | Foundation/MVP | Weeks 1-4 |
| **P1 (Early)** | ADP (payroll), RealPage (if API available) | Foundation/MVP | Weeks 3-5 |
| **P2 (Mid)** | Permitting Module (manual), Compsys (research) | Division Modules | Weeks 6-12 |
| **P3 (Later)** | Hello Data, Compsys (if ROI clear) | Advanced/Post-MVP | Weeks 13+ |

## Scalability Assessment

### Current Projected Scale

- **Users:** ~20–30 concurrent users initially (division heads, operational staff, accounting team)
- **Growth:** Scale to 50+ concurrent users as properties add staff and tenant portals are added
- **Divisions:** 8 divisions, each with 2-6 staff
- **Data Volume:**
- **Purchase Orders:** 2,000+ annually
- **Land Closings:** 20+ active closings with 500 docs each = 10,000+ documents
- **Property Tenants:** 100+ (across multiple properties)
- **Permits:** 50+ regulatory bodies, 200+ active permits
- **Employees:** 30-50 (via ADP)

### Database Scalability (PostgreSQL)

| METRIC | CAPACITY | ASSESSMENT |
|---|---|---|
| **Row Count** | 50+ million rows (before optimization needed) | Current data volume: ~500K rows; well within bounds |
| **Document Storage** | 50+ TB (before archive strategy needed) | Current: ~100 GB estimated; 10 years @ 10 GB/year = manageable |
| **Concurrent Connections** | Standard RDS instances (db.t3.large class) support several hundred connections via the AWS formula, but in production we pool via **PgBouncer or RDS Proxy** to keep active connections well below the memory-pressure ceiling | Current: 20-30 users, ~100 connections; comfortable headroom |
| **Query Performance** | Sub-second (with proper indexing) | Audit logging, financial reports may need query optimization |
| **Backup & Recovery** | Automated (RDS best practices) | Daily backups, 7-10 year retention requires tiered storage strategy |

**Scaling Strategy:** - **Vertical Scaling:** Start on AWS RDS db.t3.large (2 vCPU, 8 GB RAM), scale to db.r5.xlarge if database CPU exceeds 70% or p95 query latency exceeds 100 ms - **Horizontal Scaling:** Read replicas for reporting (separate from transactional DB) - **Archive Strategy:** Move documents older than 1-2 years to S3 Glacier for cost-effective 7-10 year retention

### Application Scalability (Web & Mobile)

| COMPONENT | CURRENT LOAD | SCALING APPROACH |
|---|---|---|
| **Web App (React)** | 20-30 concurrent users | CDN (CloudFront) for static assets; load testing at 50+ users |
| **API (Node.js/Python)** | ~100 requests/second peak | Auto-scaling group (EC2/ECS) with load balancer; horizontal scaling to 5-10 instances |
| **Mobile Apps** | Real-time field updates from 5-10 field staff | WebSocket connections for live updates; sync queue for offline scenarios |
| **Background Jobs** | Integration syncs (hourly QuickBooks, ADP) | Queue-based jobs (AWS SQS/RabbitMQ); scale workers based on queue depth |

**Monitoring Targets:** - Database query performance (aim for <100ms p95) - API response time (aim for <200ms p95) - Web app bundle size (aim for **<800 KB gzipped initial bundle, with route-level code splitting**) - Mobile app startup time (aim for <3 seconds)

### Data Retention & Compliance

- **Retention Period:** 7-10 years (for tax/legal audit requirements)
- **Scalability Strategy:**
- **Hot Storage (0-1 year):** PostgreSQL + S3
- **Warm Storage (1-7 years):** S3 Standard-IA (Infrequent Access)

- **Cold Storage (7-10 years):** S3 Glacier or Glacier Deep Archive
- **Deletion:** Auto-delete after 10 years unless longer retention required

### Field Operations Scalability

- **Offline-First Mobile:** Mobile app caches data locally; syncs when reconnected (critical for field teams in remote locations)
- **Geolocation Tracking:** If needed for field staff, use lightweight GPS logging with batch upload
- **Real-Time Notifications:** WebSocket connections for push notifications (approvals, task assignments)

---

## Security & Compliance

### Role-Based Access Control (RBAC) - 5 Tiers

| TIER | PERSONAS | PERMISSIONS | MODULES |
|---|---|---|---|
| **Owner** | Russ | Full system access, all divisions, financial/strategic data, user management | All 8 divisions, Owner Dashboard |
| **Division Owner** | Division heads (David, Tracy, Zac, Gerardo, Dylan, Johnny, Teresa) | Full access to own division, limited view of other divisions, approval authority | Own division module, cross-division reports (read-only) |
| **Staff** | Project managers, coordinators, field staff | Create/edit tasks within own division, view assigned items, limited financial data | Own division (create/edit), cross-division (read) |
| **Accounting** | Johnny, Teresa, accounting team | Full financial visibility (all POs, expenses, budgets), reporting, audit trails | Accounting Hub, Financial Dashboard (all divisions) |
| **External** | Third-party vendors, contractors, portal users | Limited read access, may sign documents, limited data (documents assigned to them) | Document portal, task assignments, read-only reports |

**Implementation:** Auth0/Clerk with custom attribute mapping (division_id, role_tier, approval_authority).

### Authentication & Security Measures

- **2-Factor Authentication (2FA):** Mandatory for Owner, Division Owners, Accounting. Optional for Staff.
- **Session Management:** 8-hour idle timeout for web, 24-hour timeout for mobile (with re-auth for sensitive actions)
- **Password Policy:** Minimum 12 characters, complexity requirements, no reuse of last 5 passwords
- **Encryption in Transit:** TLS 1.3 for all API communication; HTTPS for web and mobile apps
- **Encryption at Rest:** PostgreSQL encryption at table level; S3 server-side encryption (AES-256)

### Audit Logging

**All sensitive actions logged:** - User login/logout events - Document access and download - Financial transaction creation/modification - Approval workflows (who approved, when, comment) - Data exports - System configuration changes - Permission escalations

**Audit Log Storage:** - Immutable PostgreSQL audit table (no delete capability) - 7-10 year retention (archive to S3 Glacier after 2 years) - Searchable by user, action type, resource, timestamp - Exportable for regulatory compliance reviews

### Compliance & Standards

- **Data Privacy:** GDPR-compatible (if any EU users), with user data deletion capability (full record purge after compliance period)
- **Financial Compliance:** SOX-equivalent controls (audit trails, segregation of duties, approval workflows)
- **Industry Standards:**
- PCI DSS (if credit card processing; **minimizing PCI scope via Stripe/Square (SAQ-A compliance)**)
- OSHA compliance (field safety data, incident tracking)
- Legal/Regulatory (document retention, chain of custody for signatures)

### Data Backup & Disaster Recovery

- **Backup Frequency:** Daily automated backups; point-in-time restore to any time within last 30 days
- **Backup Location:** Multi-region (AWS: us-east-1 + us-west-2) for redundancy
- **Recovery Time Objective (RTO):** 1 hour (restore from backup)
- **Recovery Point Objective (RPO):** 1 hour (data loss limited to last hour)
- **Disaster Recovery Drill:** Quarterly restore tests to validate recovery procedures

---

## Feasibility Rating per Module

Based on scoping interviews and technical analysis, each of the 8 divisions receives a feasibility rating. **Higher complexity = longer timeline and higher risk.**

| DIVISION | MODULE | COMPLEXITY | RATIONALE | KEY DEPENDENCIES |
|---|---|---|---|---|
| Legal Close Land | Land Closing Workflow | **HIGH** | Complex multi-step process, 500 docs per closing, DocuSign integration, PO reconciliation | DocuSign API ready; RealPage integration for title verification |
| Material Procurement | PO & Procurement Mgmt | **HIGH** | 2000+ POs/year, integration with QuickBooks, vendor portal, approval workflows | QuickBooks API, vendor communication module |
| Plant Division | Equipment & Inventory Tracking | **HIGH** | Not yet built; requirements uncertain. Asset tracking, maintenance logs, depreciation sync | Final requirements from Zac; may be complex depending on automation needs |
| Field Division | Work Orders & Task Management | **HIGH** | Real-time task dispatch, mobile-first, location tracking, photo documentation, time tracking | Mobile app readiness; ADP integration for time sync; offline-first architecture |
| Management/Leasing | Tenant Portal & Lease Mgmt | **HIGH** | Integration with RealPage (highest risk), tenant self-service portal, payment tracking, lease renewals | RealPage API availability (CRITICAL DEPENDENCY) |
| Legal Rollovers | Document Mgmt & Compliance | **LOW** | Document versioning, approval workflow, storage, audit trails; less complex than Land Closing | Standard document management patterns; lower regulatory complexity |
| Land Division | Land Tracking & Title Mgmt | **HIGH** | Title tracking, zoning/permit correlation, future development pipeline, map visualization; 50+ agencies requires complex permitting coordination with tribal knowledge currently undocumented; requires structured discovery during Phase 1 | Permitting portal links; map visualization library (Mapbox/Leaflet) |
| Accounting Hub | Financial Dashboard & Reporting | **HIGH** | **Multi-entity QuickBooks sync** — each ownership entity is a separate QB realm with its own OAuth token and rate-limit bucket. QBO provides no native cross-company API; consolidated reporting must be built in the Webaroo platform. Architecture: per-realm OAuth token vault, sequential sync per realm (to respect the 10-concurrent-request app-level cap shared across all entities), in-platform | Multi-entity QB mapping complexity, API rate limits across entities, no current close process |

| DIVISION | MODULE | COMPLEXITY | RATIONALE | KEY DEPENDENCIES |
|---|---|---|---|---|
| | | | consolidation layer for cross-entity P&L/BS. Entity count to be confirmed in Phase 1 discovery — budget assumes up to 8 entities. Financial dashboards (NOI, cash, funds), budget vs actual (daily), ADP payroll (12 employees, 2 pay schedules), cost allocation engine, AP automation. Monte Carlo handled by Mirrorfish, Resortments' existing external analytics/forecasting tool (not in platform scope). | |

**Module Breakdown:** - **HIGH Complexity (7):** Land Closing, PO & Procurement, Plant, Tenant Portal, Field Operations, Land Division, Accounting Hub - **LOW (1):** Legal Rollovers

**Total Build Effort Estimate:** - **63–87 developer-months over approximately 14 months** (aligns with Budget Estimate and project phases) - Foundation & Auth (Phase 1): ~8–10 dev-months - Low-Complexity Modules (Phase 1–2): ~2–3 dev-months - High-Complexity Modules (Phases 1–4): ~53–74 dev-months - Integration, testing, and infrastructure across all phases

Team composition: 5–7 senior full-stack engineers, 1 mobile specialist, 1 QA engineer, 1 DevOps/infrastructure engineer, 1 product manager, and 1 UX/UI designer. Accounting Hub alone is ~13–17 dev-months of the high-complexity module pool.

---

## Key Technical Risks & Mitigation

### Risk 1: RealPage API Integration Uncertainty ⚠️ CRITICAL

**Risk:** RealPage does not publish a public API; integration approach uncertain. Without RealPage, Management/Leasing module cannot fetch tenant/lease data automatically.

**Impact:** - HIGH: Blocks Tenant Portal module; Dylan cannot use dashboard for occupancy tracking - **Timeline Impact:** +2–3 weeks if forced to data export approach vs. real-time API

**Mitigation:** - **Action (Week 1):** Contact RealPage support to determine API availability for Resortments account - **Fallback (Week 2):** If no API, implement daily CSV export → parse → sync pipeline - **Alternative:** Build manual tenant data entry interface while awaiting RealPage decision - **Owner:** Project Manager to coordinate with Dylan (Management/Leasing) and RealPage

**Decision Gate:** RealPage API decision must be made by end of Week 2 of Discovery phase to finalize timeline.

### Risk 2: DocuSign Volume at Scale

**Risk:** Current estimate is 10,000 signatures/year, but if closings accelerate or multi-property portfolio grows, volume could reach 50,000+/year.

**Impact:** - MEDIUM: DocuSign pricing scales with usage; high volume could trigger rate limiting or require DocuSign optimization - **Cost Impact:** DocuSign pricing increases with envelope volume

**Mitigation:** - **Monitoring:** Track monthly envelope count; establish threshold for optimization (e.g., 1,000/month) - **Optimization:** Implement envelope template batching, parallel processing for multiple

signers - **Contingency:** Early negotiation with DocuSign for volume discounts or custom rate limits - **Alternative Provider:** Consider alternative eSignature providers (Adobe Sign, Docusign competitors) if scaling becomes expensive

### Risk 3: Plant Division Requirements Uncertainty

**Risk:** Plant Division is not yet built; Equipment & Inventory Tracking module requirements are uncertain.

**Impact:** - MEDIUM-HIGH: Cannot finalize Plant module until Zac's requirements are clear - **Timeline Impact:** +1–2 weeks for requirements discovery if unclear

**Mitigation:** - **Action (Week 1-2):** Schedule detailed requirements session with Zac to understand: - Asset management vs. inventory vs. maintenance tracking needs - Automation desired (barcodes, RFID, real-time location tracking) - Integration with financial depreciation tracking - Compliance/regulatory needs (OSHA, equipment certification) - **Scope Gate:** Lock Plant module scope by end of Week 3 to prevent timeline drift - **Owner:** Project Manager to coordinate with Zac (Plant Division)

### Risk 4: Permitting Portal Fragmentation

**Risk:** 50+ regulatory bodies (cities, counties, state agencies) each have unique permitting portals with no API access. Manual status tracking is the only reliable approach.

**Impact:** - MEDIUM: Permitting module will not be fully automated; relies on manual staff updates - **User Experience:** Staff must switch between Resortments and 50+ external portals

**Mitigation:** - **Simplified UX:** Permitting module in Resortments with clickable links to each portal; calculated deadline warnings - **Process Improvement:** Assign one staff member (Legal team) to standardize permit tracking; daily or weekly status batch updates - **RPA Evaluation (Phase 2):** Investigate Robotic Process Automation (UiPath, Automation Anywhere) for top 5-10 portals; ROI analysis for Phase 2 - **Documentation:** Create portal access guide and credential storage (1Password/Vault) for quick access

### Risk 5: Mobile App Offline Sync Complexity

**Risk:** Field teams (Zac, Gerardo) may lose connectivity in remote locations. Mobile app must work offline and sync when reconnected.

**Impact:** - MEDIUM: Complex local storage, conflict resolution, and sync logic needed - **Timeline Impact:** +1–2 weeks for offline-first architecture vs. online-only

**Mitigation:** - **Technology:** Use React Native libraries (Redux Persist, WatermelonDB, or Realm) for local-first state management - **Sync Strategy:** Queue local changes; sync on reconnect with conflict detection (last-write-wins or manual conflict resolution) - **Testing:** Device testing in offline/poor connectivity scenarios (use network throttling tools) - **Owner:** Mobile lead to prototype offline sync early (Week 4-5) to validate approach

### Risk 6: Data Retention & Compliance

**Risk:** 7-10 year data retention requirement requires tiered storage strategy. Storing all data hot (in PostgreSQL) is costly.

**Impact:** - LOW: Solvable with proper architecture, but no quick fix if storage strategy is not planned - **Cost Impact:** ~$5-10K/year for proper tiered storage vs. $20K/year for hot storage only

**Mitigation:** - **Architecture Decision (Week 1):** Define tiered storage strategy: - Hot (0-1 year): PostgreSQL + S3 - Warm (1-7 years): S3-IA (Infrequent Access) - Cold (7-10 years): S3 Glacier - **Implementation:** Automated archive jobs to move data to colder tiers based on age - **Cost Monitoring:** Monthly cost tracking for storage tiers

### Risk 7: Integration Testing & Data Volume

**Risk:** Testing integrations (QuickBooks, ADP, DocuSign, RealPage) requires realistic data volume. Test data + production data may be difficult to separate.

**Impact:** - MEDIUM: Integration bugs discovered late can cause data corruption or compliance issues - **Timeline Impact:** +1 week for comprehensive integration testing

**Mitigation:** - **Sandbox Environments:** Use vendor sandboxes (QuickBooks, DocuSign, ADP all provide) for testing before production - **Test Data Isolation:** Separate test and production databases; data migration only one-way (test → staging → production) - **Integration Checklist:** Document each integration's test scenarios before coding: - Happy path (data syncs correctly) - Error handling (vendor API fails, timeout recovery) - Data validation (malformed data from vendor, graceful degradation) - **Owner:** QA lead to define integration test plan in Week 2

---

## Recommendations & Next Steps

1. **Confirm RealPage API Availability** (Week 1, Critical Path) - Contact RealPage; confirm API access or plan data export fallback - Impact on timeline: +0 weeks if API available, +2–3 weeks if data export required
2. **Finalize Plant Division Requirements** (Week 1-2) - Detailed session with Zac to clarify Equipment/Inventory module scope - Lock requirements by end of Week 3 to prevent timeline drift
3. **Complete Accounting Hub Requirements** (Week 1-2) - Follow-up interviews with Johnny and Teresa on reporting, budget tracking, integrations - Confirmed HIGH complexity (13–18 dev-months) based on scoping interview
4. **Begin Technical Infrastructure Setup** (Week 2) - AWS account provisioning, RDS for PostgreSQL, Auth0 tenant setup - CI/CD pipeline (GitHub Actions), monitoring (CloudWatch), staging environment
5. **Prototype Key Components** (Week 3-4) - React web app shell with role-based navigation - React Native mobile app shell with offline sync (test early) - API gateway and authentication flow - QuickBooks and DocuSign API sandbox integration
6. **Load Testing & Scalability Validation** (Week 5) - Load test API at 50+ concurrent users - Database query optimization (identify slow queries) - Mobile app performance testing on slower networks
7. **Establish QA & Integration Test Strategy** (Week 2) - Define test scenarios for each integration - Set up test data for sandbox environments - Plan production rollout with minimal data loss risk

---

## Conclusion

The Resortments Unified Enterprise Platform is **technically feasible** with a modern tech stack (React/Next.js frontend, React Native mobile, Node.js/Python backend, PostgreSQL database, Auth0 auth) and clear architectural patterns. The proposed modular architecture scales well to support 8 divisions, 50+ concurrent users, and complex workflows.

**Key technical risks** are manageable with proper planning: - RealPage API uncertainty (critical path, must resolve Week 1) - Plant Division requirements (must clarify Week 2) - Permitting portal fragmentation (mitigated with smart UX and RPA roadmap) - DocuSign volume at scale (monitor and optimize proactively)

**Feasibility: High** with recommended tech stack and proper engineering discipline. Timeline is 14 months from Phase 1 kickoff to full deployment.

NAVIGATE THE PACKAGE

# All Discovery Materials