SalesCompass

Complete Deployment Plan

Generated: December 1, 2025
Framework: TIPSI Analysis
Timeline: 16 Sprints (32 Weeks)

SalesCompass Deployment Readiness - TIPSI Analysis

TIPSI Scoring Guide

- **T (Target)**: Clarity of objective (1=vague, 5=crystal clear)
- **I (Information)**: Information available (1=need research, 5=fully documented)
- **P (Priority)**: Business priority (1=nice-to-have, 5=critical blocker)
- **S (Support)**: Support/resources needed (1=minimal, 5=extensive)
- **I (Impact)**: Overall project impact (1=minimal, 5=game-changer)

Total Score Range: 5-25 (Higher = More urgent/impactful)

Work Area 1: Security & Configuration Hardening

TIPSI Score: 23/25 (T=5, I=5, P=5, S=3, I=5) - **Target**: Secure production environment, prevent data breaches - **Information**: Security checklist available, Django docs comprehensive - **Priority**: CRITICAL - Cannot deploy with DEBUG=True and exposed

secrets - **Support**: DevOps knowledge, environment config tools - **Impact**: CRITICAL - Security breach would destroy business

Tasks

[ ] 1.1 Environment Configuration

Subtasks: - [ ] 1.1.1 Create `.env.example` with all required variables - [ ] 1.1.2 Move `SECRET_KEY` to environment variable - [ ] 1.1.3 Move database credentials to environment - [ ] 1.1.4 Move Stripe keys to environment - [ ] 1.1.5 Move Wazo credentials to environment - [ ] 1.1.6 Add `.env` to `.gitignore` - [ ] 1.1.7 Verify no secrets in git history

[ ] 1.2 Settings Split for Environments

Subtasks: - [ ] 1.2.1 Create `settings/base.py` with shared config - [ ] 1.2.2 Create `settings/development.py` - [ ] 1.2.3 Create `settings/staging.py` - [ ] 1.2.4 Create `settings/production.py` - [ ] 1.2.5 Set `DEBUG=False` in production - [ ] 1.2.6 Configure `ALLOWED_HOSTS` from environment - [ ] 1.2.7 Update `manage.py` and `wsgi.py` to use env-based settings

[ ] 1.3 Security Middleware & Headers

Subtasks: - [ ] 1.3.1 Enable `SecurityMiddleware` - [ ] 1.3.2 Set `SECURE_SSL_REDIRECT=True` - [ ] 1.3.3 Set `SESSION_COOKIE_SECURE=True` - [ ] 1.3.4 Set `CSRF_COOKIE_SECURE=True` - [ ] 1.3.5 Configure `SECURE_HSTS_SECONDS` - [ ] 1.3.6 Add `django-cors-headers` and configure - [ ] 1.3.7 Add CSP headers configuration

[ ] 1.4 Authentication Security

Subtasks: - [ ] 1.4.1 Implement password strength requirements - [ ] 1.4.2 Add account lockout after failed attempts - [ ] 1.4.3 Complete MFA implementation - [ ] 1.4.4 Add session timeout configuration - [ ] 1.4.5 Implement password reset security - [ ] 1.4.6 Add API key rotation mechanism

---

Work Area 2: Test Infrastructure & Coverage

TIPSI Score: 22/25 (T=5, I=4, P=5, S=4, I=4) - **Target**: 80%+ test coverage across all modules - **Information**: Some test files exist, need comprehensive expansion - **Priority**: CRITICAL - Cannot deploy untested code - **Support**: Pytest framework, factory_boy for fixtures - **Impact**: HIGH - Prevents bugs, enables CI/CD confidence

Tasks

[ ] 2.1 Test Framework Setup

Subtasks: - [ ] 2.1.1 Install pytest, pytest-django, pytest-cov - [ ] 2.1.2 Install factory_boy and faker - [ ] 2.1.3 Create `pytest.ini` configuration - [ ] 2.1.4 Create `conftest.py` with base fixtures - [ ] 2.1.5 Create tenant factory - [ ] 2.1.6 Create user factory with roles - [ ] 2.1.7 Set up test database configuration

[ ] 2.2 Model Tests (26 apps)

Subtasks: - [ ] 2.2.1 Test `core` models (User, Tenant, Role, Team) - [ ] 2.2.2 Test `leads` models (Lead, LeadSource, LeadStatus) - [ ] 2.2.3 Test `opportunities` models - [ ] 2.2.4 Test `accounts` models (Account, Contact) - [ ] 2.2.5 Test `cases` models - [ ] 2.2.6 Test `tasks` models - [ ] 2.2.7 Test `automation` models (Workflow, Trigger, Action) - [ ] 2.2.8 Test `engagement` models - [ ] 2.2.9 Test `marketing` models - [ ] 2.2.10 Test `billing` models - [ ] 2.2.11 Test remaining 16 apps

[ ] 2.3 View Tests

Subtasks: - [ ] 2.3.1 Test dashboard views - [ ] 2.3.2 Test CRUD views for all modules - [ ] 2.3.3 Test list/filter views - [ ] 2.3.4 Test form validation - [ ] 2.3.5 Test error handling (404, 403, 500)

[ ] 2.4 Permission & RBAC Tests

Subtasks: - [ ] 2.4.1 Test role-based permissions - [ ] 2.4.2 Test object-level permissions - [ ] 2.4.3 Test multi-tenant isolation - [ ] 2.4.4 Test data visibility rules - [ ] 2.4.5 Test permission decorators

[ ] 2.5 Integration Tests

Subtasks: - [ ] 2.5.1 Test lead-to-opportunity conversion flow - [ ] 2.5.2 Test automation trigger workflows - [ ] 2.5.3 Test event emission and handling - [ ] 2.5.4 Test WebSocket/Channels functionality - [ ] 2.5.5 Test Celery task execution

---

Work Area 3: Database Migration & Optimization

TIPSI Score: 24/25 (T=5, I=4, P=5, S=5, I=5) - **Target**: Migrate from SQLite to PostgreSQL with optimized queries - **Information**: Schema known, migration docs available, needs planning - **Priority**: CRITICAL - SQLite unsuitable for production - **Support**: Database admin expertise, migration tools, downtime window - **Impact**: CRITICAL - Foundation for all data operations

Tasks

[ ] 3.1 PostgreSQL Setup

Subtasks: - [ ] 3.1.1 Provision PostgreSQL instance (RDS/Cloud SQL) - [ ] 3.1.2 Configure database parameters for Django - [ ] 3.1.3 Set up connection pooling (pgBouncer) - [ ] 3.1.4 Configure SSL connections - [ ] 3.1.5 Create database users and roles - [ ] 3.1.6 Update `settings/production.py` with PostgreSQL config

[ ] 3.2 Data Migration

Subtasks: - [ ] 3.2.1 Export data from SQLite using `dumpdata` - [ ] 3.2.2 Clean and validate export data - [ ] 3.2.3 Test import on staging PostgreSQL - [ ] 3.2.4 Verify data integrity (row counts, relationships) - [ ] 3.2.5 Create rollback plan - [ ] 3.2.6 Execute production migration - [ ] 3.2.7 Verify all data migrated correctly

[ ] 3.3 Database Indexing

Subtasks: - [ ] 3.3.1 Add indexes on foreign keys (all models) - [ ] 3.3.2 Add indexes on created_at/updated_at fields - [ ] 3.3.3 Add indexes on status/state fields - [ ] 3.3.4 Create composite indexes for common queries - [ ] 3.3.5 Analyze slow queries and add targeted indexes - [ ] 3.3.6 Run `EXPLAIN ANALYZE` on critical queries

[ ] 3.4 Query Optimization

Subtasks: - [ ] 3.4.1 Add `select_related()` for foreign keys - [ ] 3.4.2 Add `prefetch_related()` for reverse relationships - [ ] 3.4.3 Optimize N+1 queries in views - [ ] 3.4.4 Add `only()` and `defer()` where appropriate - [ ] 3.4.5 Implement query result caching - [ ] 3.4.6 Add database query logging/monitoring

---

Work Area 4: Caching & Performance

TIPSI Score: 18/25 (T=4, I=4, P=3, S=3, I=4) - **Target**: Sub-2 second page loads, handle 1000+ concurrent users - **Information**: Redis configuration known, need to identify cache points - **Priority**: HIGH - Performance affects user experience - **Support**: Redis setup, cache strategy design - **Impact**: HIGH - Direct user satisfaction impact

Tasks

[ ] 4.1 Redis Setup

Subtasks: - [ ] 4.1.1 Provision Redis instance (ElastiCache/Cloud Memorystore) - [ ] 4.1.2 Configure Redis for caching - [ ] 4.1.3 Configure Redis for sessions - [ ] 4.1.4 Configure Redis for Celery broker - [ ] 4.1.5 Set up Redis persistence - [ ] 4.1.6 Configure Redis clustering (if needed)

[ ] 4.2 View Caching

Subtasks: - [ ] 4.2.1 Add `@cache_page` to dashboard views - [ ] 4.2.2 Cache widget data - [ ] 4.2.3 Cache report data - [ ] 4.2.4 Cache list views with filters - [ ] 4.2.5 Implement cache invalidation logic

[ ] 4.3 Application Optimization

Subtasks: - [ ] 4.3.1 Enable template caching - [ ] 4.3.2 Implement fragment caching - [ ] 4.3.3 Optimize static file serving - [ ] 4.3.4 Enable gzip compression - [ ] 4.3.5 Minify CSS/JS assets - [ ] 4.3.6 Implement CDN for static files

---

Work Area 5: Stripe Billing Integration

TIPSI Score: 21/25 (T=5, I=3, P=5, S=4, I=4) - **Target**: Complete subscription management and payment processing - **Information**: Stripe API documented, need implementation details - **Priority**: CRITICAL - Required for revenue generation - **Support**: Stripe documentation, webhook testing - **Impact**: HIGH - Core business functionality

Tasks

[ ] 5.1 Stripe Configuration

Subtasks: - [ ] 5.1.1 Set up Stripe account (production mode) - [ ] 5.1.2 Configure webhook endpoints - [ ] 5.1.3 Add Stripe keys to environment - [ ] 5.1.4 Install/update Stripe SDK - [ ] 5.1.5 Configure webhook signature verification

[ ] 5.2 Subscription Management

Subtasks: - [ ] 5.2.1 Implement subscription creation flow - [ ] 5.2.2 Add payment method management - [ ] 5.2.3 Implement subscription updates/cancellation - [ ] 5.2.4 Handle subscription renewal - [ ] 5.2.5 Implement trial period logic - [ ] 5.2.6 Add proration handling

[ ] 5.3 Webhook Handlers

Subtasks: - [ ] 5.3.1 Handle `customer.subscription.created` - [ ] 5.3.2 Handle `customer.subscription.updated` - [ ] 5.3.3 Handle `customer.subscription.deleted` - [ ] 5.3.4 Handle `invoice.payment_succeeded` - [ ] 5.3.5 Handle `invoice.payment_failed` - [ ] 5.3.6 Handle `payment_method.attached` - [ ] 5.3.7 Add error handling and logging

[ ] 5.4 Billing UI

Subtasks: - [ ] 5.4.1 Complete billing dashboard - [ ] 5.4.2 Add subscription status display - [ ] 5.4.3 Implement payment method UI - [ ] 5.4.4 Add invoice history page - [ ] 5.4.5 Implement plan upgrade/downgrade UI

Work Area 6: Wazo Telephony Integration

TIPSI Score: 17/25 (T=4, I=2, P=3, S=5, I=3) - **Target**: Full call center functionality with Wazo platform - **Information**: Limited - need Wazo documentation study - **Priority**: MEDIUM - Important but not deployment blocker - **Support**: High - Wazo expertise required, complex integration - **Impact**: MEDIUM - Adds differentiation but not core CRM

Tasks

[ ] 6.1 Wazo Infrastructure

Subtasks: - [ ] 6.1.1 Deploy Wazo services (auth, confd, call-logd, agentd) - [ ] 6.1.2 Configure Wazo authentication - [ ] 6.1.3 Test Wazo API connectivity - [ ] 6.1.4 Set up Wazo webhooks - [ ] 6.1.5 Configure SIP trunking

[ ] 6.2 Call Features

Subtasks: - [ ] 6.2.1 Implement click-to-call - [ ] 6.2.2 Add call recording retrieval - [ ] 6.2.3 Implement call logging to engagement - [ ] 6.2.4 Add real-time call status - [ ] 6.2.5 Implement call queue management - [ ] 6.2.6 Add voicemail integration

[ ] 6.3 Agent Management

Subtasks: - [ ] 6.3.1 Sync users to Wazo agents - [ ] 6.3.2 Implement agent status tracking - [ ] 6.3.3 Add agent performance metrics - [ ] 6.3.4 Implement queue assignment

---

Work Area 7: Email/SMS & Communication

TIPSI Score: 19/25 (T=5, I=4, P=4, S=3, I=3) - **Target**: Reliable email/SMS delivery with tracking - **Information**: Good - SMTP/API docs available - **Priority**: HIGH - Critical for marketing and engagement - **Support**: Medium - Provider setup, template design - **Impact**: MEDIUM - Enhances communication but alternatives exist

Tasks

[ ] 7.1 Email Service Setup

Subtasks: - [ ] 7.1.1 Choose provider (SendGrid/SES/Mailgun) - [ ] 7.1.2 Configure SMTP settings - [ ] 7.1.3 Set up email authentication (SPF/DKIM/DMARC) - [ ] 7.1.4 Configure email templates - [ ] 7.1.5 Implement email tracking (opens/clicks) - [ ] 7.1.6 Add unsubscribe handling

[ ] 7.2 SMS Integration

Subtasks: - [ ] 7.2.1 Set up Twilio account - [ ] 7.2.2 Configure Twilio credentials - [ ] 7.2.3 Implement SMS sending - [ ] 7.2.4 Add SMS delivery tracking - [ ] 7.2.5 Implement opt-out handling

[ ] 7.3 Template Management

Subtasks: - [ ] 7.3.1 Complete email template builder - [ ] 7.3.2 Add template variables/placeholders - [ ] 7.3.3 Implement template preview - [ ] 7.3.4 Add template versioning

---

Work Area 8: Production Infrastructure

TIPSI Score: 23/25 (T=5, I=4, P=5, S=4, I=5) - **Target**: Production-ready containerized deployment - **Information**: Good - Docker best practices documented - **Priority**: CRITICAL - Required for deployment - **Support**: Medium-High - DevOps expertise needed - **Impact**: CRITICAL - Entire deployment depends on this

Tasks

[ ] 8.1 Docker Production Config

Subtasks: - [ ] 8.1.1 Create multi-stage Dockerfile - [ ] 8.1.2 Create production docker-compose.yml - [ ] 8.1.3 Add Nginx reverse proxy - [ ] 8.1.4 Configure SSL/TLS certificates - [ ] 8.1.5 Add health check endpoints - [ ] 8.1.6 Configure container logging - [ ] 8.1.7 Implement security scanning

[ ] 8.2 Web Server Configuration

Subtasks: - [ ] 8.2.1 Configure Gunicorn for WSGI - [ ] 8.2.2 Configure Daphne for ASGI (WebSockets) - [ ] 8.2.3 Set up Nginx upstream config - [ ] 8.2.4 Configure load balancing - [ ] 8.2.5 Add rate limiting - [ ] 8.2.6 Configure request logging

[ ] 8.3 Backup & Recovery

Subtasks: - [ ] 8.3.1 Implement automated database backups - [ ] 8.3.2 Set up backup storage (S3/GCS) - [ ] 8.3.3 Create backup restoration scripts - [ ] 8.3.4 Test backup/restore process - [ ] 8.3.5 Document recovery procedures - [ ] 8.3.6 Set up backup monitoring/alerts

---

Work Area 9: Monitoring & Observability

TIPSI Score: 20/25 (T=5, I=4, P=4, S=4, I=3) - **Target**: Complete visibility into system health and performance - **Information**: Good - Tools documented, need integration - **Priority**: HIGH - Critical for production operations - **Support**: Medium-High - Monitoring stack setup - **Impact**: MEDIUM-HIGH - Enables proactive issue resolution

Tasks

[ ] 9.1 Error Tracking

Subtasks: - [ ] 9.1.1 Set up Sentry account - [ ] 9.1.2 Install Sentry SDK - [ ] 9.1.3 Configure error capturing - [ ] 9.1.4 Set up alert notifications - [ ] 9.1.5 Configure error grouping - [ ] 9.1.6 Add custom error context

[ ] 9.2 Application Monitoring

Subtasks: - [ ] 9.2.1 Set up Prometheus - [ ] 9.2.2 Add Django metrics exporter - [ ] 9.2.3 Set up Grafana dashboards - [ ] 9.2.4 Configure uptime monitoring - [ ] 9.2.5 Add custom business metrics - [ ] 9.2.6 Set up alerting rules

[ ] 9.3 Logging Infrastructure

Subtasks: - [ ] 9.3.1 Configure centralized logging (ELK/CloudWatch) - [ ] 9.3.2 Set up log aggregation - [ ] 9.3.3 Configure log retention policies - [ ] 9.3.4 Add structured logging - [ ] 9.3.5 Create log analysis dashboards

---

Work Area 10: CI/CD Pipeline

TIPSI Score: 19/25 (T=5, I=5, P=4, S=3, I=2) - **Target**: Automated testing and deployment pipeline - **Information**: Excellent - GitHub Actions/GitLab CI documented - **Priority**: HIGH - Enables rapid, safe deployments - **Support**: Medium - CI/CD platform setup - **Impact**: LOW-MEDIUM - Improves development velocity

Tasks

[ ] 10.1 CI Pipeline

Subtasks: - [ ] 10.1.1 Create GitHub Actions workflow - [ ] 10.1.2 Add automated test runs - [ ] 10.1.3 Add code quality checks (flake8, black) - [ ] 10.1.4 Add security scanning - [ ] 10.1.5 Add dependency vulnerability checks - [ ] 10.1.6 Configure branch protection rules

[ ] 10.2 CD Pipeline

Subtasks: - [ ] 10.2.1 Create staging deployment workflow - [ ] 10.2.2 Create production deployment workflow - [ ] 10.2.3 Add deployment approval gates - [ ] 10.2.4 Implement blue-green deployment - [ ] 10.2.5 Add rollback capability - [ ] 10.2.6 Configure deployment notifications

---

Work Area 11: Missing Features & UI Fixes

TIPSI Score: 16/25 (T=3, I=3, P=3, S=4, I=3) - **Target**: Complete all missing views and fix broken navigation - **Information**: Partially known - need to audit each orphan - **Priority**: MEDIUM - Improves completeness but many are nice-to-have - **Support**:

Medium-High - Requires design and implementation - **Impact**: MEDIUM - Better UX but not critical

Tasks

[ ] 11.1 Complete Missing Dashboard Views

Subtasks: - [ ] 11.1.1 Implement `dashboard:cockpit` - [ ] 11.1.2 Implement `dashboard:admin_dashboard` - [ ] 11.1.3 Implement `dashboard:manager_dashboard` - [ ] 11.1.4 Implement `dashboard:support_dashboard`

[ ] 11.2 Complete Commissions Module

Subtasks: - [ ] 11.2.1 Create `commissions:list` view - [ ] 11.2.2 Create `commissions:history` view - [ ] 11.2.3 Add commission tracking UI

[ ] 11.3 Fix Navigation Issues

Subtasks: - [ ] 11.3.1 Audit 100 broken links (403 errors) - [ ] 11.3.2 Fix permission decorators - [ ] 11.3.3 Update navigation templates - [ ] 11.3.4 Test all navigation flows

[ ] 11.4 Clean Up Orphaned Templates

Subtasks: - [ ] 11.4.1 Review 104 orphaned templates - [ ] 11.4.2 Integrate valuable orphans - [ ] 11.4.3 Remove deprecated templates - [ ] 11.4.4 Document intentional orphans

---

Work Area 12: Documentation & Training

TIPSI Score: 15/25 (T=5, I=4, P=2, S=3, I=1) - **Target**: Comprehensive documentation for users and developers - **Information**: Good - structure known, content needed - **Priority**: LOW-MEDIUM - Important for adoption, not deployment - **Support**: Medium - Technical writing effort - **Impact**: LOW - Helps adoption but system works without it

Tasks

[ ] 12.1 Technical Documentation

Subtasks: - [ ] 12.1.1 Write API documentation - [ ] 12.1.2 Document architecture - [ ] 12.1.3 Create deployment guide - [ ] 12.1.4 Write operations runbook - [ ] 12.1.5 Document troubleshooting procedures

[ ] 12.2 User Documentation

Subtasks: - [ ] 12.2.1 Create user guide - [ ] 12.2.2 Write feature tutorials - [ ] 12.2.3 Create video walkthroughs - [ ] 12.2.4 Build help center content

[ ] 12.3 Legal Documents

Subtasks: - [ ] 12.3.1 Draft Terms of Service - [ ] 12.3.2 Draft Privacy Policy - [ ] 12.3.3 Create Data Processing Agreement - [ ] 12.3.4 Add Cookie Policy

---

Priority Matrix (by TIPSI Score)

Critical (23-25 points) - Start Immediately: 1. Security & Configuration Hardening (23) 2. Database Migration & Optimization (24) 3. Production Infrastructure (23)

High Priority (19-22 points) - Sprint 2-3: 4. Test Infrastructure & Coverage (22) 5. Stripe Billing Integration (21) 6. Monitoring & Observability (20) 7. Email/SMS & Communication (19) 8. CI/CD Pipeline (19)

Medium Priority (16-18 points) - Sprint 4-5: 9. Caching & Performance (18) 10. Wazo Telephony Integration (17) 11. Missing Features & UI Fixes (16)

Lower Priority (15 and below) - Post-Launch: 12. Documentation & Training (15)

SalesCompass Sprint Schedule - TIPSI Prioritized Deployment Plan

Schedule Overview

Total Duration: 16 sprints (32 weeks / 8 months)
Team Size: 2-3 developers
Sprint Length: 2 weeks
Deployment Target: Sprint 16 completion

Sprint Allocation by TIPSI Score

Work Area	TIPSI Score	Sprints	Phase
Security & Configuration	23	Sprint 1	Phase 1
Database Migration	24	Sprint 2-3	Phase 1
Production Infrastructure	23	Sprint 4-5	Phase 1
Test Infrastructure	22	Sprint 6-7	Phase 2
Stripe Billing	21	Sprint 8	Phase 2
Monitoring & Observability	20	Sprint 9	Phase 3
Email/SMS Communication	19	Sprint 10	Phase 3
CI/CD Pipeline	19	Sprint 11	Phase 3
Caching & Performance	18	Sprint 12	Phase 4

Work Area	TIPSI Score	Sprints	Phase
Wazo Telephony	17	Sprint 13	Phase 4
Missing Features & UI	16	Sprint 14	Phase 4
Documentation	15	Sprint 15-16	Phase 5

Phase 1: Critical Infrastructure (Sprints 1-5)

Sprint 1: Security & Configuration Hardening

TIPSI Score: 23 | **Weeks: 1-2**

Goal: Eliminate critical security vulnerabilities

Deliverables: - [x] Environment variable configuration (`.env` , `.env.example`) - [x] Settings split (base, dev, staging, production) - [x] `DEBUG=False` in production - [x] `SECRET_KEY` moved to environment - [x] Security middleware enabled (SSL redirect, HSTS, secure cookies) - [x] CORS headers configured - [x] Password policies implemented - [x] Account lockout mechanism - [x] MFA completion

Acceptance Criteria: - [ ] `python manage.py check --deploy` passes with no warnings - [ ] No secrets in git repository - [ ] SSL/HTTPS enforced - [ ] Session cookies marked secure

Dependencies: None
Blockers: None
Risk: Low - established patterns

Sprint 2: Database Migration - Part 1 (PostgreSQL Setup)

TIPSI Score: 24 | **Weeks: 3-4**

Goal: Set up production PostgreSQL infrastructure

Deliverables: - [x] PostgreSQL instance provisioned - [x] Connection pooling (pgBouncer) configured - [x] SSL connections enabled - [x] Database users and roles created - [x] Settings updated for PostgreSQL - [x] Data export from SQLite completed - [x] Data validation scripts created

Acceptance Criteria: - [ ] PostgreSQL accepting connections - [ ] Data exported successfully from SQLite - [ ] Row counts match between SQLite and export

Dependencies: Sprint 1 (production settings)
Blockers: Database hosting decision
Risk: Medium - data migration complexity

Sprint 3: Database Migration - Part 2 (Migration & Optimization)

TIPSI Score: 24 | **Weeks: 5-6**

Goal: Complete migration and optimize database

Deliverables: - [x] Data imported to PostgreSQL - [x] Data integrity verified - [x] Indexes on foreign keys - [x] Indexes on timestamp fields - [x] Composite indexes for common queries - [x] Query optimization (select_related, prefetch_related) - [x] Rollback plan documented

Acceptance Criteria: - [ ] All data migrated successfully - [ ] Foreign key relationships intact - [ ] Query performance improved >50% - [ ] Backup/restore tested

Dependencies: Sprint 2
Blockers: None
Risk: High - potential data loss if not careful

Sprint 4: Production Infrastructure - Part 1 (Containerization)

TIPSI Score: 23 | **Weeks: 7-8**

Goal: Production-ready Docker setup

Deliverables: - [x] Multi-stage Dockerfile - [x] Production docker-compose.yml - [x] Nginx reverse proxy configured - [x] SSL/TLS certificates (Let's Encrypt) - [x] Health check endpoints - [x] Container logging configured - [x] Security scanning integrated

Acceptance Criteria: - [ ] Docker image builds successfully - [ ] All services start via docker-compose - [ ] HTTPS working with valid certificates - [ ] Health checks passing

Dependencies: Sprint 1 (env config), Sprint 3 (PostgreSQL)
Blockers: Hosting platform decision
Risk: Medium - infrastructure complexity

Sprint 5: Production Infrastructure - Part 2 (Backup & Web Server)

TIPSI Score: 23 | **Weeks: 9-10**

Goal: Complete production infrastructure

Deliverables: - [x] Gunicorn configured for WSGI - [x] Daphne configured for ASGI (WebSockets) - [x] Load balancing configured - [x] Rate limiting implemented - [x] Automated database backups - [x] Backup storage (S3/GCS) configured - [x] Backup restoration tested - [x] Disaster recovery plan documented

Acceptance Criteria: - [ ] Application serving requests via Gunicorn/Daphne - [ ] WebSockets working - [ ] Backups running automatically - [ ] Restoration successful from backup

Dependencies: Sprint 4
Blockers: None
Risk: Low - proven technologies

Phase 2: Testing & Core Integrations (Sprints 6-8)

Sprint 6: Test Infrastructure - Part 1 (Framework & Models)

TIPSI Score: 22 | **Weeks: 11-12**

Goal: Establish testing framework and test core models

Deliverables: - [x] pytest, pytest-django, pytest-cov installed - [x] factory_boy and faker installed - [x] `pytest.ini` and `conftest.py` created - [x] Base fixtures (tenant, user, roles) - [x] Model tests for 10 core apps (core, leads, opportunities, accounts, cases, tasks, automation, engagement, marketing, billing) - [x] Code coverage >60%

Acceptance Criteria: - [ ] Test suite runs successfully - [ ] Coverage report generated - [ ] All model creation tests pass - [ ] Multi-tenant isolation verified

Dependencies: Sprint 3 (PostgreSQL for test DB)
Blockers: None
Risk: Low - standard testing setup

Sprint 7: Test Infrastructure - Part 2 (Views & Integration)

TIPSI Score: 22 | **Weeks: 13-14**

Goal: Complete test coverage

Deliverables: - [x] View tests for all modules - [x] Permission & RBAC tests - [x] Object-level permission tests - [x] Integration tests (lead→opportunity flow, automation triggers) - [x] WebSocket/Channels tests - [x] Celery task tests - [x] Code coverage >80%

Acceptance Criteria: - [ ] All critical user flows tested - [ ] Permission system fully tested - [ ] 80%+ code coverage achieved - [ ] All tests passing

Dependencies: Sprint 6
Blockers: None
Risk: Medium - comprehensive testing effort

Sprint 8: Stripe Billing Integration

TIPSI Score: 21 | **Weeks: 15-16**

Goal: Complete payment processing and subscriptions

Deliverables: - [x] Stripe production account configured - [x] Webhook endpoints implemented - [x] Subscription creation flow - [x] Payment method management - [x] Subscription updates/cancellation - [x] Webhook handlers (all critical events) - [x] Invoice generation - [x] Billing dashboard UI - [x] Error handling and retry logic

Acceptance Criteria: - [ ] Test subscription created successfully - [ ] Webhooks processing correctly - [ ] Payment methods can be added/removed - [ ] Invoices generated and stored - [ ] Failed payment handling works

Dependencies: Sprint 1 (env config for keys)
Blockers: Stripe account approval
Risk: Medium - webhook complexity

Phase 3: Observability & Communication (Sprints 9-11)

Sprint 9: Monitoring & Observability

TIPSI Score: 20 | **Weeks: 17-18**

Goal: Complete visibility into system health

Deliverables: - [x] Sentry configured for error tracking - [x] Prometheus metrics exporter - [x] Grafana dashboards - [x] Uptime monitoring - [x] Custom business metrics - [x] Alert rules configured - [x] Centralized logging (ELK/CloudWatch) - [x] Log retention policies

Acceptance Criteria: - [ ] Errors captured in Sentry - [ ] Dashboards showing key metrics - [ ] Alerts triggering correctly - [ ] Logs searchable and aggregated

Dependencies: Sprint 5 (production environment)
Blockers: Monitoring tool selection
Risk: Low - standard monitoring setup

Sprint 10: Email/SMS Communication

TIPSI Score: 19 | **Weeks: 19-20**

Goal: Reliable communication channels

Deliverables: - [x] Email provider configured (SendGrid/SES) - [x] SMTP/API integration complete - [x] SPF/DKIM/DMARC configured - [x] Email tracking (opens/clicks) - [x] Unsubscribe handling - [x] Twilio SMS integration - [x] SMS delivery tracking - [x] Opt-out handling - [x] Template builder complete - [x] Template preview working

Acceptance Criteria: - [ ] Emails delivering successfully - [ ] Tracking pixels working - [ ] SMS sending and tracking working - [ ] Templates can be created and edited

Dependencies: Sprint 1 (env config)
Blockers: Provider account setup
Risk: Low - established integrations

Sprint 11: CI/CD Pipeline

TIPSI Score: 19 | **Weeks: 21-22**

Goal: Automated deployment pipeline

Deliverables: - [x] GitHub Actions workflow created - [x] Automated test runs on PR - [x] Code quality checks (flake8, black) - [x] Security scanning (Bandit, Safety) - [x] Dependency vulnerability checks - [x] Staging deployment workflow - [x] Production deployment workflow - [x] Deployment approval gates - [x] Rollback capability

Acceptance Criteria: - [ ] Tests run automatically on every PR - [ ] Deployments triggered by merge to main - [ ] Staging environment auto-deploys - [ ] Production requires manual approval - [ ] Rollback tested and working

Dependencies: Sprint 5 (infrastructure), Sprint 7 (tests)
Blockers: CI/CD platform access
Risk: Low - standard DevOps practice

Phase 4: Performance & Feature Completion (Sprints 12-14)

Sprint 12: Caching & Performance

TIPSI Score: 18 | **Weeks: 23-24**

Goal: Optimize application performance

Deliverables: - [x] Redis instance provisioned - [x] Redis for caching configured - [x] Redis for sessions configured - [x] View caching implemented - [x] Widget data caching - [x] Template fragment caching - [x] Static file optimization (minify, compress) - [x] CDN configured - [x] Performance benchmarking complete

Acceptance Criteria: - [ ] Page load times <2 seconds (p95) - [ ] Dashboard loads <1 second with cache - [ ] 50%+ reduction in database queries - [ ] Static files served via CDN

Dependencies: Sprint 3 (database optimization), Sprint 5 (infrastructure)
Blockers: CDN provider selection
Risk: Low - standard caching patterns

Sprint 13: Wazo Telephony Integration

TIPSI Score: 17 | **Weeks: 25-26**

Goal: Call center functionality

Deliverables: - [x] Wazo services deployed - [x] Wazo authentication configured - [x] API connectivity tested - [x] Click-to-call implemented - [x] Call recording retrieval - [x] Call logging to engagement - [x] Real-time call status - [x] Agent management - [x] Queue management

Acceptance Criteria: - [ ] Calls can be initiated from CRM - [ ] Call logs stored correctly - [ ] Agent status syncs properly - [ ] Call recordings accessible

Dependencies: Sprint 5 (infrastructure)
Blockers: Wazo expertise/consultant
Risk: High - complex integration, limited documentation

Sprint 14: Missing Features & UI Fixes

TIPSI Score: 16 | **Weeks: 27-28**

Goal: Complete missing features and fix navigation

Deliverables: - [x] Dashboard views (cockpit, admin, manager, support) - [x] Commissions module (list, history) - [x] Sales module URLs fixed - [x] Navigation issues resolved (100 broken links) - [x] Orphaned templates reviewed (104 items) - [x] Valuable orphans integrated - [x] Deprecated templates removed - [x] Mobile responsiveness tested - [x] Accessibility improvements (WCAG 2.1)

Acceptance Criteria: - [ ] All navigation links working - [ ] No 404 errors on navigation - [ ] Permission-based navigation working - [ ] Mobile experience smooth - [ ] Accessibility audit passed

Dependencies: Sprint 7 (tests for new features)
Blockers: None
Risk: Medium - scope creep potential

Phase 5: Launch Preparation (Sprints 15-16)

Sprint 15: User Acceptance Testing & Bug Fixes

TIPSI Score: N/A | **Weeks: 29-30**

Goal: Validate system with real users

Deliverables: - [x] UAT environment configured - [x] 10 beta users onboarded - [x] UAT test plan executed - [x] All P0/P1 bugs fixed - [x] All P2 bugs triaged - [x] Performance testing completed - [x] Security penetration testing - [x] Load testing (1000+ concurrent users)

Acceptance Criteria: - [ ] No P0 bugs remaining - [ ] <5 P1 bugs remaining (documented) - [ ] Load test targets met - [ ] No critical security vulnerabilities - [ ] User feedback >4/5 satisfaction

Dependencies: All previous sprints
Blockers: Beta user availability
Risk: Medium - unpredictable bug severity

Sprint 16: Documentation & Go-Live

TIPSI Score: 15 | **Weeks: 31-32**

Goal: Production launch

Deliverables: - [x] API documentation complete - [x] Deployment guide written - [x] Architecture documentation - [x] Operations runbook - [x] Troubleshooting guide - [x] User guide and tutorials - [x] Terms of Service - [x] Privacy Policy - [x] Data Processing

Agreement - [x] Support team training - [x] Production deployment - [x] Post-launch monitoring

Acceptance Criteria: - [ ] All documentation published - [ ] Legal docs approved - [ ] Support team trained - [ ] Production deployed successfully - [ ] No critical issues 24 hours post-launch

Dependencies: Sprint 15 (all bugs fixed)
Blockers: Legal review
Risk: Low - final polish

Sprint Dependencies Map

```
graph TD
    S1[Sprint 1: Security] --> S2[Sprint 2: DB Setup]
    S1 --> S4[Sprint 4: Docker]
    S2 --> S3[Sprint 3: DB Migration]
    S3 --> S4
    S3 --> S6[Sprint 6: Tests 1]
    S4 --> S5[Sprint 5: Infrastructure]
    S5 --> S9[Sprint 9: Monitoring]
    S6 --> S7[Sprint 7: Tests 2]
    S1 --> S8[Sprint 8: Stripe]
    S1 --> S10[Sprint 10: Email/SMS]
    S5 --> S11[Sprint 11: CI/CD]
    S7 --> S11
    S3 --> S12[Sprint 12: Caching]
    S5 --> S12
    S5 --> S13[Sprint 13: Wazo]
    S7 --> S14[Sprint 14: UI Fixes]
    S1 --> S15[Sprint 15: UAT]
    S8 --> S15
    S9 --> S15
    S10 --> S15
    S11 --> S15
```

```
    S12 --> S15
    S13 --> S15
    S14 --> S15
    S15 --> S16[Sprint 16: Go-Live]
```

Resource Allocation

Developer 1 (Backend Focus)

- Sprints 1-3: Security & Database
- Sprints 6-7: Testing
- Sprints 8, 10: Integrations (Stripe, Email)
- Sprint 13: Wazo integration
- Sprint 15: Bug fixes

Developer 2 (DevOps/Infrastructure Focus)

- Sprints 4-5: Infrastructure
- Sprint 9: Monitoring
- Sprint 11: CI/CD
- Sprint 12: Caching
- Sprint 15: Performance testing

Developer 3 (Full-Stack, Optional)

- Sprint 14: UI fixes
- Sprints 6-7: Testing support
- Sprint 15: UAT coordination
- Sprint 16: Documentation

Success Metrics

Sprint-Level Metrics

- **Velocity**: Target 30-40 story points per sprint
- **Sprint Goal Achievement**: >90%
- **Defect Escape Rate**: <5%
- **Test Coverage Increase**: +10% per sprint (Sprints 6-7)

Project-Level Metrics

Technical Metrics: - Test Coverage: ≥80% [Target: Sprint 7] - Response Time (p95): <2s [Target: Sprint 12] - Uptime: ≥99.9% [Target: Sprint 16] - Error Rate: <0.1% [Target: Sprint 9]

Performance Metrics: - Page Load Time: <3s [Target: Sprint 12] - Database Queries per Request: <50 [Target: Sprint 3] - Cache Hit Rate: >80% [Target: Sprint 12] - Concurrent Users Supported: 1000+ [Target: Sprint 15]

Security Metrics: - Critical Vulnerabilities: 0 [Target: Sprint 1] - Security Score (Mozilla Observatory): A+ [Target: Sprint 15] - Penetration Test Pass: Yes [Target: Sprint 15]

Risk Management

Critical Risks

Risk	TIPSI Impact	Mitigation	Sprint
Data migration failure	24 (Database)	Thorough testing, rollback plan, staging first	S2-3
Wazo integration complexity	17 (Telephony)	Early spike, consultant if needed, MVP scope	S13
	18 (Caching)	Load testing early, scaling strategy	

Risk	TIPSI Impact	Mitigation	Sprint
Performance under load			S12, S15
Security breach pre-launch	23 (Security)	Security audit, pen testing	S1, S15

Medium Risks

Risk	Mitigation	Sprint
Stripe webhook issues	Extensive testing, retry logic	S8
Test coverage takes longer	Pair programming, parallel work	S6-7
UAT uncovers major bugs	Buffer time, P0/P1 focus	S15

Phase Gates

Phase 1 Gate (End of Sprint 5)

Criteria: - [ ] Security audit passed - [ ] Database migrated successfully - [ ] Production infrastructure deployed - [ ] Health checks passing - [ ] Backups automated

Decision: GO/NO-GO to Phase 2

Phase 2 Gate (End of Sprint 8)

Criteria: - [ ] Test coverage ≥80% - [ ] Stripe integration tested end-to-end - [ ] All tests passing in CI/CD - [ ] No P0 bugs in backlog

Decision: GO/NO-GO to Phase 3

Phase 3 Gate (End of Sprint 11)

Criteria: - [ ] Monitoring dashboards operational - [ ] Email/SMS sending reliably - [ ] CI/CD deploying to staging automatically - [ ] Alerts configured and tested

Decision: GO/NO-GO to Phase 4

Phase 4 Gate (End of Sprint 14)

Criteria: - [ ] Performance targets met - [ ] All navigation working - [ ] Mobile responsive - [ ] Feature complete per MVP scope

Decision: GO/NO-GO to UAT

Phase 5 Gate (End of Sprint 16)

Criteria: - [ ] UAT passed (user satisfaction ≥4/5) - [ ] All P0/P1 bugs resolved - [ ] Documentation complete - [ ] Legal docs approved - [ ] Production deployed successfully - [ ] 24-hour stability confirmed

Decision: GO-LIVE APPROVED

Contingency Plans

If Database Migration Fails (Sprint 3)

- **Plan A**: Restore to SQLite, extend Sprint 3 by 1 week
- **Plan B**: Use hybrid approach temporarily
- **Plan C**: Hire database consultant

If Test Coverage Below Target (Sprint 7)

- **Plan A**: Extend Sprint 7 by 1 week, add developer
- **Plan B**: Focus on critical paths only, defer nice-to-have tests
- **Plan C**: Accept 70% coverage, commit to post-launch improvement

If UAT Reveals Critical Issues (Sprint 15)

- **Plan A**: Add Sprint 15.5 (1 week) for fixes
- **Plan B**: Delay launch by 1 sprint, full regression testing
- **Plan C**: Soft launch with limited users

If Wazo Integration Blocked (Sprint 13)

- **Plan A**: Defer to post-launch, use basic calling features
- **Plan B**: Hire Wazo consultant for 2-week engagement
- **Plan C**: Consider alternative telephony solution

Post-Launch Plan (Month 1-3)

Month 1

- **Week 1-2**: Intensive monitoring, hot-fix releases
- **Week 3-4**: User feedback collection, analytics review

Month 2

- **Week 5-6**: Performance optimization based on real usage
- **Week 7-8**: Feature enhancements from user feedback

Month 3

- **Week 9-10**: Technical debt reduction
- **Week 11-12**: Planning for next major release

Document Version: 1.0
Last Updated: 2025-12-01
Owner: Development Team
Next Review: After Sprint 5 (Phase 1 Gate)

SalesCompass Module Analysis - TIPSI Framework

Module Collaboration Map

```
graph TB
    subgraph "Core Foundation"
        CORE[Core<br/>Users, Roles, Auth]
        TENANTS[Tenants<br/>Multi-tenancy]
        BILLING[Billing<br/>Subscriptions]
    end

    subgraph "CRM Pipeline"
        LEADS[Leads<br/>Lead Management]
        OPPS[Opportunities<br/>Deal Management]
        ACCOUNTS[Accounts<br/>Customer Data]
        CONTACTS[Contacts<br/>in Accounts]
    end

    subgraph "Supporting Apps"
        TASKS[Tasks<br/>Activity Management]
        ENGAGEMENT[Engagement<br/>Interaction Tracking]
        CASES[Cases<br/>Support Tickets]
        NPS[NPS<br/>Satisfaction]
    end

    subgraph "Revenue"
        PRODUCTS[Products<br/>Catalog]
        PROPOSALS[Proposals<br/>Quotes]
        COMMISSIONS[Commissions<br/>Incentives]
        SALES[Sales]
    end
```

```
    subgraph "Marketing & Automation"
        MARKETING[Marketing<br/>Campaigns]
        AUTOMATION[Automation<br/>Workflows]
        COMMUNICATION[Communication<br/>Email/SMS]
    end

    subgraph "Insights"
        REPORTS[Reports<br/>Analytics]
        DASHBOARD[Dashboard<br/>Views]
        LEARN[Learn<br/>Knowledge Base]
    end

    subgraph "System"
        SETTINGS[Settings<br/>Configuration]
        DEVELOPER[Developer<br/>API Access]
        AUDIT[Audit Logs]
        FEATURES[Feature Flags]
        ALERTS[Global Alerts]
        INFRA[Infrastructure]
    end

    CORE --> |authenticates| LEADS
    CORE --> |authenticates| OPPS
    CORE --> |authenticates| ACCOUNTS
    TENANTS --> |isolates| LEADS
    TENANTS --> |isolates| OPPS
    BILLING --> |limits| LEADS
    BILLING --> |limits| CORE

    LEADS --> |converts to| OPPS
    OPPS --> |belongs to| ACCOUNTS
    ACCOUNTS --> |has| CONTACTS

    OPPS --> |contains| PRODUCTS
    OPPS --> |generates| PROPOSALS
```

```
    OPPS --> |triggers| COMMISSIONS

    ENGAGEMENT --> |tracks| LEADS
    ENGAGEMENT --> |tracks| OPPS
    ENGAGEMENT --> |tracks| ACCOUNTS

    TASKS --> |assigned for| LEADS
    TASKS --> |assigned for| OPPS
    TASKS --> |assigned for| CASES

    AUTOMATION --> |triggers| TASKS
    AUTOMATION --> |triggers| COMMUNICATION
    AUTOMATION --> |triggers| CASES

    MARKETING --> |generates| LEADS
    MARKETING --> |uses| COMMUNICATION

    REPORTS --> |analyzes| LEADS
    REPORTS --> |analyzes| OPPS
    REPORTS --> |analyzes| COMMISSIONS

    DASHBOARD --> |displays| REPORTS
    DASHBOARD --> |displays| TASKS

    SETTINGS --> |configures| AUTOMATION
    SETTINGS --> |configures| LEADS
    SETTINGS --> |configures| OPPS

    AUDIT --> |logs| CORE
    AUDIT --> |logs| SETTINGS
    FEATURES --> |controls| MARKETING
    ALERTS --> |notifies| CORE
```

MODULE ANALYSIS

Phase 1: Core Foundation Modules

1. CORE (User, Role, Permission, Team Management)

TIPSI Score: 25/25 (T=5, I=5, P=5, S=5, I=5)

Target

Provide authentication, authorization (RBAC), and user management foundation for entire system

Information Available

- User, Role, Permission models exist
- RBAC decorators implemented
- Multi-tenant user isolation in place
- MFA field exists but incomplete

Priority

CRITICAL - Every module depends on this

Support Required

- Complete MFA implementation
- Role seeding for default roles
- Permission documentation
- User management UI completion

Impact

CRITICAL - Foundation for security and access control across all 25 other modules

Tasks

Task 1.1: Complete Authentication System - [x] 1.1.1 User model with tenant isolation - [ ] 1.1.2 Complete MFA implementation (TOTP) - [ ] 1.1.3 Password reset flow - [ ] 1.1.4 Email verification - [ ] 1.1.5 Session management - [ ] 1.1.6 OAuth2/OIDC integration

Task 1.2: RBAC Implementation - [x] 1.2.1 Role model with permissions list - [x] 1.2.2 Permission decorator (`@require_permission`) - [x] 1.2.3 Permission mixin for CBVs - [ ] 1.2.4 Seed default roles (Admin, Manager, Rep, Support) - [ ] 1.2.5 Role assignment UI - [ ] 1.2.6 Permission testing suite

Task 1.3: User Management - [ ] 1.3.1 User CRUD views - [ ] 1.3.2 User profile management - [ ] 1.3.3 Avatar upload - [ ] 1.3.4 User activity tracking - [ ] 1.3.5 User deactivation workflow

Collaboration Dependencies: - **Provides to**: All 25 modules (authentication/authorization) - **Depends on**: tenants (multi-tenant isolation) - **Critical for**: billing (user limits), audit_logs (user actions)

---

2. TENANTS (Multi-Tenancy Management)

TIPSI Score: 24/25 (T=5, I=5, P=5, S=4, I=5)

Target

Complete tenant isolation, provisioning, and lifecycle management

Information Available

- TenantModel base class exists
- Tenant_id isolation working
- Plan selection flow partially complete

Priority

CRITICAL - Needed for SaaS deployment

Support Required

- Tenant provisioning automation
- Domain/subdomain configuration
- Tenant migration tools

Impact

CRITICAL - Enables multi-customer SaaS model

Tasks

Task 2.1: Tenant Provisioning - [ ] 2.1.1 Tenant signup flow - [ ] 2.1.2 Automatic tenant_id assignment - [ ] 2.1.3 Default data seeding per tenant - [ ] 2.1.4 Tenant configuration dashboard - [ ] 2.1.5 Subdomain/domain mapping

Task 2.2: Tenant Lifecycle - [ ] 2.2.1 Tenant activation/deactivation - [ ] 2.2.2 Tenant suspension (billing issues) - [ ] 2.2.3 Tenant data export (GDPR) - [ ] 2.2.4 Tenant deletion with data cleanup - [ ] 2.2.5 Tenant migration between plans

Task 2.3: Module Entitlements - [ ] 2.3.1 Feature flag per tenant - [ ] 2.3.2 Module access control by plan - [ ] 2.3.3 Usage limit enforcement integration - [ ] 2.3.4 Entitlement dashboard

Collaboration Dependencies: - **Provides to**: All modules (tenant isolation) - **Depends on**: billing (plan-based limits) - **Integrated with**: feature_flags (tenant-specific features)

---

3. BILLING (Subscription & Payment Management)

TIPSI Score: 21/25 (T=5, I=3, P=5, S=4, I=4)

Target

Subscription management, payment processing, usage-based limits

Information Available

- Subscription, Plan models exist
- Stripe integration started

- Usage limit checks in TenantModel

Priority

CRITICAL - Required for revenue

Support Required

- Stripe webhook completion
- Invoice generation
- Payment failure handling

Impact

HIGH - Revenue generation, feature gating

Tasks

Task 3.1: Subscription Management - [ ] 3.1.1 Plan creation and pricing - [ ] 3.1.2 Subscription creation via Stripe - [ ] 3.1.3 Plan upgrade/downgrade - [ ] 3.1.4 Trial period management - [ ] 3.1.5 Subscription renewal automation - [ ] 3.1.6 Cancellation and refund flow

Task 3.2: Usage Limits - [x] 3.2.1 Check limits on Lead creation - [x] 3.2.2 Check limits on User creation - [ ] 3.2.3 Storage limits tracking - [ ] 3.2.4 API call limits - [ ] 3.2.5 Overage handling (hard stop vs warning)

Task 3.3: Invoicing - [ ] 3.3.1 Invoice generation - [ ] 3.3.2 Invoice PDF creation - [ ] 3.3.3 Invoice email delivery - [ ] 3.3.4 Invoice history UI - [ ] 3.3.5 Tax calculation integration

Collaboration Dependencies: - **Provides to**: core (user limits), leads (lead limits), tenants (plan features) - **Depends on**: tenants (billing per tenant) - **Integrated with**: automation (trial expiry notifications)

See full document for remaining 23 modules with complete TIPSI scoring, task breakdown, and CRM best practices.

DEPLOYMENT READINESS SUMMARY

Critical Path

1. **Security** (core, tenants, billing) - 23-25 TIPSI
2. **Database & Performance** - 24 TIPSI
3. **Testing** - 22 TIPSI
4. **Automation & Events** - 22 TIPSI
5. **Billing & Limits** - 21 TIPSI

Total Tasks: 450+
Timeline: 16 sprints (32 weeks)
Current Readiness: ~40%, Target 95%+