Product Requirements Document (PRD)

DeelFlowAI Platform - Comprehensive Testing Specification

Document Version: 1.0
Last Updated: October 11, 2025
Prepared For: TestSprite QA Team
Product: DeelFlowAI - Real Estate Investment & Lead Management Platform

---

📋 Table of Contents

---

1. Executive Summary

1.1 Product Purpose

DeelFlowAI is a comprehensive real estate investment platform that combines property management, lead generation, campaign automation, deal tracking, and AI-powered analytics to streamline real estate investment operations.

1.2 Testing Scope

This PRD covers end-to-end testing requirements for all platform modules including:

- Property Management System
- Lead Management & Scoring
- Marketing Campaign Management
- Deal & Milestone Tracking
- User & Organization Management
- Tenant Management
- Role-Based Access Control (RBAC)
- Dashboard Analytics
- Payment Processing
- AI-Powered Features

1.3 Target Users

- Real Estate Investors
- Property Wholesalers
- Real Estate Agents
- Investment Firms
- Property Management Companies

---

2. Product Overview

2.1 Core Value Proposition

DeelFlowAI automates and optimizes the entire real estate investment workflow from lead generation to deal closure, leveraging AI for property analysis, lead scoring, and personalized marketing campaigns.

2.2 Key Features

- **Property Management**: Comprehensive property tracking with AI valuation
- **Lead Management**: Automated lead scoring and qualification
- **Campaign Automation**: Multi-channel marketing with AI personalization
- **Deal Pipeline**: Complete deal tracking with milestone management
- **Analytics Dashboard**: Real-time insights and performance metrics
- **Payment Processing**: Integrated subscription and payment management
- **Multi-Tenancy**: Organization and user hierarchy support

---

3. Technical Environment

3.1 System Architecture

```
Frontend: React.js (Web Application)
Backend: FastAPI + Django ORM
Base URL: http://dev.deelflowai.com:8140
API Prefix: /api
```

3.2 Technology Stack

- **Frontend Framework**: React.js
- **Backend Framework**: FastAPI
- **Database**: PostgreSQL (via Django ORM)
- **Authentication**: JWT-based authentication
- **API Style**: RESTful
- **Data Format**: JSON

3.3 Environment Details

- **Development Environment**: http://dev.deelflowai.com:8140
- **API Base**: http://dev.deelflowai.com:8140/api
- **Protocol**: HTTP/HTTPS
- **API Version**: Current (no versioning in URL)

---

4. Module Specifications

4.1 Property Management Module

4.1.1 Purpose

Manage real estate property inventory with detailed information, financial analysis, and AI-powered valuation.

4.1.2 Data Model (20 Fields)

javascript

```javascript
{
  // Address Information
  street_address: String (required),
  unit_apt: String (optional),
  city: String (required),
  state: String (required),
  zip_code: String (required),
  county: String (optional),

  // Property Details
  property_type: String (required),
  bedrooms: String (required),
  bathrooms: String (required),
  square_feet: String (required),
  lot_size: String (optional),
  year_built: String (optional),

  // Financial Information
  purchase_price: String (required),
  arv: String (After Repair Value),
  repair_estimate: String (optional),
  holding_costs: String (optional),
  transaction_type: String (required),
  assignment_fee: String (optional),

  // Additional Information
  property_description: String (optional),
  seller_notes: String (optional)
}
```

4.1.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/properties/	List all properties (with pagination)	Yes
GET	/api/properties/{id}/	Get single property details	Yes
POST	/api/properties/	Create new property	Yes
PUT	/api/properties/{id}/	Update property	Yes
DELETE	/api/properties/{id}/	Delete property	Yes
GET	/api/properties/{id}/ai-analysis/	Get AI property analysis	Yes

4.1.4 Business Rules

- All prices must be positive numbers
- ZIP code must be valid US format (5 or 9 digits)
- Square feet and lot size must be positive
- Year built cannot be in the future
- Transaction type options: wholesale, retail, rental
- Property type options: single_family, multi_family, condo, townhouse, land

---

4.2 Lead Management Module

4.2.1 Purpose

Capture, qualify, and nurture potential property sellers and buyers through automated scoring and tracking.

4.2.2 Data Model (16 Fields)

javascript

```javascript
{
  // Contact Information
  first_name: String (required),
  last_name: String (required),
  email: String (required, unique),
  phone: String (required),

  // Property Information
  property_address: String (required),
  property_city: String (required),
  property_state: String (required),
  property_zip: String (required),
  property_type: String (default: "single_family"),

  // Lead Details
  source: String (optional),
  estimated_value: String (optional),
  mortgage_balance: String (optional),
  asking_price: String (optional),

  // Management
  preferred_contact_method: String (optional),
  lead_type: String (optional),
  status: String (default: "new")
}
```

4.2.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/leads/	List all leads (with filters)	Yes
GET	/api/leads/{id}/	Get single lead details	Yes
POST	/api/leads/	Create new lead	Yes
PUT	/api/leads/{id}/	Update lead	Yes
DELETE	/api/leads/{id}/	Delete lead	Yes
GET	/api/leads/{id}/ai-score/	Get AI lead score	Yes

4.2.4 Business Rules

- Email must be valid format and unique
- Phone must be valid US format
- Status options: new, contacted, qualified, unqualified, converted, lost
- Lead type options: seller, buyer, both
- Contact method options: email, phone, text, any
- AI scoring range: 0-100

4.3 Campaign Management Module

4.3.1 Purpose

Create and manage multi-channel marketing campaigns with AI-powered personalization for targeted lead generation.

4.3.2 Data Model (37 Fields)

javascript

```
{
  // Basic Campaign Information
  name: String (required),
  campaign_type: String (required),
  channel: Array<String> (required, frontend sends string),
  budget: Number (default: 1000.0),
  scheduled_at: String (ISO datetime),
  subject_line: String (required for email),
  email_content: String (required for email),
  use_ai_personalization: Boolean (default: true),
  status: String (optional),

  // Geographic Scope
  geographic_scope_type: String (optional),
  geographic_scope_values: String (optional),

  // Property Filters
  location: String (optional),
  property_type: String (optional),
  minimum_equity: Number (default: 0),
  min_price: Number (default: 250000),
  max_price: Number (default: 750000),
  distress_indicators: Array<String> (default: []),

  // Buyer Finder - Demographics
  last_qualification: String (optional),
  age_range: String (optional),
  ethnicity: String (optional),
  salary_range: String (optional),
  marital_status: String (optional),
  employment_status: String (optional),
  home_ownership_status: String (optional),

  // Buyer Finder - Geography
  buyer_country: String (optional),
  buyer_state: String (optional),
  buyer_counties: String (optional),
  buyer_city: String (optional),
  buyer_districts: String (optional),
  buyer_parish: String (optional),

  // Seller Finder - Geography
  seller_country: String (optional),
  seller_state: String (optional),
  seller_counties: String (optional),
  seller_city: String (optional),
  seller_districts: String (optional),
  seller_parish: String (optional),

  // Seller Finder - Additional
  property_year_built_min: String (optional),
  property_year_built_max: String (optional),
  seller_keywords: String (optional)
}
```

4.3.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/campaigns/	List all campaigns	Yes
GET	/api/campaigns/{id}/	Get campaign details	Yes
POST	/api/create_campaign/	Create new campaign	Yes
PUT	/api/campaigns/{id}/	Update campaign	Yes
DELETE	/api/campaigns/{id}/	Delete campaign	Yes
GET	/api/campaigns/{id}/recipients/	Get campaign recipients	Yes
GET	/api/active_campaign_summary/	Get active campaigns summary	Yes
GET	/api/campaign_property_stats/	Get campaign property stats	Yes
GET	/api/campaign_performance_overview/	Get performance overview	Yes
GET	/api/channel_response_rates/	Get channel response rates	Yes
GET	/api/lead_conversion_funnel/	Get conversion funnel	Yes

4.3.4 Business Rules

- Campaign name must be unique per organization
- Channel options: email, sms, direct_mail, phone, social_media
- Status options: draft, scheduled, active, paused, completed, cancelled

- Budget must be positive
- Scheduled date cannot be in the past
- Email campaigns require subject_line and email_content
- Geographic scope type options: national, state, county, city, zip_code
- Distress indicators: foreclosure, tax_lien, pre_foreclosure, probate, divorce

---

4.4 Deal Management Module

4.4.1 Purpose

Track real estate transactions from contract to closing with milestone management and financial tracking.

4.4.2 Data Model (17 Fields)

javascript

```javascript
{
  // Core References
  property_id: String (required),
  lead_id: String (required),
  deal_type: String (required),

  // Parties
  buyer_id: String (optional),
  seller_id: String (optional),

  // Timeline
  inspection_period: String (optional),
  contract_date: String (required),
  closing_date: String (required),

  // Financial
  purchase_price: String (required),
  sale_price: String (optional),
  assignment_fee: String (optional),
  earnest_money: String (optional),

  // Contingencies
  financing_contingency: Boolean (default: false),
  inspection_contingency: Boolean (default: false),
  appraisal_contingency: Boolean (default: false),
  title_contingency: Boolean (default: false),

  // Additional
  notes: String (optional)
}
```

4.4.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/deals/	List all deals	Yes
GET	/api/deals/{id}/	Get deal details	Yes
POST	/api/deals/	Create new deal	Yes
PUT	/api/deals/{id}/	Update deal	Yes
DELETE	/api/deals/{id}/	Delete deal	Yes
GET	/api/deals/{id}/milestones/	Get deal milestones	Yes

4.4.4 Business Rules

- Deal type options: wholesale, assignment, double_close, retail
- Closing date must be after contract date
- Purchase price must be positive
- Property and lead must exist and belong to organization
- Status progression: pending → under_contract → in_progress → closed → cancelled

---

4.5 Deal Milestones Module

4.5.1 Purpose

Track critical milestones and tasks within a deal lifecycle to ensure timely completion.

4.5.2 Data Model (6 Fields)

javascript

```
{
  deal_id: String (required),
  milestone_type: String (required),
  title: String (required),
  description: String (optional),
  due_date: String (required, ISO datetime),
  is_critical: Boolean (default: false)
}
```

4.5.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/deal-milestones/	List all milestones	Yes
GET	/api/deal-milestones/{id}/	Get milestone details	Yes
POST	/api/deal-milestones/	Create milestone	Yes
PUT	/api/deal-milestones/{id}/	Update milestone	Yes
DELETE	/api/deal-milestones/{id}/	Delete milestone	Yes
PATCH	/api/deal-milestones/{id}/complete/	Mark as complete	Yes

4.5.4 Business Rules

- Milestone type options: contract_signed, inspection_scheduled, inspection_completed, financing_approved, appraisal_ordered, appraisal_completed, title_search, closing_scheduled, closing_completed
- Due date cannot be in the past for new milestones
- Critical milestones cannot be deleted if deal is active

4.6 User Management Module

4.6.1 Purpose

Manage user accounts, permissions, and authentication within the multi-tenant system.

4.6.2 Data Model (14 Fields)

javascript

```javascript
{
  // Basic Information
  first_name: String (required),
  last_name: String (required),
  email: String (required, unique),
  phone: String (optional),

  // Organization
  role: String (required),
  status: String (default: "active"),
  organization_id: String (required),
  department: String (optional),
  position: String (optional),

  // Settings
  avatar: String (URL, optional),
  timezone: String (optional),
  language: String (optional),
  notifications_enabled: Boolean (default: true),
  two_factor_enabled: Boolean (default: false)
}
```

4.6.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
POST	/api/create-user/	Register new user	No
POST	/api/auth/login	User login	No
POST	/api/logout/	User logout	Yes
GET	/api/user/	Get current user	Yes
GET	/api/users/	Get all users	Yes (Admin)
POST	/api/invitations/	Send user invitation	Yes (Admin)
GET	/api/validate-invitation/	Validate invitation token	No
POST	/api/invitee-register/	Register via invitation	No

4.6.4 Business Rules

- Email must be unique across system
- Password minimum 8 characters with complexity requirements
- Status options: active, inactive, suspended, pending
- Only admins can invite users
- Users can only access their organization's data

4.7 Organization Management Module

4.7.1 Purpose

Manage organization settings, configuration, and multi-tenant isolation.

4.7.2 Data Model (14 Fields)

javascript

```javascript
{
  // Basic Information
  name: String (required, unique),
  industry: String (optional),
  organization_size: String (optional),

  // Contact Information
  business_email: String (required),
  business_phone: String (optional),
  website: String (optional),
  support_email: String (optional),

  // Address
  street_address: String (optional),
  city: String (optional),
  state_province: String (optional),
  zip_postal_code: String (optional),
  country: String (optional),

  // Settings
  timezone: String (default: "UTC"),
  language: String (default: "en")
}
```

4.7.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/organization/	Get organization details	Yes
POST	/api/organization/	Create organization	Yes (System)
PUT	/api/organization/{id}/	Update organization	Yes (Admin)

4.8 Tenant Management Module

4.8.1 Purpose

Manage tenant subscriptions, features, and billing for SaaS model.

4.8.2 Data Model (15 Fields)

javascript

```
{
  // Basic Information
  name: String (required),
  email: String (required, unique),
  organization_name: String (required),

  // Subscription
  plan_type: String (required),
  status: String (default: "active"),
  subscription_start_date: String (required),
  subscription_end_date: String (optional),
  max_users: String (required),
  features: Array<String> (default: []),

  // Billing
  billing_address: String (optional),
  contact_person: String (required),
  phone: String (optional),

  // Additional
  website: String (optional),
  industry: String (optional),
  organization_size: String (optional)
}
```

4.8.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/tenants/	List all tenants	Yes (SuperAdmin)
GET	/api/tenants/{id}/	Get tenant details	Yes (SuperAdmin)
POST	/api/tenants/	Create tenant	Yes (SuperAdmin)
PUT	/api/tenants/{id}/	Update tenant	Yes (SuperAdmin)
DELETE	/api/tenants/{id}/	Delete tenant	Yes (SuperAdmin)

4.8.4 Business Rules

- Plan type options: free, starter, professional, enterprise
- Status options: active, trial, suspended, cancelled, expired
- Features array includes: properties, leads, campaigns, deals, ai_analysis, api_access

---

4.9 Role Management Module

4.9.1 Purpose

Manage role-based access control (RBAC) with granular permissions.

4.9.2 Data Model (5 Fields)

javascript

```javascript
{
  name: String (required),
  description: String (optional),
  permissions: Array<String> (required),
  is_active: Boolean (default: true),
  organization_id: String (required)
}
```

4.9.3 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/roles/	List all roles	Yes (Admin)
GET	/api/roles/{id}/	Get role details	Yes (Admin)
POST	/api/roles/	Create role	Yes (Admin)
PUT	/api/roles/{id}/	Update role	Yes (Admin)
DELETE	/api/roles/{id}/	Delete role	Yes (Admin)
PATCH	/api/roles/{id}/permissions/	Update permissions	Yes (Admin)

4.9.4 Business Rules

- Default roles: super_admin, admin, manager, agent, viewer
- Permissions format: module.action (e.g., properties.create, leads.view)
- Cannot delete role if assigned to users
- Super admin role cannot be deleted or modified

4.10 Dashboard & Analytics Module

4.10.1 Purpose

Provide real-time insights, metrics, and performance analytics across all modules.

4.10.2 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/total-revenue/	Get total revenue	Yes
GET	/api/active-users/	Get active users count	Yes
GET	/api/properties-listed/	Get properties count	Yes
GET	/api/ai-conversations/	Get AI conversation count	Yes
GET	/api/total-deals/	Get deals count	Yes
GET	/api/monthly-profit/	Get monthly profit	Yes
GET	/api/voice-calls-count/	Get voice calls count	Yes
GET	/api/compliance-status/	Get compliance status	Yes
GET	/api/live-activity-feed/	Get activity feed	Yes
GET	/api/deal-completions-scheduling/	Get deal schedules	Yes
GET	/api/revenue-user-growth-chart-data/	Get growth chart data	Yes
GET	/api/voice-ai-calls-count/	Get AI call metrics	Yes
GET	/api/vision-analysis/	Get vision analysis metrics	Yes
GET	/api/nlp-processing/	Get NLP metrics	Yes
GET	/api/blockchain-txns/	Get blockchain metrics	Yes
GET	/api/tenant-management/stats/	Get tenant stats	Yes (Admin)
POST	/api/recent_activity/	Get recent activity	Yes
GET	/api/analytics/opportunity-cost-analysis/	Get cost analysis	Yes
GET	/api/ai-metrics/overall-accuracy/	Get AI accuracy	Yes
GET	/api/compliance-status/details/	Get compliance details	Yes
GET	/api/market-alerts/recent/	Get market alerts	Yes

4.11 Property Save Module

4.11.1 Purpose

Allow users to save/bookmark properties for later review.

4.11.2 API Endpoints

Method	Endpoint	Purpose	Auth Required
GET	/api/property-saves/	List saved properties	Yes
POST	/api/property-saves/	Save a property	Yes
DELETE	/api/property-saves/{id}/	Unsave property	Yes

4.12 Payment Processing Module

4.12.1 Purpose

Handle subscription payments and payment intent processing.

4.12.2 API Endpoints

Method	Endpoint	Purpose	Auth Required
POST	/api/create-payment-intent/	Create payment intent	Yes
POST	/api/confirm-payment/	Confirm payment	Yes
GET	/api/current-subscription/	Get current subscription	Yes

---

5. API Testing Requirements

5.1 General API Testing

5.1.1 HTTP Methods Testing

Test all CRUD operations for each module:

- **GET**: Retrieve resources (list and detail views)
- **POST**: Create new resources
- **PUT**: Full update of resources
- **PATCH**: Partial update of resources
- **DELETE**: Remove resources

5.1.2 Status Code Validation

Status Code	When to Expect
200 OK	Successful GET, PUT, PATCH
201 Created	Successful POST
204 No Content	Successful DELETE
400 Bad Request	Invalid data format or validation errors
401 Unauthorized	Missing or invalid authentication
403 Forbidden	Insufficient permissions
404 Not Found	Resource doesn't exist
409 Conflict	Duplicate resource (e.g., email)
422 Unprocessable Entity	Valid format but business rule violation
500 Internal Server Error	Server-side error

5.1.3 Response Format Testing

All successful responses should return JSON with:

json

```json
{
  "data": {...},
  "message": "Success message",
  "status": "success"
}
```

Error responses should return:

json

```json
{
  "error": "Error message",
  "details": {...},
  "status": "error"
}
```

5.2 Authentication Testing

5.2.1 Login Flow

```
POST /api/auth/login
Body: {
  "email": "user@example.com",
  "password": "SecurePassword123"
}

Expected Response:
{
  "token": "jwt_token_here",
  "user": {
    "id": "user_id",
    "email": "user@example.com",
    "role": "admin"
  }
}
```

5.2.2 Token Usage

- Include JWT token in Authorization header: `Bearer {token}`
- Test token expiration handling
- Test refresh token mechanism (if implemented)

5.2.3 Security Tests

- Login with invalid credentials
- Access protected endpoints without token
- Access with expired token
- Access with malformed token
- Test rate limiting on login endpoint

5.3 Pagination Testing

Test list endpoints with pagination:

```
GET /api/properties/?page=1&limit=20
GET /api/leads/?page=2&limit=50
```

Verify response includes:

- `total`: Total number of items
- `page`: Current page
- `limit`: Items per page
- `pages`: Total pages
- `data`: Array of items

5.4 Filtering & Searching

Test filtering capabilities:

```
GET /api/properties/?city=Austin&property_type=single_family
GET /api/leads/?status=new&source=website
GET /api/campaigns/?status=active&campaign_type=buyer_finder
```

5.5 Sorting

Test sorting parameters:

```
GET /api/properties/?sort=purchase_price&order=desc
GET /api/leads/?sort=created_at&order=asc
```

---

6. Functional Requirements

6.1 Property Management

Test Cases:

1. **Create Property**
   - Create with all required fields
   - Create with optional fields
   - Validate field constraints
   - Test duplicate address handling
2. **Update Property**
   - Full update (PUT)
   - Partial update (PATCH)
   - Update non-existent property
   - Update with invalid data
3. **Delete Property**
   - Delete existing property
   - Delete non-existent property
   - Verify cascade deletion (if applicable)
   - Test soft delete vs hard delete
4. **List Properties**
   - List with pagination
   - Filter by city, state, property_type
   - Sort by price, date
   - Search by address
5. **AI Analysis**
   - Request AI analysis for property
   - Verify analysis response format
   - Test with incomplete property data

6.2 Lead Management

Test Cases:

1. **Create Lead**
   - Create with valid email
   - Create with duplicate email
   - Create with invalid phone format
   - Verify auto-status assignment
2. **Update Lead Status**
   - Progress through status workflow
   - Invalid status transitions
   - Update multiple fields simultaneously
3. **AI Lead Scoring**
   - Request score for qualified lead
   - Request score for unqualified lead
   - Verify score range (0-100)
4. **Lead Filtering**
   - Filter by status
   - Filter by source
   - Filter by property location
   - Combined filters

6.3 Campaign Management

Test Cases:

1. **Create Campaign**
   - Create email campaign with required fields
   - Create SMS campaign
   - Create multi-channel campaign
   - Test channel format (string vs array)
2. **Geographic Targeting**
   - Test buyer finder with demographics
   - Test seller finder with geography

    - Combine multiple filters
    - Validate filter combinations
3. **Campaign Scheduling**
    - Schedule future campaign
    - Schedule past date (should fail)
    - Update scheduled campaign
    - Cancel scheduled campaign
4. **Campaign Analytics**
    - Get active campaigns summary
    - Get performance metrics
    - Get recipient list
    - Test conversion funnel

6.4 Deal Management

Test Cases:

1. **Create Deal**
    - Create with valid property and lead
    - Create with non-existent property
    - Create with invalid dates
    - Test contingency flags
2. **Deal Milestones**
    - Create milestone for deal
    - Mark milestone complete
    - Delete milestone
    - Test critical milestone handling
3. **Deal Progression**
    - Update deal status
    - Add financial information
    - Complete deal
    - Cancel deal
4. **Deal Reporting**
    - Calculate profit margins
    - Track days in pipeline
    - Generate deal summary

---

7. Non-Functional Requirements

7.1 Performance Requirements

Metric	Requirement	Priority
API Response Time	< 500ms for simple queries	High
API Response Time	< 2s for complex queries	High
List Endpoint	< 1s for 100 items	Medium
Database Query	< 200ms average	High
Concurrent Users	100+ simultaneous	Medium
File Upload	< 5s for 10MB	Low

7.2 Scalability Requirements

- Support 10,000+ properties per organization
- Support 50,000+ leads per organization
- Handle 1,000+ concurrent API requests
- Database should handle 1M+ total records

7.3 Reliability Requirements

- 99.5% uptime
- Automated error recovery
- Data backup every 24 hours
- Transaction rollback on failure

7.4 Security Requirements

- HTTPS for all communications
- JWT token expiration: 24 hours
- Password hashing: bcrypt or similar
- SQL injection prevention
- XSS protection
- CSRF protection
- Rate limiting: 100 requests/minute per user

7.5 Data Integrity Requirements

- Foreign key constraints enforced
- Cascade delete rules defined
- Audit trail for critical actions
- Data validation at API level
- Transaction support for multi-step operations

---

8. Test Scenarios & Use Cases

8.1 End-to-End User Workflows

Workflow 1: Property Acquisition Process

```
1. User logs in
2. Creates new property listing
3. Property gets AI analysis
4. Creates campaign targeting buyers
5. Receives lead responses
6. Qualifies leads with AI scoring
7. Creates deal with qualified lead
8. Tracks deal through milestones
9. Closes deal
10. Views profit analytics
```

Workflow 2: Lead Generation Campaign

```
1. User creates buyer finder campaign
2. Sets demographic filters
3. Sets geographic targeting
4. Schedules campaign for future date
5. Campaign auto-launches at scheduled time
6. System tracks responses
7. Auto-creates leads from responses
8. AI scores incoming leads
9. User reviews high-score leads
10. Converts leads to deals
```

Workflow 3: Organization Setup

```
1. Super admin creates new tenant
2. Sets subscription plan
3. Creates organization
4. Invites admin user
5. Admin accepts invitation
6. Admin creates team members
7. Admin assigns roles
8. Admin configures organization settings
9. Team members log in
10. Start using platform
```

8.2 Edge Cases & Boundary Testing

Test Scenarios:

1. **Maximum Data Limits**
   - Create property with maximum field lengths
   - Upload maximum file sizes
   - Create campaign with 1000+ recipients
   - Pagination with 10,000+ records
2. **Minimum Data Limits**
   - Create records with only required fields
   - Zero values in numeric fields
   - Empty arrays and null values
   - Minimum string lengths
3. **Concurrent Operations**
   - Multiple users updating same property
   - Simultaneous campaign creation
   - Race conditions in deal status updates
   - Lock handling for critical operations
4. **Data Type Validation**
   - String inputs in numeric fields
   - Invalid date formats
   - Invalid email formats
   - Special characters in text fields
   - SQL injection attempts
   - XSS attack patterns
5. **Permission Boundaries**
   - User accessing another org's data
   - Viewer role attempting create/update
   - Deleted user token usage
   - Suspended account access

---

9. Data Validation Rules

9.1 Property Validation

Field	Validation Rule	Error Message
street_address	Required, max 255 chars	"Street address is required"
city	Required, max 100 chars	"City is required"
state	Required, 2 chars uppercase	"Valid state code required"
zip_code	Required, format: 12345 or 12345-6789	"Valid ZIP code required"
property_type	Required, enum values	"Invalid property type"
bedrooms	Numeric string, 0-50	"Bedrooms must be 0-50"
bathrooms	Numeric string, 0-50	"Bathrooms must be 0-50"
square_feet	Positive number	"Square feet must be positive"
year_built	1800-current year	"Invalid year built"
purchase_price	Positive number	"Purchase price must be positive"
arv	Positive number	"ARV must be positive"

9.2 Lead Validation

Field	Validation Rule	Error Message
first_name	Required, max 100 chars	"First name is required"
last_name	Required, max 100 chars	"Last name is required"
email	Required, valid email, unique	"Valid unique email required"
phone	Required, format: (XXX) XXX-XXXX	"Valid phone number required"
property_address	Required	"Property address is required"
property_zip	Format: 12345 or 12345-6789	"Valid ZIP code required"
status	Enum: new, contacted, qualified, etc.	"Invalid status"
estimated_value	Positive number or empty	"Must be positive number"

9.3 Campaign Validation

Field	Validation Rule	Error Message
name	Required, unique per org, max 200	"Campaign name is required and must be unique"
campaign_type	Required, enum values	"Invalid campaign type"
channel	Required, array of valid channels	"At least one valid channel required"
budget	Positive number, min 100	"Budget must be at least $100"
scheduled_at	Future datetime or empty	"Schedule date must be in future"
subject_line	Required for email, max 200	"Email subject required"
email_content	Required for email	"Email content required"
min_price	Less than max_price	"Min price must be less than max"
max_price	Greater than min_price	"Max price must be greater than min"

9.4 Deal Validation

Field	Validation Rule	Error Message
property_id	Required, valid UUID, exists	"Valid property required"
lead_id	Required, valid UUID, exists	"Valid lead required"
deal_type	Required, enum values	"Invalid deal type"
contract_date	Required, valid date	"Contract date required"
closing_date	Required, after contract_date	"Closing must be after contract"
purchase_price	Positive number	"Purchase price must be positive"
sale_price	Greater than purchase_price	"Sale price must exceed purchase"

9.5 User Validation

Field	Validation Rule	Error Message
email	Required, valid email, unique	"Valid unique email required"
password	Min 8 chars, uppercase, lowercase, number	"Password too weak"
first_name	Required, max 100 chars	"First name required"
last_name	Required, max 100 chars	"Last name required"
role	Required, valid role	"Valid role required"
organization_id	Required, valid UUID	"Valid organization required"

---

10. Integration Points

10.1 Frontend-Backend Integration

Request Headers

javascript

```javascript
{
  "Content-Type": "application/json",
  "Authorization": "Bearer {jwt_token}",
  "X-Organization-ID": "{org_id}" // Optional, for multi-tenant
}
```

Response Headers

javascript

```
{
  "Content-Type": "application/json",
  "X-RateLimit-Limit": "100",
  "X-RateLimit-Remaining": "95",
  "X-RateLimit-Reset": "1697040000"
}
```

10.2 Third-Party Integrations

AI Services

- **Property Analysis**: GET /api/properties/{id}/ai-analysis/
- **Lead Scoring**: GET /api/leads/{id}/ai-score/
- **AI Personalization**: Used in campaign content generation

Payment Processing

- **Payment Intent**: POST /api/create-payment-intent/
- **Payment Confirmation**: POST /api/confirm-payment/
- **Subscription Management**: GET /api/current-subscription/

Email/SMS Services

- Campaign delivery through /api/create_campaign/
- Template rendering with AI personalization
- Delivery tracking and analytics

10.3 Database Integration

Django ORM with FastAPI

- All database operations use Django ORM
- Async wrappers (sync_to_async) for FastAPI compatibility
- Transaction management for multi-step operations
- Connection pooling for performance

---

11. Security & Authentication

11.1 Authentication Flow

mermaid

```
sequenceDiagram
    User->>Frontend: Enter credentials
    Frontend->>Backend: POST /api/auth/login
    Backend->>Database: Verify credentials
    Database->>Backend: User found
    Backend->>Backend: Generate JWT
    Backend->>Frontend: Return JWT + user data
    Frontend->>Frontend: Store JWT
    Frontend->>Backend: API request with JWT
    Backend->>Backend: Validate JWT
    Backend->>Frontend: Return data
```

11.2 Authorization Levels

Role	Permissions	Description
super_admin	Full system access	Manage all tenants and organizations
admin	Organization full access	Manage organization, users, all modules
manager	Team management	Create/edit properties, leads, campaigns, deals
agent	Standard access	View and create records, limited editing
viewer	Read-only	View-only access to assigned data

11.3 Permission Matrix

Module	super_admin	admin	manager	agent	viewer
Properties	CRUD	CRUD	CRUD	CRU	R
Leads	CRUD	CRUD	CRUD	CRU	R
Campaigns	CRUD	CRUD	CRUD	CR	R
Deals	CRUD	CRUD	CRUD	CRU	R
Users	CRUD	CRUD (org only)	R	R	-
Organization	CRUD	RU (own)	R	-	-
Tenants	CRUD	-	-	-	-
Roles	CRUD	CRUD (org only)	R	-	-

Legend: C=Create, R=Read, U=Update, D=Delete, -=No Access

11.4 Security Test Cases

1. **Authentication Tests**
   - Login with valid credentials
   - Login with invalid credentials
   - Login with suspended account
   - Token expiration handling
   - Logout functionality
2. **Authorization Tests**
   - Access resources within permission
   - Access resources outside permission
   - Cross-organization data access attempt
   - Role-based endpoint access
   - Admin-only endpoint protection
3. **Data Protection Tests**
   - SQL injection attempts
   - XSS attack vectors
   - CSRF token validation
   - Password hashing verification
   - Sensitive data masking in logs
4. **Rate Limiting Tests**
   - Exceed rate limit threshold
   - Rate limit reset timing
   - Rate limit per endpoint
   - Rate limit bypass attempts

---

12. Error Handling

12.1 Standard Error Response Format

json

```json
{
  "status": "error",
  "error_code": "VALIDATION_ERROR",
  "message": "Validation failed",
  "details": {
    "field_name": ["Error message for field"],
    "another_field": ["Another error message"]
  },
  "timestamp": "2025-10-11T10:30:00Z",
  "request_id": "uuid-here"
}
```

12.2 Error Code Catalog

Error Code	HTTP Status	Description
VALIDATION_ERROR	400	Input validation failed
AUTHENTICATION_REQUIRED	401	No valid authentication
INVALID_TOKEN	401	JWT token invalid or expired
FORBIDDEN	403	Insufficient permissions
NOT_FOUND	404	Resource not found
DUPLICATE_ENTRY	409	Resource already exists
BUSINESS_RULE_VIOLATION	422	Business logic constraint
RATE_LIMIT_EXCEEDED	429	Too many requests
INTERNAL_ERROR	500	Server error
SERVICE_UNAVAILABLE	503	Service temporarily down

12.3 Error Handling Test Cases

1. **Validation Errors**
   - Missing required fields
   - Invalid field formats
   - Out-of-range values
   - Type mismatches
2. **Authentication Errors**
   - Missing token
   - Expired token
   - Invalid token format
   - Revoked token
3. **Authorization Errors**
   - Insufficient permissions
   - Cross-tenant access
   - Deleted resource access
   - Invalid resource ownership
4. **Business Logic Errors**
   - Duplicate email registration
   - Invalid status transitions
   - Constraint violations
   - Orphaned record creation
5. **System Errors**
   - Database connection failure
   - Third-party service timeout
   - File system errors
   - Memory exhaustion

13. Performance Benchmarks

13.1 Response Time Targets

Endpoint Type	Target	Maximum
Simple GET	100ms	500ms
List with pagination	300ms	1000ms
Complex query	500ms	2000ms
POST/PUT	200ms	800ms
DELETE	100ms	500ms
AI Analysis	2000ms	5000ms
File Upload	1000ms	5000ms

13.2 Load Testing Scenarios

Scenario 1: Normal Load

- Concurrent Users: 50
- Duration: 30 minutes
- Success Rate: > 99%
- Avg Response Time: < 500ms

Scenario 2: Peak Load

- Concurrent Users: 200
- Duration: 15 minutes
- Success Rate: > 95%
- Avg Response Time: < 1000ms

Scenario 3: Stress Test

- Concurrent Users: 500
- Duration: 5 minutes
- Success Rate: > 90%
- Identify breaking point

13.3 Database Performance

- Query execution time: < 200ms (95th percentile)
- Connection pool: 20-50 connections
- Index usage: All foreign keys indexed
- Query optimization: N+1 queries eliminated

13.4 API Performance Metrics

Test and monitor:

- Throughput (requests per second)
- Error rate (%)
- Response time distribution (p50, p95, p99)
- Resource utilization (CPU, memory, database)

14. Acceptance Criteria

14.1 Functional Acceptance

✅ All CRUD operations work for each module

- Properties: Create, Read, Update, Delete
- Leads: Create, Read, Update, Delete
- Campaigns: Create, Read, Update, Delete
- Deals: Create, Read, Update, Delete
- Users: Create, Read, Update, Delete
- Organizations: Create, Read, Update
- Tenants: Create, Read, Update, Delete
- Roles: Create, Read, Update, Delete

✅ Authentication & Authorization

- User registration works
- Login returns valid JWT
- JWT authentication on protected endpoints
- Role-based access control enforced
- Multi-tenant isolation verified

✅ Data Validation

- All required fields validated
- Field format validation works
- Business rule constraints enforced
- Appropriate error messages returned

✅ API Functionality

- Pagination works correctly
- Filtering returns accurate results
- Sorting functions properly
- Search capabilities operational

✅ **Special Features**

- AI property analysis returns results
- AI lead scoring functions
- Campaign scheduling works
- Deal milestone tracking operational
- Dashboard analytics load correctly

14.2 Non-Functional Acceptance

✅ **Performance**

- 95% of requests complete within target time
- System handles 100 concurrent users
- Database queries optimized
- No memory leaks detected

✅ **Security**

- All endpoints require authentication
- Authorization properly enforced
- SQL injection tests pass
- XSS prevention verified
- Password hashing implemented
- Rate limiting functional

✅ **Reliability**

- Error handling comprehensive
- Transaction rollback on failure
- Data integrity maintained
- Graceful degradation implemented

✅ **Usability**

- Error messages clear and helpful
- API responses consistent
- Documentation accurate
- Response formats standardized

14.3 Integration Acceptance

✅ **Frontend-Backend Integration**

- All frontend forms submit successfully
- Data formats match between systems
- Error handling properly displayed
- File uploads work correctly

✅ **Third-Party Integration**

- Payment processing functional
- Email/SMS delivery operational
- AI services responding
- External APIs accessible

14.4 Documentation Acceptance

✅ **API Documentation**

- All endpoints documented
- Request/response examples provided
- Error codes cataloged
- Authentication flow explained

✅ **Test Documentation**

- Test cases documented
- Test results recorded
- Bug reports formatted properly
- Coverage reports generated

---

15. Test Deliverables

15.1 Required Test Reports

1. **Functional Test Report**
    - Test cases executed
    - Pass/fail status
    - Defects found
    - Screenshots/evidence
2. **API Test Report**
    - Endpoint coverage
    - Status code validation
    - Response time metrics
    - Payload validation results
3. **Security Test Report**
    - Authentication tests
    - Authorization tests
    - Vulnerability scan results
    - Penetration test findings
4. **Performance Test Report**
    - Load test results

    - Stress test results
    - Response time distribution
    - Resource utilization
5. **Integration Test Report**
    - End-to-end workflow results
    - Cross-module integration
    - Third-party integration status

15.2 Bug Reporting Format

markdown

```
**Bug ID**: BUG-001
**Title**: Clear, concise description
**Severity**: Critical / High / Medium / Low
**Priority**: P0 / P1 / P2 / P3

**Module**: Property Management
**Endpoint**: POST /api/properties/
**Environment**: Development

**Steps to Reproduce**:
1. Step 1
2. Step 2
3. Step 3

**Expected Result**:
What should happen

**Actual Result**:
What actually happens

**Screenshots/Logs**:
Attached evidence

**Additional Info**:
- Browser/Tool: Postman v10
- User Role: Admin
- Request Payload: {...}
- Response: {...}
```

15.3 Test Coverage Requirements

- **Functional Coverage**: 100% of features
- **API Endpoint Coverage**: 100% of endpoints
- **Code Coverage**: Minimum 80%
- **Security Test Coverage**: All OWASP Top 10
- **Performance Test Coverage**: All critical paths

---

16. Testing Tools & Environment

16.1 Recommended Testing Tools

API Testing:

- Postman (automated collections)
- REST Assured
- SoapUI
- Insomnia

Load Testing:

- JMeter
- Gatling
- Locust
- K6

Security Testing:

- OWASP ZAP
- Burp Suite
- Nmap
- SQLMap

Automation:

- Selenium (if UI testing needed)
- PyTest
- Jest
- Cypress

16.2 Test Environment Details

```
Environment: Development
Base URL: http://dev.deelflowai.com:8140
API Base: http://dev.deelflowai.com:8140/api
Database: PostgreSQL
Authentication: JWT Bearer Token
```

16.3 Test Data Requirements

Test Users:

- Super Admin account
- Admin account
- Manager account
- Agent account
- Viewer account

Test Organizations:

- Minimum 3 test organizations
- Different subscription levels
- Various organization sizes

Test Data Sets:

- 100+ test properties
- 200+ test leads
- 50+ test campaigns
- 30+ test deals
- Variety of data scenarios

---

17. Risk Assessment

17.1 High-Risk Areas

1. **Multi-Tenant Data Isolation**
   - Risk: Cross-tenant data leakage
   - Mitigation: Extensive authorization testing
   - Priority: Critical
2. **Payment Processing**
   - Risk: Financial transaction errors
   - Mitigation: Thorough integration testing
   - Priority: Critical
3. **AI Service Integration**
   - Risk: Service unavailability affecting core features
   - Mitigation: Fallback mechanisms, timeout handling
   - Priority: High
4. **Data Migration**
   - Risk: Data loss or corruption during updates
   - Mitigation: Backup procedures, rollback plans
   - Priority: High
5. **Authentication System**
   - Risk: Unauthorized access
   - Mitigation: Security-focused testing, penetration testing
   - Priority: Critical

17.2 Medium-Risk Areas

1. **Campaign Delivery**
   - Risk: Failed email/SMS delivery
   - Mitigation: Retry mechanisms, delivery tracking
2. **File Uploads**
   - Risk: Corrupt or malicious files
   - Mitigation: File validation, virus scanning
3. **Performance Degradation**
   - Risk: Slow response times under load
   - Mitigation: Load testing, optimization

17.3 Testing Priorities

Phase 1 - Critical (Week 1-2):

- Authentication & Authorization
- Core CRUD operations
- Multi-tenant isolation
- Payment processing
- Security fundamentals

Phase 2 - High (Week 3-4):

- AI integrations
- Campaign management
- Deal workflows
- Dashboard analytics
- Performance testing

Phase 3 - Medium (Week 5):

- Edge cases
- Error handling
- UI integration
- Documentation validation
- Final regression

---

18. Glossary

Term	Definition
ARV	After Repair Value - estimated property value after repairs
Assignment Fee	Fee paid to wholesaler for assigning contract
Contingency	Condition that must be met for deal to proceed
Deal Pipeline	Stages a deal progresses through to closing
Earnest Money	Deposit showing buyer's good faith
Lead Scoring	AI-driven qualification rating for leads
Multi-Tenancy	System architecture supporting multiple organizations
RBAC	Role-Based Access Control
Wholesaling	Strategy of contracting property and assigning to buyer

---

19. Appendices

Appendix A: Sample Test Data

Sample Property

json

```json
{
  "street_address": "123 Main Street",
  "unit_apt": "Apt 4B",
  "city": "Austin",
  "state": "TX",
  "zip_code": "78701",
  "county": "Travis",
  "property_type": "single_family",
  "bedrooms": "3",
  "bathrooms": "2",
  "square_feet": "1500",
  "lot_size": "5000",
  "year_built": "2005",
  "purchase_price": "250000",
  "arv": "350000",
  "repair_estimate": "50000",
  "holding_costs": "5000",
  "transaction_type": "wholesale",
  "assignment_fee": "10000",
  "property_description": "Great investment opportunity",
  "seller_notes": "Motivated seller"
}
```

Sample Lead

json

```
{
  "first_name": "John",
  "last_name": "Doe",
  "email": "john.doe@example.com",
  "phone": "(512) 555-1234",
  "property_address": "456 Oak Avenue",
  "property_city": "Austin",
  "property_state": "TX",
  "property_zip": "78702",
  "property_type": "single_family",
  "source": "website",
  "estimated_value": "300000",
  "mortgage_balance": "200000",
  "asking_price": "280000",
  "preferred_contact_method": "email",
  "lead_type": "seller",
  "status": "new"
}
```

Appendix B: Postman Collection Structure

```
DeelFlowAI API Tests/
├── Authentication/
│   ├── Login
│   ├── Register
│   ├── Get Current User
│   └── Logout
├── Properties/
│   ├── Create Property
│   ├── List Properties
│   ├── Get Property
│   ├── Update Property
│   ├── Delete Property
│   └── Get AI Analysis
├── Leads/
│   ├── Create Lead
│   ├── List Leads
│   ├── Get Lead
│   ├── Update Lead
│   ├── Delete Lead
│   └── Get AI Score
├── Campaigns/
├── Deals/
├── Milestones/
├── Users/
├── Organizations/
└── Dashboard/
```

Appendix C: Contact Information

Project Stakeholders:

- Product Owner: [Name]
- Technical Lead: [Name]
- QA Lead: TestSprite Team
- DevOps: [Name]

Support Contacts:

- Technical Support: support@deelflowai.com
- Bug Reporting: bugs@deelflowai.com
- Documentation: docs@deelflowai.com

Document Control

Version	Date	Author	Changes
1.0	2025-10-11	Product Team	Initial PRD for TestSprite

Document Status: Final for Testing
Next Review Date: Upon completion of testing phase
Distribution: TestSprite QA Team, Development Team, Product Team

---

END OF DOCUMENT