Add Property Testing - Product Requirements Document (PRD)

📋 **Document Overview**
- **Document Type**: Product Requirements Document (PRD)
- **Feature**: Add Property Functionality Testing
- **Version**: 1.0
- **Date**: October 9, 2025
- **Status**: Ready for Testing

🎯 **Objective**
Test the complete Add Property workflow using the frontend interface to ensure data is properly inserted into the database through the backend API.

📝 **Feature Description**

Primary Goal
Validate that users can successfully add new properties through the frontend form and have the data correctly stored in the database via the backend API.

User Story
As a real estate investor, I want to add new properties to my pipeline through a web form so that I can track and manage my property investments effectively.

🔧 **Technical Requirements**

Frontend Components
- **Form Location**: `/app/properties/add`
- **Form Fields**: 20+ input fields including address, property details, financial information
- **Validation**: Client-side and server-side validation
- **API Integration**: POST request to `/properties/` endpoint

Backend API
- **Endpoint**: `POST /properties/`
- **Database**: SQLite (development) / PostgreSQL (production)
- **Response Format**: JSON with success/error status
- **Data Validation**: Pydantic model validation

Database Schema
- **Table**: Properties
- **Fields**: address, city, state, zip, property_type, bedrooms, bathrooms, etc.
- **Relationships**: Links to leads, deals, and AI analysis

▯ **Test Scenarios**

Scenario 1: Valid Property Creation
Objective: Test successful property creation with complete data

Test Data:
```
Address: "123 Test Street"
Unit: "Apt 5B"
City: "Atlanta"
State: "Georgia"
ZIP: "30309"
County: "Fulton"
Property Type: "Single Family"
Bedrooms: 3
Bathrooms: 2.5
Square Feet: 1800
Lot Size: 0.5
Year Built: 2020
Purchase Price: 250000
ARV: 320000
Repair Estimate: 25000
Holding Costs: 8000
Transaction Type: "Wholesale"
Assignment Fee: 15000
Description: "Beautiful single family home in great neighborhood"
Seller Notes: "Motivated seller, needs quick closing"
```

Expected Result:
- ✅ Form submission successful
- ✅ API returns 200 status
- ✅ Property data saved to database
- ✅ Success message displayed
- ✅ Redirect to properties list

Scenario 2: Minimal Required Data
Objective: Test property creation with only required fields

Test Data:
```
Address: "456 Minimal Ave"
City: "Miami"
State: "Florida"
ZIP: "33101"
County: "Miami-Dade"
Property Type: "Condo"
Transaction Type: "Retail"
Description: "Minimal test property"
```

~~~
```

**Expected Result**:
- ✅ Form submission successful
- ✅ Optional fields handled gracefully
- ✅ Property created with default values

### **Scenario 3: Financial Calculations**
**Objective**: Test automatic profit calculation

**Test Data**:
```
Purchase Price: 100000
ARV: 150000
Repair Estimate: 20000
```

**Expected Result**:
- ✅ Potential Profit: $30,000 (ARV - Purchase - Repairs)
- ✅ Calculation displayed in form
- ✅ Calculation saved to database

### **Scenario 4: Validation Errors**
**Objective**: Test form validation with invalid data

**Test Data**:
```
Address: "" (empty)
City: "" (empty)
State: "" (empty)
ZIP: "invalid"
Bedrooms: -1
Bathrooms: "not a number"
```

**Expected Result**:
- ❌ Form validation errors displayed
- ❌ API returns 422 validation error
- ❌ Property not created
- ❌ Error messages shown to user

### **Scenario 5: Database Integration**
**Objective**: Verify data persistence in database

**Test Steps**:
1. Create property via frontend
2. Check database directly
3. Retrieve property via API
4. Verify data consistency
~~~

Expected Result:
- ✅ Data exists in database
- ✅ All fields correctly stored
- ✅ Timestamps accurate
- ✅ Data retrievable via API

🔍 **Test Criteria**

Functional Requirements
- [ ] Form loads correctly
- [ ] All input fields accept data
- [ ] Validation works for required fields
- [ ] Financial calculations are accurate
- [ ] API integration functions properly
- [ ] Database insertion successful
- [ ] Success/error messages appropriate
- [ ] Navigation works after submission

Non-Functional Requirements
- [ ] Form submission under 3 seconds
- [ ] API response under 1 second
- [ ] Database operation under 500ms
- [ ] Form responsive on mobile devices
- [ ] Error handling graceful
- [ ] Data validation comprehensive

Data Integrity Requirements
- [ ] All form data captured
- [ ] Data types correct in database
- [ ] No data loss during submission
- [ ] Timestamps accurate
- [ ] Unique IDs generated
- [ ] Foreign key relationships maintained

🛠️ **Test Environment Setup**

Prerequisites
- Backend server running on `localhost:8140`
- Frontend server running on `localhost:5173`
- Database accessible and writable
- TestSprite configured and ready

Test Data Preparation
- Clean database state
- Valid test user credentials
- Test property data sets ready

```
- API endpoints verified working

### **Test Execution Order**
1. **Setup**: Start servers, verify connectivity
2. **Smoke Test**: Basic form load and display
3. **Happy Path**: Valid property creation
4. **Edge Cases**: Boundary value testing
5. **Error Handling**: Invalid data scenarios
6. **Integration**: End-to-end workflow
7. **Cleanup**: Reset test data

---

## 📊 **Success Metrics**

### **Primary Metrics**
- **Success Rate**: 100% for valid data
- **Error Rate**: 0% for valid data
- **Response Time**: < 3 seconds total
- **Data Accuracy**: 100% field mapping

### **Secondary Metrics**
- **User Experience**: Smooth form interaction
- **Error Messages**: Clear and helpful
- **Data Validation**: Comprehensive coverage
- **API Performance**: Fast response times

---

## 🚨 **Risk Assessment**

### **High Risk**
- Database connection failures
- API endpoint unavailability
- Data validation bypass
- Cross-browser compatibility

### **Medium Risk**
- Form submission timeouts
- Data type mismatches
- UI responsiveness issues
- Error message clarity

### **Low Risk**
- Minor UI styling issues
- Non-critical field validation
- Performance optimizations
- Documentation updates

---
```

```
## 📋 **Test Deliverables**

### **Test Results**
- [ ] Test execution report
- [ ] Bug reports (if any)
- [ ] Performance metrics
- [ ] Database verification results

### **Documentation Updates**
- [ ] API documentation updates
- [ ] User guide updates
- [ ] Database schema documentation
- [ ] Troubleshooting guide

---

## 🎯 **Acceptance Criteria**

### **Must Have**
- ✅ Property creation works end-to-end
- ✅ All required fields validated
- ✅ Data saved to database correctly
- ✅ Success feedback provided
- ✅ Error handling functional

### **Should Have**
- ✅ Financial calculations accurate
- ✅ Form responsive design
- ✅ Fast submission times
- ✅ Clear error messages
- ✅ Data retrieval verification

### **Could Have**
- ✅ Advanced validation rules
- ✅ Real-time form validation
- ✅ Auto-save functionality
- ✅ Bulk property import
- ✅ Property templates

---

## 📅 **Timeline**

- **Test Planning**: 30 minutes
- **Test Execution**: 45 minutes
- **Bug Fixing**: 30 minutes (if needed)
- **Documentation**: 15 minutes
- **Total**: 2 hours
```

```
---

## 👥 **Stakeholders**

- **Product Owner**: Feature requirements
- **Frontend Developer**: UI/UX implementation
- **Backend Developer**: API and database
- **QA Tester**: Test execution
- **End User**: Property investors

---

## 📞 **Contact Information**

- **Test Lead**: AI Assistant
- **Backend Developer**: Available for API issues
- **Frontend Developer**: Available for UI issues
- **Database Admin**: Available for data issues

---

**Document Status**: ✅ Ready for TestSprite Execution
**Next Step**: Run TestSprite test suite for Add Property functionality
```