# OrPaynter Autonomous Roofing Lead Generation and Qualification MCP Server

A comprehensive Model Context Protocol (MCP) server that provides autonomous roofing lead generation, qualification, contractor matching, and appointment scheduling capabilities. Built with FastMCP framework for production-ready deployment and scalability.

## 🚀 Features

### 1. Lead Qualification Chatbot

* Intelligent conversation flow to gather critical roofing information
* Automated data validation and lead scoring (1-10 scale)
* Collects: property details, damage type, urgency, insurance info, contact details
* Determines qualification status for contractor referral

### 2. Database Management

* Structured SQLite database for all lead information
* Complete lead lifecycle tracking (new, qualified, contacted, scheduled, closed)
* Data export capabilities for CRM integration
* Comprehensive audit trail and analytics

### 3. Contractor Matching & Notifications

* Smart matching based on location, specialization, and availability
* Instant email and SMS notifications to qualified contractors
* Performance tracking and success rate monitoring
* Contractor capacity and response time management

### 4. Appointment Scheduling

* Calendar integration for inspection appointments
* Automated confirmation and reminder notifications
* Reschedule and cancellation handling
* Multi-timezone support

### 5. Document Management

* Photo upload processing for damage assessment
* Insurance document handling
* Inspection report generation
* Organized file storage with metadata tracking

## 📋 System Requirements

* Python 3.10+
* SQLite (built-in)
* Internet connection for email/SMS notifications
* Optional: SendGrid account for email notifications
* Optional: Twilio account for SMS notifications

## 📦 Installation

### Quick Install with uv (Recommended)

# Clone or create the project directory  
git clone <repository-url> orpaynter-mcp-server  
cd orpaynter-mcp-server  
  
# Install with uv  
uv sync  
  
# Run the server  
uv run server.py

### Manual Installation

# Create virtual environment  
python -m venv .venv  
source .venv/bin/activate # Linux/Mac  
# or  
.venv\Scripts\activate # Windows  
  
# Install dependencies  
pip install -r requirements.txt  
  
# Run the server  
python server.py

## 🛠️ MCP Tools

The server exposes the following tools to LLMs:

### qualify\_lead

Process chatbot responses and score leads (1-10) based on: - Damage severity and urgency - Insurance coverage and claims status - Decision-making authority - Property details and documentation

### match\_contractors

Find and notify appropriate contractors based on: - Geographic location (ZIP code matching) - Specialization requirements - Availability and capacity - Performance ratings and response times

### schedule\_appointment

Handle appointment booking with: - Calendar integration - Automated confirmations - Reminder notifications - Rescheduling capabilities

### store\_lead\_data

Manage lead information with: - Structured data validation - Full CRUD operations - Audit trail maintenance - Data integrity checks

### send\_notifications

Handle communications via: - Email notifications (SendGrid) - SMS notifications (Twilio) - Template-based messaging - Delivery tracking and retries

### process\_documents

Manage file uploads including: - Photo damage assessment - Insurance document processing - File organization and metadata - Image quality analysis

### generate\_reports

Create comprehensive reports for: - Individual lead summaries - Contractor performance metrics - Weekly analytics dashboards - Business intelligence insights

### track\_analytics

Monitor and track: - Conversion rates and performance - Lead qualification metrics - Contractor response times - System performance indicators

## 📊 Resources

### Lead Status Tracking

orpaynter://leads/{lead\_id}

Get real-time status and details of specific leads.

### Contractor Availability

orpaynter://contractors/available

Get list of currently available contractors with ratings and specialties.

## 🤖 Prompts

### Lead Qualification Conversation

Intelligent conversation prompts for gathering lead information through natural dialogue, with context-aware follow-up questions based on customer responses.

## ⚙️ Configuration

### Environment Variables

#### Database Configuration

* ORPAYNTER\_DB\_PATH: Path to SQLite database file (default: /tmp/orpaynter.db)
* ORPAYNTER\_UPLOADS\_DIR: Directory for file uploads (default: /tmp/orpaynter\_uploads)

#### Email Notifications (Optional)

* SENDGRID\_API\_KEY: SendGrid API key for email functionality

#### SMS Notifications (Optional)

* TWILIO\_ACCOUNT\_SID: Twilio Account SID
* TWILIO\_AUTH\_TOKEN: Twilio Auth Token
* TWILIO\_PHONE\_NUMBER: Twilio phone number for sending SMS

### MCP Server Configuration

{  
 "name": "agent\_generated\_orpaynter\_lead\_generation",  
 "exhibit\_name": "OrPaynter Lead Generation System",  
 "command": "sh /path/to/orpaynter-mcp-server/run.sh",  
 "env": {  
 "ORPAYNTER\_DB\_PATH": "/your/database/path/orpaynter.db",  
 "SENDGRID\_API\_KEY": "your\_sendgrid\_api\_key",  
 "TWILIO\_ACCOUNT\_SID": "your\_twilio\_sid",  
 "TWILIO\_AUTH\_TOKEN": "your\_twilio\_token",  
 "TWILIO\_PHONE\_NUMBER": "your\_twilio\_phone"  
 }  
}

## 🚀 Usage Examples

### Qualifying a New Lead

# Via MCP tool call  
result = await call\_tool("qualify\_lead", {  
 "contact\_name": "John Smith",  
 "contact\_email": "john@example.com",  
 "contact\_phone": "(555) 123-4567",  
 "property\_address": "123 Main St",  
 "city": "Springfield",  
 "state": "IL",  
 "zip\_code": "62701",  
 "property\_type": "residential",  
 "damage\_type": "storm damage",  
 "damage\_severity": "severe",   
 "damage\_description": "Missing shingles and visible leak",  
 "urgency\_level": 8,  
 "has\_insurance": True,  
 "is\_decision\_maker": True,  
 "roof\_age": 15,  
 "insurance\_company": "State Farm"  
})

### Matching Contractors

# Find contractors for qualified lead  
contractors = await call\_tool("match\_contractors", {  
 "lead\_id": "LEAD\_20250708\_145632",  
 "specialty\_required": "storm damage",  
 "max\_contractors": 3  
})

### Scheduling Appointments

# Schedule inspection appointment  
appointment = await call\_tool("schedule\_appointment", {  
 "lead\_id": "LEAD\_20250708\_145632",  
 "contractor\_id": "CONT\_001",  
 "appointment\_date": "2025-07-15T10:00:00",  
 "appointment\_type": "inspection",  
 "notes": "Storm damage assessment needed"  
})

## 📈 Analytics and Reporting

The system provides comprehensive analytics including:

* **Lead Conversion Rates**: Track qualification and appointment rates
* **Contractor Performance**: Response times, completion rates, customer satisfaction
* **Revenue Analytics**: Pipeline value, average deal size, seasonal trends
* **Operational Metrics**: System performance, notification delivery rates

## 🔒 Security Features

* Input validation and sanitization
* SQL injection prevention
* File upload security and virus scanning
* Rate limiting for API calls
* Secure credential management
* Audit logging for all activities

## 🏗️ Architecture

┌─────────────────┐ ┌──────────────────┐ ┌─────────────────┐  
│ MCP Client │───▶│ FastMCP Server │───▶│ SQLite Database│  
│ (AI Assistant) │ │ (OrPaynter) │ │ (Lead Data) │  
└─────────────────┘ └──────────────────┘ └─────────────────┘  
 │  
 ▼  
 ┌──────────────────┐  
 │ External APIs │  
 │ • SendGrid │  
 │ • Twilio │  
 │ • File Storage │  
 └──────────────────┘

## 🧪 Testing

### Unit Tests

# Run all tests  
uv run pytest  
  
# Run with coverage  
uv run pytest --cov=server --cov-report=html

### Integration Tests

# Test MCP server functionality  
uv run pytest tests/test\_integration.py -v

### Manual Testing

# Start server in development mode  
uv run server.py --transport stdio  
  
# Use MCP client to test tools  
fastmcp test server.py

## 🚢 Deployment

### Local Development

# Development server with auto-reload  
uv run server.py --transport stdio --dev

### Production Deployment

# Production server via HTTP  
uv run server.py --transport http --host 0.0.0.0 --port 8000  
  
# Or via SSE  
uv run server.py --transport sse --host 0.0.0.0 --port 8000

### Docker Deployment

FROM python:3.10-slim  
WORKDIR /app  
COPY . .  
RUN pip install uv && uv sync  
CMD ["uv", "run", "server.py", "--transport", "http", "--host", "0.0.0.0", "--port", "8000"]

## 🎯 Performance Optimization

* **Database Indexing**: Optimized queries for lead and contractor searches
* **Connection Pooling**: Efficient database connection management
* **Caching**: In-memory caching for frequently accessed data
* **Async Operations**: Non-blocking I/O for all external API calls
* **Rate Limiting**: Protection against API abuse

## 🔄 Lead Workflow

1. **Initial Contact** → Lead enters system via chatbot/form
2. **Qualification** → AI-driven conversation gathers requirements
3. **Scoring** → Automated scoring based on multiple criteria
4. **Contractor Matching** → Location and specialty-based matching
5. **Notification** → Instant alerts to qualified contractors
6. **Appointment Scheduling** → Calendar integration and confirmations
7. **Follow-up** → Automated reminders and status tracking
8. **Analytics** → Performance monitoring and optimization

## 📚 API Documentation

### Lead Qualification Score Calculation

The system uses a sophisticated scoring algorithm:

* **Base Score**: 5 points
* **Damage Severity**: Emergency (+3), Severe (+2), Moderate (+1), Minor (0)
* **Urgency Level**: 8-10 (+2), 6-7 (+1), 1-5 (0)
* **Insurance Coverage**: Has insurance (+1), Claim filed (+1)
* **Decision Authority**: Is decision maker (+1)
* **Documentation**: Photos uploaded (+1)

**Total Range**: 1-10 points **Qualification Threshold**: 6+ points

### Contractor Matching Algorithm

Contractors are matched based on:

1. **Geographic Coverage**: ZIP code service areas
2. **Specialization Match**: Damage type expertise
3. **Availability Status**: Current capacity
4. **Performance Metrics**: Rating and response time
5. **Workload Balancing**: Even distribution of leads

## 🤝 Contributing

We welcome contributions! Please see our contributing guidelines for:

* Code style and standards
* Testing requirements
* Pull request process
* Issue reporting

## 📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

## 📞 Support

For support and questions:

* Create an issue in the repository
* Contact the development team
* Check the documentation wiki

## 🎯 Roadmap

### Phase 1 (Current)

* ✅ Core lead qualification system
* ✅ Contractor matching and notifications
* ✅ Basic appointment scheduling
* ✅ Document processing

### Phase 2 (Planned)

* 🔄 Advanced analytics dashboard
* 🔄 Machine learning lead scoring
* 🔄 CRM integrations (Salesforce, HubSpot)
* 🔄 Mobile app for contractors

### Phase 3 (Future)

* 📋 Video call integration
* 📋 Automated follow-up sequences
* 📋 Multi-language support
* 📋 Advanced reporting and BI

Built with ❤️ using [FastMCP](https://github.com/jlowin/fastmcp) framework.