Spitch.ch Integration Technical Specification

PowerOn Platform | *Version 1.0* | *Date: August 2025*

📋 Executive Summary

Technical specification for integrating spitch.ch telephone service into PowerOn platform. Enables external customers to call companies via phone, interact with AI using mandate-specific data, and stores call transcripts in PowerOn database.

1. Project Overview

1.1 Integration Scope

- New "spitch" router for external service communication
- Enhanced mandate management with spitch-specific data
- New call transcript data model and storage
- Frontend UI for mandate-user management and call permissions
- API key authentication for external service endpoints
- **NEW:** Customer registration workflow for telephone service
- **NEW:** Document content extraction engine integration

1.2 Service Logic Summary

PowerOn Side:

- Stores mandate-specific customer data (general info, staff, services, FAQs)
- Receives and stores call transcripts from spitch.ch
- Provides customer data to spitch.ch via API endpoints
- Manages user permissions for call data access
- Maintains mandate isolation and security
- **NEW:** Generates AI-powered documents for Spitch using document content extraction engine
- **NEW:** Manages customer registration workflow for telephone service

Spitch.ch Side:

- Receives customer data from PowerOn for AI conversation context
- Handles phone calls and AI conversations with customers
- Sends call transcripts back to PowerOn for storage
- Uses mandate-specific data to personalize customer interactions
- **NEW:** Creates technical SIP numbers for customers
- **NEW:** Operates calls independently as a service

1.3 Product Manager View - Customer Journey

🎯 **Streamlined Customer Journey (Confirmed by Spitch PM Vadim)**

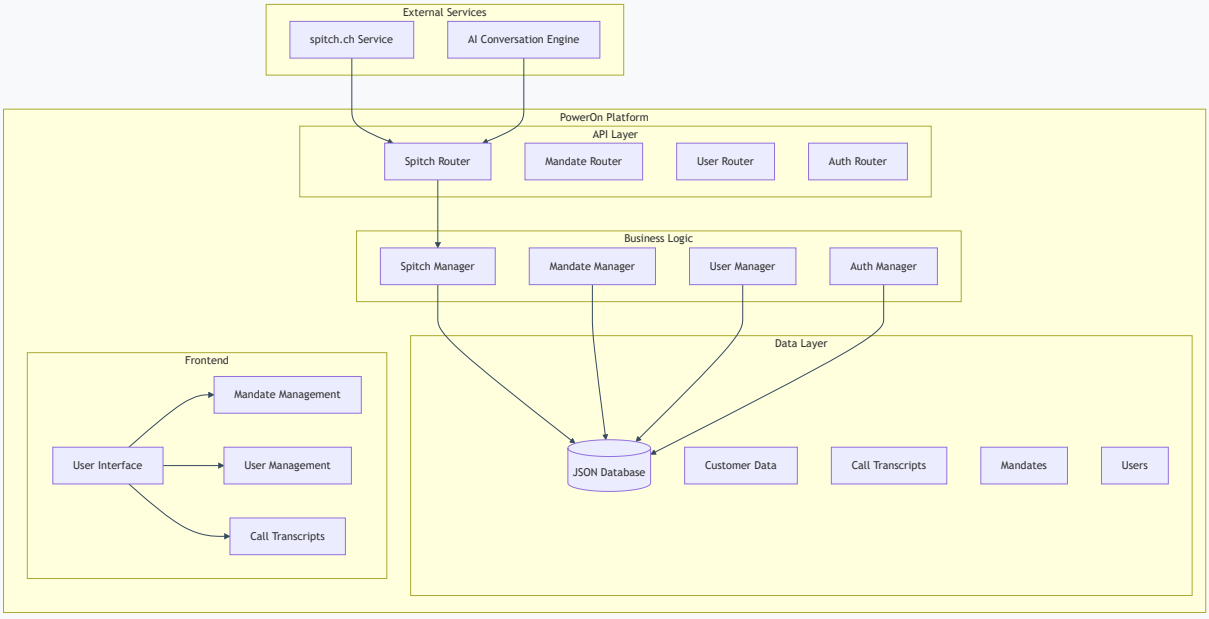
1. **Customer Registration:** Customer registers with PowerOn for telephone service with company profile
2. **Initial Setup:** Initially manual, final workflow-driven where customer drops documents into the box and provides links and text fragments
3. **Integration:** This becomes another connector like Outlook or SharePoint
4. **Profile Transfer:** PowerOn sends new customer profile to Spitch
5. **Technical Setup:** Spitch initializes customer and creates technical SIP number
6. **Call Forwarding:** Customer sets up call forwarding to the technical number (can remove anytime if issues arise - ensuring BCM safety)
7. **Document Generation:** PowerOn AI generates documents for Spitch (already active, document content extraction engine)
8. **Ready State:** Customer can switch their phone number to the technical number (SIP) anytime, saving telephony costs

1.4 Data Synchronization Approach

🔄 **Data Flow Strategy (Confirmed by Spitch PM Vadim)**

- **Independent Call Operation:** Spitch operates calls independently as a service
- **Regular Data Sync:** Spitch reads current documents per customer from PowerOn API daily
- **Real-time Transcripts:** Spitch feeds call transcripts to PowerOn API after each call
- **JSON Document Format:** Spitch receives a JSON document per customer, no files

2. System Architecture Overview



3. Data Models

3.1 Enhanced Mandate Model

```
class Mandate(BaseModel, ModelMixin):
    id: str = Field(default_factory=lambda: str(uuid.uuid4()))
    name: str = Field(description="Name of the mandate")
    language: str = Field(default="en", description="Default language")
    enabled: bool = Field(default=True, description="Mandate status")
    spitch_enabled: bool = Field(default=False, description="Spitch service enabled")
    spitch_config: Optional[Dict[str, Any]] = Field(default=None, description="Spitch configuration")
    created_at: datetime = Field(default_factory=datetime.now)
    modified_at: datetime = Field(default_factory=datetime.now)
    created_by: str = Field(default="")
    modified_by: str = Field(default="")
```

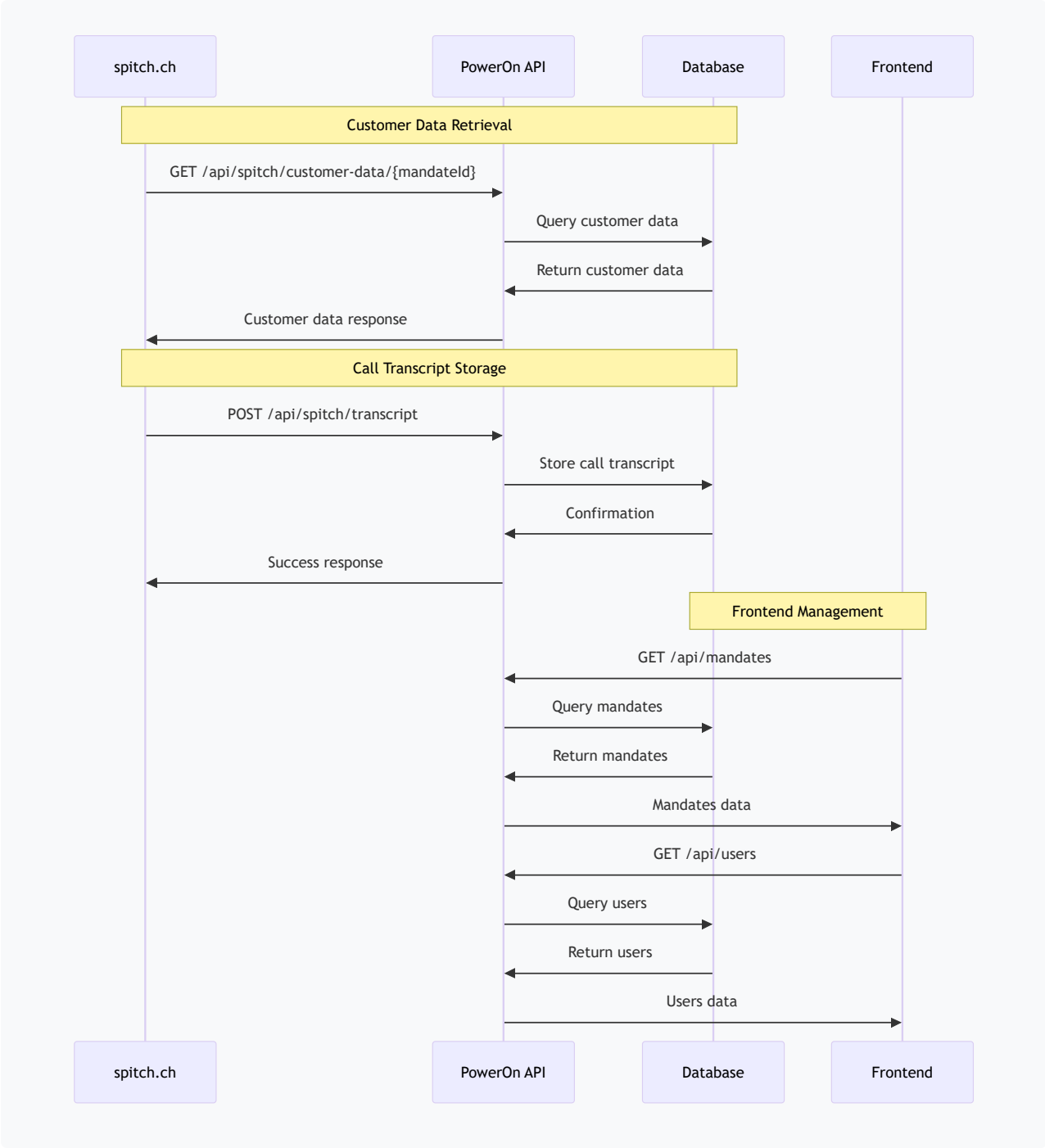
3.2 Spitch Customer Data Model

```
class SpitchCustomerData(BaseModel, ModelMixin):
    id: str = Field(default_factory=lambda: str(uuid.uuid4()))
    mandateId: str = Field(description="Associated mandate ID")
    customer_general: Dict[str, Any] = Field(description="General customer information")
    customer_staff: List[Dict[str, Any]] = Field(description="Staff information")
    customer_service: List[Dict[str, Any]] = Field(description="Service information")
    customer_faq: List[Dict[str, Any]] = Field(description="FAQ data")
    mandates: List[Dict[str, Any]] = Field(description="List of customer mandates with data about main topics
    contacts: List[Dict[str, Any]] = Field(description="List of contacts with phone numbers and references to
    last_updated: datetime = Field(default_factory=datetime.now)
    created_by: str = Field(description="User ID who created the data")
    modified_by: str = Field(description="User ID who last modified the data")
```

3.3 Call Transcript Model

```
class CallTranscript(BaseModel, ModelMixin):
    id: str = Field(default_factory=lambda: str(uuid.uuid4()))
    mandateId: str = Field(description="Associated mandate ID")
    start_datetime: datetime = Field(description="Call start time")
    finish_datetime: datetime = Field(description="Call end time")
    caller_phone: str = Field(description="Caller's phone number")
    recipient_phone: str = Field(description="Recipient's phone number")
    transcript_text: str = Field(description="Call transcript content")
    subject: str = Field(description="Call subject/topic")
    tags: List[str] = Field(default_factory=list, description="Call tags")
    spitch_call_id: str = Field(description="External spitch call identifier")
    created_at: datetime = Field(default_factory=datetime.now)
    created_by: str = Field(default="")
```

3.4 Data Flow Diagram



4. Pilot Customer Data

4.1 Initial Pilot Customers

Pilot Customer 1: ValueOn AG

- **Company:** ValueOn AG
- **Phone Number:** +41 44 9437040
- **Provider:** Swisscom Fixnet AG
- **Status:** Ready for spitch.ch integration

Pilot Customer 2: PamoCreate AG

- **Company:** PamoCreate AG

- **Phone Number:** +41 55 2462345
- **Provider:** iWay AG
- **Status:** Ready for spitch.ch integration

4.2 Pilot Customer Setup Requirements

- Mandate creation in PowerOn with spitch_enabled = True
- Customer data population (general info, staff, services, FAQs)
- Phone number mapping configuration in spitch.ch system
- Test call validation and transcript storage verification
- User permission setup for call data access

5. Spitch.ch Service Perspective

5.1 Data Received from PowerOn per Customer/Mandate

```
{
    "mandateId": "uuid-string",
    "customer_general": {
        "company_name": "Company Name",
        "industry": "Industry Type",
        "contact_info": {
            "email": "contact@company.com",
            "phone": "+41 123 456 789",
            "address": "Company Address"
        },
        "business_hours": "9:00-17:00",
        "timezone": "Europe/Zurich"
    },
    "customer_staff": [
        {
            "name": "Staff Member Name",
            "role": "Position/Role",
            "department": "Department",
            "contact": "+41 123 456 789",
            "expertise": ["Area 1", "Area 2"]
        }
    ],
    "customer_service": [
        {
            "service_name": "Service Name",
            "description": "Service Description",
            "pricing": "Price Information",
            "availability": "Availability Details",
            "requirements": ["Requirement 1", "Requirement 2"]
        }
    ],
    "customer_faq": [
        {
            "question": "Frequently Asked Question",
            "answer": "Standard Answer",
            "category": "Question Category",
            "keywords": ["keyword1", "keyword2"]
        }
    ],
    "mandates": [
        {
```

```
            "mandate_id": "uuid-string",
            "name": "Mandate Name",
            "main_topics": ["Topic 1", "Topic 2", "Topic 3"],
            "description": "Mandate description",
            "status": "active",
            "priority": "high"
        }
    ],
    "contacts": [
        {
            "name": "Contact Person Name",
            "phone_number": "+41 123 456 789",
            "email": "contact@company.com",
            "role": "Contact Role",
            "mandate_references": ["uuid-string-1", "uuid-string-2"],
            "is_primary": true
        }
    ]
}
```

5.2 Data Delivered to PowerOn per Call

```
{
    "mandateId": "uuid-string",
    "start_datetime": "2025-01-15T10:30:00Z",
    "finish_datetime": "2025-01-15T10:45:00Z",
    "caller_phone": "+41 987 654 321",
    "recipient_phone": "+41 123 456 789",
    "transcript_text": "Full conversation transcript...",
    "subject": "Service Inquiry",
    "tags": ["service", "inquiry", "new-customer"],
    "spitch_call_id": "spitch-call-uuid-123"
}
```

5.3 Call Processing Workflow

1. Customer calls spitch.ch service
2. spitch.ch identifies mandate based on phone number
3. spitch.ch requests customer data from PowerOn API
4. AI uses customer data for personalized conversation
5. Call concludes, transcript generated
6. spitch.ch sends transcript to PowerOn for storage

6. API Endpoints

6.1 Spitch Router Endpoints

Endpoint	Method	Description	Authentication	Rate Limit
/api/spitch/customer-data/{mandateId}	GET	Retrieve customer data for a specific mandate	API Key	100/minute

Endpoint	Method	Description	Authentication	Rate Limit
/api/spitch/customer-data/{mandateId}?since={date}	GET	Retrieve changed customer data since specific date	API Key	100/minute
/api/spitch/transcript	POST	Store call transcript for a mandate	API Key	50/minute

6.2 Enhanced Mandate Endpoints

Endpoint	Method	Description	Authentication
/api/mandates/{mandateId}/spitch-config	PUT	Update spitch configuration for mandate	JWT (Admin/SysAdmin)
/api/mandates/{mandateId}/customer-data	GET	Get customer data for mandate	JWT
/api/mandates/{mandateId}/customer-data	PUT	Update customer data for mandate	JWT (Admin/SysAdmin)

7. Security Implementation

7.1 API Key Authentication

```
# New security middleware for spitch endpoints
class SpitchAPIKeyAuth:
    def __init__(self, api_keys: List[str]):
        self.api_keys = set(api_keys)

    async def __call__(self, request: Request):
        api_key = request.headers.get("X-API-Key")
        if not api_key or api_key not in self.api_keys:
            raise HTTPException(
                status_code=status.HTTP_401_UNAUTHORIZED,
                detail="Invalid API key"
            )
        return api_key
```

7.2 Rate Limiting

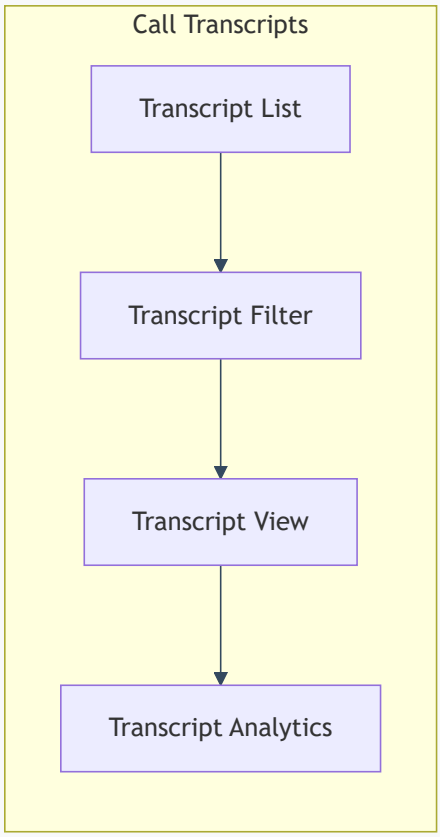
- **Customer Data Retrieval:** 100 requests per minute per API key
- **Transcript Storage:** 50 requests per minute per API key
- **IP-based Limiting:** Additional protection against abuse

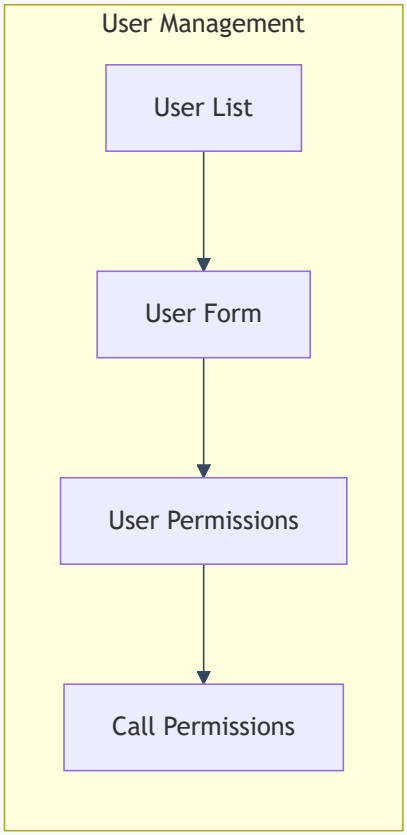
7.3 Mandate Isolation

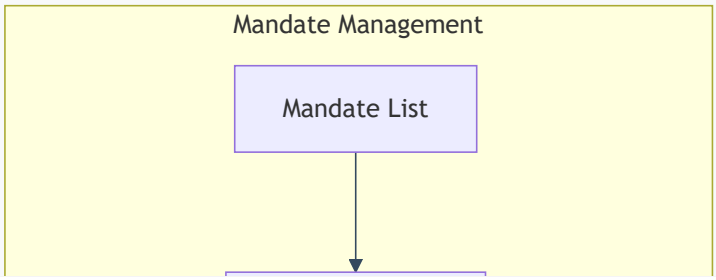
- All data queries filtered by mandate ID
- Users can only access data from their assigned mandate
- API key requests validated against mandate permissions

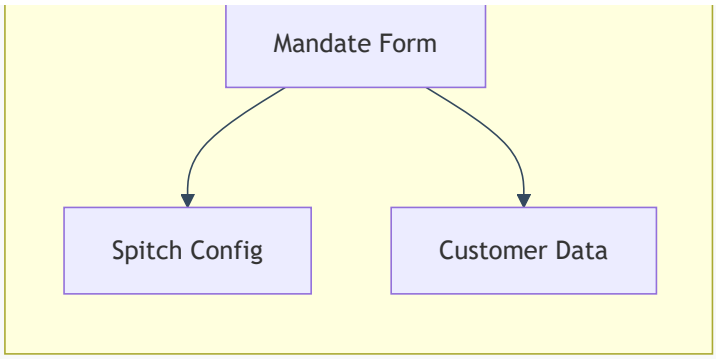
8. Frontend Implementation

8.1 Enhanced Mandate Management UI


Call Transcripts
Transcript List
Transcript Filter
Transcript View
Transcript Analytics


User Management
User List
User Form
User Permissions
Call Permissions


Mandate Management
Mandate List



8.2 New UI Components

Component	File	Description	Dependencies
SpitchConfigPanel	js/modules/spitchConfig.js	Mandate spitch configuration management	formGeneric.js, apiCalls.js
CustomerDataManager	js/modules/customerData.js	Customer data CRUD operations	formGeneric.js, apiCalls.js
CallTranscriptsView	js/modules/callTranscripts.js	Call transcript display and filtering	formGeneric.js, apiCalls.js
UserCallPermissions	js/modules/userCallPermissions.js	User permissions for call data access	formGeneric.js, apiCalls.js

9. Implementation Sprints

Sprint 1: Backend Foundation (2 weeks)

- Create new data models (SpitchCustomerData, CallTranscript)
- Enhance existing Mandate model with spitch fields
- Implement spitch router with basic endpoints
- Add API key authentication middleware
- Create database migration scripts

Package Price: 18,000 CHF

Sprint 2: Core API Implementation (2 weeks)

- Implement customer data retrieval endpoints
- Implement call transcript storage endpoints
- Add rate limiting and security measures
- Create comprehensive error handling
- Write API documentation and tests

Package Price: 20,000 CHF

Sprint 3: Frontend Development (2 weeks)

- Create spitch configuration UI components
- Implement customer data management interface
- Build call transcript viewing and filtering
- Add user permission management for call data
- Integrate with existing mandate/user management

Package Price: 22,000 CHF

Sprint 4: Testing & Integration (2 weeks)

- End-to-end testing with spitch.ch service
- Performance testing and optimization
- Security audit and penetration testing
- User acceptance testing
- Production deployment preparation

Package Price: 20,000 CHF

10. Project Pricing

Sprint Package Pricing

Sprint	Duration	Package Price (CHF)	Description
Sprint 1	2 weeks	18,000	Backend Foundation
Sprint 2	2 weeks	20,000	Core API Implementation
Sprint 3	2 weeks	22,000	Frontend Development
Sprint 4	2 weeks	20,000	Testing & Integration
Total Project	**8 weeks**	**80,000**	**Complete Integration**

Investment Schedule

- **Sprint 1:** 18,000 CHF (upon completion)
- **Sprint 2:** 20,000 CHF (upon completion)
- **Sprint 3:** 22,000 CHF (upon completion)
- **Sprint 4:** 20,000 CHF (upon completion)

Project Scope

- Complete backend API implementation
- Full frontend UI development

- Comprehensive testing and security audit
- Production deployment and documentation
- Post-deployment support (2 weeks)

11. Risk Assessment

11.1 Technical Risks

- **API Integration Complexity:** External service dependencies may cause reliability issues
- **Data Volume:** High call volume could impact database performance
- **Security Vulnerabilities:** New endpoints increase attack surface

11.2 Mitigation Strategies

- Implement comprehensive error handling and fallback mechanisms
- Add database indexing and archiving strategies for call transcripts
- Regular security audits and penetration testing
- Implement monitoring and alerting for all new endpoints

12. Success Metrics

12.1 Technical Metrics

- API response time < 200ms for customer data retrieval
- 99.9% uptime for spitch integration endpoints
- Zero data loss for call transcripts
- Successful integration with spitch.ch service

12.2 Business Metrics

- Reduced customer service response time
- Improved customer satisfaction through AI-powered phone support
- Increased mandate engagement and retention
- Cost savings through automated customer interactions

13. Conclusion

The spitch.ch integration extends PowerOn platform capabilities for external telephone service integration while maintaining existing security and mandate isolation principles. Implementation follows established patterns and leverages existing infrastructure for maintainability and scalability.

Total project cost: CHF 80,000 for complete integration delivered in 8 weeks across 4 sprints.

14. Project Status & Next Steps

📞 Final Q&A Session Results (Spitch PM Vadim)

Status: All questions clarified and requirements confirmed. Project is ready to proceed.

14.1 Confirmed Requirements

- **Customer Journey:** Streamlined workflow confirmed - customer registration → profile transfer → SIP setup → call forwarding
- **Data Format:** JSON document per customer (no files) with enhanced mandates[] and contacts[] attributes
- **Integration Approach:** PowerOn as another connector (like Outlook/SharePoint) with document content extraction engine
- **BCM Safety:** Customers can remove call forwarding anytime if issues arise
- **Cost Savings:** Customers can switch to SIP numbers to save telephony costs

14.2 Next Steps Timeline

1. **This Week:** Spitch team calculates costs and timeline
2. **Next Week:** Vadim informs Alexey for implementation decision
3. **Following Week:** Call with Dominic and Alexey for launch approval

14.3 Key Benefits Confirmed

- **Customer Experience:** Seamless integration with existing PowerOn workflows
- **Technical Efficiency:** Leverages existing document content extraction engine
- **Business Continuity:** BCM-safe with removable call forwarding
- **Cost Optimization:** Potential telephony cost savings through SIP integration
- **Scalability:** Framework for additional external service integrations

📞 Next Steps

1. Review technical specification
2. Establish development timeline
3. Begin Phase 1 implementation
4. Schedule progress reviews
5. Plan implementation deployment
