ESG Copilot MVP

Technical Documentation & Implementation Guide

Product Name: ESG Copilot
Platform: Sweep Sustainability Reporting Platform
Technology: MCP LLM Integration
Document Date: September 24, 2025
Document Owner: Shireen Hussain J
Version: 1.0 - MVP Specification
Classification: Internal - Development Team

Table of Contents

---

1. Overview & Business Case

1.1 Executive Summary

The ESG Copilot is an AI-powered conversational assistant embedded directly within the Sweep platform, designed to democratize ESG expertise and accelerate user productivity through natural language interaction. By leveraging MCP (Model Context Protocol) LLM integration, the Copilot provides contextual guidance, explains complex compliance requirements, validates data quality, and assists with platform navigation—reducing time-to-value for new users and increasing efficiency for experienced teams.

1.2 Business Problem

Current Challenges:

- **Steep Learning Curve**: ESG reporting involves complex regulations (CSRD, SFDR, ISSB, GRI) that take weeks to master
- **Manual Documentation Review**: Users spend hours searching help documentation and regulatory guidance
- **Data Validation Confusion**: Users struggle to understand AI-flagged anomalies and data quality scores
- **Audit Trail Complexity**: Explaining audit logs and compliance status requires ESG expertise
- **Onboarding Friction**: New users need 15+ hours of training before productive use

Business Impact:

- 40% of support tickets are "how-to" questions answerable by documentation
- Average 2.5 hours per week per user spent searching for guidance
- 6-week average time to proficiency for new users
- 35% of potential customers cite complexity as barrier to adoption

1.3 Solution: ESG Copilot

An intelligent chat assistant that:

- **Explains Compliance**: Translates regulatory jargon into plain language
- **Validates Data**: Interprets AI anomaly flags and quality scores
- **Guides Navigation**: Helps users find features and complete workflows
- **Summarizes Insights**: Converts complex dashboards into actionable summaries
- **Accelerates Onboarding**: Reduces training time through interactive learning

1.4 Success Metrics (MVP)

Metric	Target	Measurement
Support Ticket Reduction	25% decrease	Month-over-month comparison
Time to First Value	<15 minutes	From signup to first successful action
User Engagement	60% weekly active usage	Users interacting with Copilot weekly
Query Resolution Rate	75% first-attempt success	Queries resolved without escalation
User Satisfaction	4.2/5.0 rating	In-app feedback survey
Onboarding Time	<3 hours to proficiency	Time tracking in user sessions

1.5 MVP Scope

In Scope:

- Natural language Q&A about platform features and ESG compliance
- Dashboard and report summaries
- Data validation explanations (anomaly flags, quality scores)
- Audit trail interpretations
- Guided onboarding assistance
- Context-aware responses based on user role and current page

Out of Scope (Future Phases):

- Direct data manipulation ("Add this emission record")
- Report generation ("Create CSRD report for 2024")
- Complex calculations or analysis
- Multi-turn complex workflows
- Voice interaction
- Mobile app integration

---

2. Architecture Design

2.1 High-Level Architecture

```
┌──────────────────────────────────────────────────────────────────────────┐
│                SWEEP PLATFORM                                │
│  ┌──────────────────────────────────────────────────────────────────┐   │
│  │             Frontend Layer (React)                │ │
│  │  ┌─────────────────┐  ┌────────────────┐  ┌─────────────────────┐  │ │
│  │  │ Chat Widget │ │ Main UI │ │ Context Manager │ │ │
│  │  │ (Chat UI)  │ │ Dashboard │ │ (Page/User) │ │ │
│  │  └────────┬────────┘  └────────────────┘  └──────────┬──────────┘  │ │
│  └───────────┼────────────────────────────────────────────┼────────────┘ │
│      │                │      │
│      │ HTTPS/WSS            │      │
│      ▼                ▼      │
│  ┌──────────────────────────────────────────────────────────────────────┐
│
│  │             API Gateway Layer                 │ │
│  │  ┌──────────────────────┐  ┌──────────────────────────────────────┐  │ │
│  │  │ Authentication │ │ Rate Limiting & Validation │ │ │
│  │  │ (OAuth2/JWT) │ │ (100 queries/user/hour) │ │ │
│  │  └──────────────────────┘  └──────────────────────────────────────┘  │ │
│  └──────────────────────────────────┬───────────────────────────────────┘
│
│            │              │
│            ▼              │
│  ┌──────────────────────────────────────────────────────────────────────┐
│
│  │        ESG Copilot Service (Node.js/FastAPI)          │ │
│  │  ┌─────────────────┐  ┌─────────────────┐  ┌────────────────────┐  │ │
│  │  │ Session Mgmt │ │ Context │ │ Query Router │ │ │
│  │  │ & History │ │ Enrichment │ │ & Intent Detect │ │ │
│  │  └────────┬────────┘  └────────┬────────┘  └─────────┬──────────┘  │ │
│  └───────────┼────────────────────┼────────────────────┼─────────────┘ │
│      │       │       │      │
│      ▼       ▼       ▼      │
│  ┌──────────────────────────────────────────────────────────────────────┐
│
│  │          MCP LLM Proxy Service                 │ │
│  │  ┌─────────────────┐  ┌─────────────────┐  ┌────────────────────┐  │ │
│  │  │ Prompt │ │ MCP Protocol │ │ Response │ │ │
│  │  │ Engineering │ │ Handler │ │ Post-Processing │ │ │
│  │  └────────┬────────┘  └────────┬────────┘  └─────────┬──────────┘  │ │
│  └───────────┼────────────────────┼────────────────────┼─────────────┘ │
│      │       │       │      │
│      ▼       ▼       ▼      │
│    ┌──────────────────────────────────────────────┐      │
│       │   External MCP LLM Service          │         │
│       │  (Claude/GPT-4 via MCP Protocol)     │         │
│       └──────────────────────────────────────────────┘         │
│                 ▲                         │
│                 │                        │
│    ┌───────────────────────────┬──────────────────────────────┐    │
│    │         Data Context & Retrieval Layer          │  │
│    │  ┌──────────┐  ┌──────────┐  ┌──────────┐  ┌─────────┐  │  │
│    │  │ ESG Data │  │ User     │  │ Audit    │  │ Help     │  │  │
│    │  │ Service  │  │ Profile  │  │ Logs     │  │ Docs     │  │  │
│    │  └──────────┘  └──────────┘  └──────────┘  └─────────┘  │  │
│    └──────────────────────────────────────────────────────────┘    │
└────────────────────────────────────────────────────────────────────┘
```

2.2 Component Descriptions

2.2.1 Frontend Layer - Chat Widget (React)

Responsibilities:

- Render conversational UI with message history
- Handle user input and real-time streaming responses
- Display rich content (tables, charts, code snippets)
- Manage loading states and error handling
- Track user engagement analytics

Key Technologies:

- React 18 with TypeScript
- WebSocket for streaming responses
- Markdown rendering for formatted responses
- Context hooks for page awareness

Component Structure:

typescript

```
<ChatWidget>
  <ChatHeader />
  <MessageList>
    <UserMessage />
    <CopilotMessage />
    <TypingIndicator />
  </MessageList>
  <InputArea>
    <TextInput />
    <SendButton />
    <SuggestedPrompts />
  </InputArea>
</ChatWidget>
```

2.2.2 API Gateway Layer

Responsibilities:

- Authenticate requests using JWT tokens
- Validate user permissions and data access scopes
- Rate limit queries (100/hour per user, 1000/hour per org)
- Route requests to appropriate backend services
- Log all interactions for audit and analytics

Security Headers:

http

```http
Authorization: Bearer <JWT_TOKEN>
X-User-ID: user-123
X-Organization-ID: org-456
X-Session-ID: session-789
X-Context: dashboard-emissions-overview
```

2.2.3 ESG Copilot Service

Responsibilities:

- Manage chat sessions and conversation history
- Enrich queries with user context (role, page, recent actions)
- Route queries based on intent detection
- Retrieve relevant data from Sweep services
- Format responses for optimal user experience
- Track query performance and success rates

Core Functions:

python

```python
# Query Processing Pipeline
async def process_query(query: str, context: UserContext):
    # 1. Session Management
    session = await get_or_create_session(context.user_id)

    # 2. Context Enrichment
    enriched_context = await enrich_context(query, context, session)

    # 3. Intent Detection
    intent = await detect_intent(query, enriched_context)

    # 4. Data Retrieval (if needed)
    relevant_data = await fetch_relevant_data(intent, enriched_context)

    # 5. MCP LLM Call
    response = await call_mcp_llm(query, enriched_context, relevant_data)

    # 6. Post-Processing
    formatted_response = await format_response(response, intent)

    # 7. Audit Logging
    await log_interaction(query, response, context, session)

    return formatted_response
```

2.2.4 MCP LLM Proxy Service

Responsibilities:

- Implement MCP (Model Context Protocol) for LLM communication
- Manage prompt engineering and system instructions
- Handle streaming responses from LLM
- Implement retry logic and error handling
- Monitor token usage and costs
- Provide fallback responses for service unavailability

System Prompt Template:

```
You are ESG Copilot, an AI assistant for the Sweep sustainability platform.

Context:
- User Role: {user_role}
- Current Page: {current_page}
- Organization: {organization_name}
- Recent Actions: {recent_actions}

Capabilities:
- Explain ESG compliance requirements (CSRD, SFDR, ISSB, GRI, CDP)
- Interpret data quality scores and anomaly flags
- Summarize dashboards and reports
- Guide users through platform features
- Explain audit trails and compliance status

Guidelines:
- Provide clear, concise responses in business language
- Reference specific Sweep features when relevant
- Suggest next actions when appropriate
- Admit uncertainty rather than guess
- Maintain data privacy - never expose data from other organizations
```

2.2.5 Data Context & Retrieval Layer

Responsibilities:

- Retrieve user profile, permissions, and preferences
- Fetch relevant ESG data based on query context
- Access audit logs and activity history
- Search help documentation and regulatory guidance
- Provide data summaries for LLM context

Data Sources:

python

```
data_sources = {
    'user_profile': UserProfileService,
    'esg_data': ESGDataManagementService,
    'audit_logs': AuditTrailService,
    'help_docs': DocumentationService,
    'compliance_frameworks': RegulatoryKnowledgeBase,
    'dashboard_state': DashboardStateService
}
```

2.3 Data Flow Sequence

Example: User asks "Why was this emission flagged as anomalous?"

```
1. User Input
   └──> Frontend Chat Widget captures query

2. Context Gathering
   └──> Frontend sends: query + page context + user ID + metric ID

3. Authentication & Validation
   └──> API Gateway validates JWT, checks permissions

4. Session Management
   └──> Copilot Service retrieves/creates chat session

5. Context Enrichment
   └──> Copilot Service enriches with:
     - User role: ESG_ANALYST
     - Current page: /metrics/detail/metric-abc123
     - Metric data: {value: 8900, historical_avg: 1200}
     - Anomaly details: {score: 0.89, confidence: 0.94}

6. Data Retrieval
   └──> Fetch anomaly detection result from AI Service
   └──> Fetch metric history from ESG Data Service

7. MCP LLM Call
   └──> MCP Proxy constructs prompt with enriched context
   └──> Sends to LLM via MCP protocol
   └──> Receives streaming response

8. Response Processing
   └──> Format response with markdown
   └──> Add suggested follow-up prompts
   └──> Include relevant links

9. Audit Logging
   └──> Log query, response, data accessed, timestamp

10. Frontend Display
    └──> Stream response to user in real-time
    └──> Display suggested actions
```

---

3. Data Privacy & Security

3.1 Security Architecture

3.1.1 Authentication & Authorization

JWT Token Structure:

json

```json
{
  "sub": "user-123",
  "org_id": "org-456",
  "role": "ESG_ANALYST",
  "permissions": [
    "read:esg_data",
    "read:reports",
    "read:audit_logs"
  ],
  "data_scope": {
    "entities": ["entity-dubai-hq", "entity-abu-dhabi-plant"],
    "geographies": ["UAE", "KSA"]
  },
  "exp": 1735689600,
  "iat": 1735603200
}
```

Permission Enforcement:

- All queries check user permissions before data retrieval
- LLM context limited to user's data scope
- Responses filtered based on role-based access control (RBAC)
- Cross-organization data leakage prevented by org_id validation

3.1.2 Data Scoping

Principle: Users can only query data they have permission to view in the platform.

Implementation:

python

```
async def enforce_data_scope(query_result, user_context):
    # Filter results by user's data scope
    allowed_entities = user_context.data_scope.entities
    allowed_geographies = user_context.data_scope.geographies

    filtered_result = [
        record for record in query_result
        if record.entity_id in allowed_entities
        and record.geography in allowed_geographies
    ]

    return filtered_result
```

Examples:

- ESG Analyst in Dubai office: Can only see Dubai facility data
- Sustainability Manager: Can see all facility data within organization
- Auditor: Read-only access to audit logs and reports, no raw data manipulation
- Supplier User: Can only see own supplier's data and submissions

3.2 Privacy Protections

3.2.1 Data Minimization

Strategy: Only send necessary context to LLM, not entire datasets.

Before LLM Call:

python

```python
# Bad - Sends entire dataset
context = {
    'all_emissions_data': fetch_all_emissions()  # 50,000 records
}

# Good - Sends only relevant summary
context = {
    'metric_summary': {
        'current_value': 8900,
        'historical_avg': 1200,
        'anomaly_score': 0.89,
        'flag_reason': 'Statistical outlier (6.8σ above mean)'
    }
}
```

3.2.2 PII Protection

Rules:

- Never send personally identifiable information (PII) to LLM
- Anonymize user names in context ("User submitted data" vs. "John Doe submitted")
- Redact sensitive fields before LLM processing
- Hash identifiers when needed for tracking

Redaction Example:

python

```python
def redact_sensitive_fields(data):
    sensitive_fields = ['email', 'phone', 'address', 'employee_id']
    for field in sensitive_fields:
        if field in data:
            data[field] = '[REDACTED]'
    return data
```

3.2.3 Audit Logging

All Copilot interactions logged:

json

```json
{
 "log_id": "log-123abc",
 "timestamp": "2025-09-24T16:30:00Z",
 "user_id": "user-123",
 "organization_id": "org-456",
 "session_id": "session-789",
 "query": "Why was this emission flagged?",
 "intent": "EXPLAIN_ANOMALY",
 "data_accessed": ["metric-abc123", "anomaly-result-def456"],
 "response_summary": "Explained statistical outlier anomaly",
 "tokens_used": 450,
 "latency_ms": 1250,
 "success": true
}
```

3.3 Rate Limiting & Abuse Prevention

Limits:

- 100 queries per hour per user
- 1,000 queries per hour per organization
- Max 2,000 characters per query
- Max 10 concurrent sessions per user
- 24-hour conversation history retention

Abuse Detection:

- Flag queries attempting to extract bulk data
- Detect prompt injection attempts
- Monitor for unusual query patterns
- Alert security team for potential attacks

---

4. Product Features

4.1 MVP Feature Set

Feature 1: Compliance Guidance

Description: Answer questions about ESG regulations and reporting requirements

Example Prompts:

- "What is CSRD and do we need to comply?"
- "Explain ESRS E1 Climate Change requirements"
- "What's the difference between Scope 2 location-based and market-based?"
- "When is our CSRD reporting deadline?"
- "What are the mandatory disclosures for GRI 305 Emissions?"

Response Capabilities:

- Plain-language explanations of regulatory requirements
- Specific applicability to user's organization
- Deadline tracking and reminders
- Links to official regulatory guidance
- Comparison between different frameworks

Success Criteria:

- 80% of compliance questions answered without escalation
- Average response time <3 seconds
- User satisfaction rating >4.0/5.0

Feature 2: Data Validation Assistant

Description: Explain AI-flagged anomalies, quality scores, and data issues

Example Prompts:

- "Why was this emission record flagged as anomalous?"
- "What does a data quality score of 87.3 mean?"
- "How do I fix this data validation error?"
- "Why does the system think this is a duplicate?"
- "Explain the confidence interval on this imputed value"

Response Capabilities:

- Detailed explanations of AI detection logic
- Statistical context (standard deviations, historical comparisons)
- Step-by-step remediation guidance
- Impact assessment ("This affects Q3 Scope 1 total by 2.3%")
- Links to relevant data entry forms

Success Criteria:

- 70% of data validation queries resolved without manual review
- 50% reduction in support tickets about anomaly flags
- 90% user understanding of data quality scores (measured by follow-up questions)

Feature 3: Dashboard Summarization

Description: Provide natural language summaries of complex dashboards and charts

Example Prompts:

- "Summarize my emissions dashboard"
- "What are the key trends in Q3 energy consumption?"
- "Which facilities are highest emitters?"
- "How are we tracking against our 2025 targets?"
- "Explain the changes since last month"

Response Capabilities:

- Extract key insights from dashboard data
- Highlight significant changes and trends
- Identify anomalies or areas of concern
- Provide comparative analysis (vs. targets, previous periods, peers)
- Suggest focus areas for improvement

Success Criteria:

- Dashboard summaries generated in <5 seconds
- 75% of users find summaries actionable
- 30% reduction in time spent analyzing dashboards

Feature 4: Audit Trail Interpretation

Description: Explain audit logs, data lineage, and compliance documentation

Example Prompts:

- "Show me the audit trail for this metric"
- "Who made changes to our Q2 emissions data?"
- "Explain the data lineage for this calculated metric"
- "What evidence supports our CSRD compliance?"
- "When was this report last approved?"

Response Capabilities:

- Translate audit log entries into readable narratives
- Explain data transformations and calculations
- Show approval workflow status
- Identify gaps in documentation
- Provide compliance evidence summaries

Success Criteria:

- 85% of audit queries answered comprehensively
- 40% reduction in time spent preparing for audits
- 95% auditor satisfaction with documentation transparency

Feature 5: Onboarding Assistant

Description: Guide new users through platform setup and initial workflows

Example Prompts:

- "How do I get started with Sweep?"
- "What's the next step for setting up my organization?"
- "How do I upload emissions data?"
- "Walk me through creating my first report"
- "What should I do after data upload?"

Response Capabilities:

- Personalized onboarding path based on user role
- Step-by-step guided tutorials
- Progress tracking ("You've completed 3 of 7 setup steps")
- Contextual help based on current page
- Proactive suggestions for next actions

Success Criteria:

- New users complete onboarding in <3 hours (vs. 6 hours baseline)
- 90% onboarding completion rate
- 60% of new users reach "first value" milestone within 1 day

---

Feature 6: Feature Discovery & Navigation

Description: Help users find and use platform features

Example Prompts:

- "How do I generate a CSRD report?"
- "Where can I see supplier emissions?"
- "Show me how to set reduction targets"
- "How do I invite team members?"
- "Where are the emission factor settings?"

Response Capabilities:

- Direct navigation links to relevant pages
- Feature explanations with screenshots (future)
- Permission-aware guidance ("This feature requires Admin role")
- Alternative path suggestions if feature unavailable
- Quick action buttons for common tasks

Success Criteria:

- 80% of navigation queries result in successful feature use
- 25% reduction in "how to" support tickets
- 70% of users discover 3+ features through Copilot

4.2 Suggested Prompts (Contextual)

On Dashboard Page:

- "Summarize this dashboard"
- "What changed since last month?"
- "Which metrics need attention?"

On Data Upload Page:

- "What file format should I use?"
- "How do I map my columns?"
- "What happens after upload?"

On Report Generation Page:

- "Which template should I choose?"
- "What data is required for this report?"
- "How long does generation take?"

On Audit Trail Page:

- "Explain these changes"
- "Who approved this data?"
- "Show me the data lineage"

---

5. API Contracts

5.1 Chat Query Endpoint

POST /api/v1/copilot/chat

Purpose: Send user query to ESG Copilot and receive response

Request Schema:

json

```json
{
  "session_id": "string (optional, creates new if not provided)",
  "query": "string (required, max 2000 chars)",
  "context": {
    "page": "string (current page URL path)",
    "page_type": "enum (dashboard|report|data_entry|audit|settings)",
    "entity_ids": ["array of entity IDs visible on page"],
    "metric_ids": ["array of metric IDs in context"],
    "report_id": "string (if on report page)",
    "user_action": "string (recent action, e.g., 'uploaded_data')"
  },
  "options": {
    "streaming": "boolean (default: true)",
    "include_suggestions": "boolean (default: true)",
    "max_tokens": "integer (default: 500)"
  }
}
```

Request Example:

json

```json
{
  "session_id": "session-789abc",
  "query": "Why was metric-abc123 flagged as anomalous?",
  "context": {
    "page": "/metrics/detail/metric-abc123",
    "page_type": "data_entry",
    "metric_ids": ["metric-abc123"],
    "entity_ids": ["entity-dubai-hq"]
  },
  "options": {
    "streaming": true,
    "include_suggestions": true
  }
}
```

Response Schema (Non-Streaming):

json

```
{
 "session_id": "string",
 "message_id": "string",
 "response": {
  "text": "string (markdown formatted)",
  "confidence": "float (0-1)",
  "intent": "enum (explain|summarize|navigate|guide)",
  "data_sources": ["array of data sources used"],
  "suggested_prompts": ["array of 3 follow-up suggestions"],
  "actions": [
   {
    "type": "navigation|data_action|report_action",
    "label": "string",
    "url": "string (optional)",
    "api_call": "object (optional)"
   }
  ]
 },
 "metadata": {
  "tokens_used": "integer",
  "latency_ms": "integer",
  "cached": "boolean"
 },
 "timestamp": "ISO 8601 datetime"
}
```

Response Example:

json

```json
{
  "session_id": "session-789abc",
  "message_id": "msg-456def",
  "response": {
    "text": "This emission record was flagged as **anomalous** because:\n\n1. **Statistical Outlier**: The value (8,90
    "confidence": 0.94,
    "intent": "explain",
    "data_sources": [
      "ai_cleansing_service:anomaly_detection",
      "esg_data_service:metric_history"
    ],
    "suggested_prompts": [
      "Show me the historical trend for this facility",
      "How do I correct this value?",
      "What impact does this have on my Q3 report?"
    ],
    "actions": [
      {
        "type": "navigation",
        "label": "View Historical Data",
        "url": "/metrics/history/metric-abc123"
      },
      {
        "type": "data_action",
        "label": "Flag for Review",
        "api_call": {
          "method": "POST",
          "endpoint": "/api/v1/metrics/metric-abc123/flag",
          "payload": {"reason": "anomaly_review"}
        }
      }
    ]
  },
  "metadata": {
    "tokens_used": 450,
    "latency_ms": 1250,
    "cached": false
  },
  "timestamp": "2025-09-24T16:30:15Z"
}
```

Response Schema (Streaming via Server-Sent Events):

```
event: token
data: {"text": "This emission record was flagged as ", "session_id": "session-789abc"}

event: token
data: {"text": "**anomalous** because:\n\n", "session_id": "session-789abc"}

event: token
data: {"text": "1. **Statistical Outlier**: The value...", "session_id": "session-789abc"}

event: complete
data: {"session_id": "session-789abc", "message_id": "msg-456def", "metadata": {...}}
```

Error Responses:

json

```
// 400 Bad Request - Invalid query
{
  "error": {
    "code": "INVALID_QUERY",
    "message": "Query exceeds maximum length of 2000 characters"
  }
}

// 401 Unauthorized
{
  "error": {
    "code": "UNAUTHORIZED",
    "message": "Invalid or expired authentication token"
  }
}

// 429 Rate Limit Exceeded
{
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Query limit of 100 per hour exceeded",
    "retry_after": 3600
  }
}

// 503 Service Unavailable
{
  "error": {
    "code": "LLM_SERVICE_UNAVAILABLE",
    "message": "AI service temporarily unavailable, please try again",
    "fallback_response": "I'm currently unavailable. Please try again in a moment or contact support for immediate assis
  }
}
```

5.2 Session Management Endpoints

GET /api/v1/copilot/sessions/{session_id}

Purpose: Retrieve conversation history for a session

Response:

json

```json
{
  "session_id": "session-789abc",
  "user_id": "user-123",
  "created_at": "2025-09-24T15:00:00Z",
  "last_activity": "2025-09-24T16:30:15Z",
  "message_count": 12,
  "messages": [
    {
      "message_id": "msg-001",
      "role": "user",
      "content": "What is CSRD?",
      "timestamp": "2025-09-24T15:05:00Z"
    },
    {
      "message_id": "msg-002
```