# Arctic Tracker Sprint Plan - August 2025

## 🎯 Sprint Overview

**Sprint Name**: Backend Data Processing & Frontend Infrastructure Sprint  
**Duration**: August 2025 (Full Month)  
**Sprint Goal**: Complete all species data processing, establish data quality standards, migrate to professional domain, and implement verification systems

## 🌐 PRIORITY 1: Domain Migration to ArcticTracker.thearctic.is

### Objectives:

* Migrate from GitHub Pages to professional hosting
* Set up custom domain with Arctic-specific TLD (.is - Iceland)
* Ensure zero downtime during migration
* Implement proper redirects and SEO preservation
* Establish Arctic regional identity with .is domain

### Benefits of ArcticTracker.thearctic.is:

* **Regional Authority**: .is (Iceland) domain reinforces Arctic connection
* **Professional Branding**: Move from github.io to dedicated domain
* **Improved SEO**: Custom domain ranks better than subdomain
* **Better Performance**: Professional hosting vs GitHub Pages limitations
* **Enhanced Trust**: Official domain for research credibility

### Tasks:

* **Domain & Hosting Setup**
  + Register ArcticTracker.thearctic.is domain
  + Set up hosting infrastructure (recommend Vercel/Netlify for React apps)
  + Configure DNS records
  + SSL certificate setup (auto-provisioned)
* **Migration Steps**
  + Deploy application to new hosting
  + Test thoroughly on new domain
  + Set up 301 redirects from old GitHub Pages URL
  + Update all internal links and references
  + Update sitemap.xml
  + Submit change of address in Google Search Console
* **Configuration Updates**
  + Update vite.config.ts base URL
  + Update environment variables
  + Update CORS settings in Supabase
  + Update OAuth redirect URLs if applicable
* **Post-Migration**
  + Monitor 404 errors
  + Check all external integrations
  + Update documentation with new URL
  + Notify stakeholders of domain change
  + Update social media profiles

## 📋 Sprint Backlog

### 1. CI/CD Pipeline & Branching Strategy 🔧

#### Objectives:

* Implement industry-standard branching strategy
* Set up automated CI/CD workflows
* Establish environment separation (dev/test/prod)

#### Tasks:

* **Create Branching Structure**
  + main → Production (protected)
  + staging → Pre-production testing
  + develop → Integration branch
  + feature/\* → Feature branches
  + hotfix/\* → Emergency fixes
* **GitHub Actions Workflows**
* # .github/workflows/ci.yml  
  - Automated testing on PR  
  - TypeScript compilation checks  
  - ESLint/Prettier validation  
  - Build verification
* **Environment Configuration**
  + Development: Local development
  + Staging: staging.ArcticTracker.thearctic.is (subdomain)
  + Production: ArcticTracker.thearctic.is
* **Deployment Automation**
  + Auto-deploy develop → staging environment
  + Manual approval for staging → production
  + Rollback procedures
* **Branch Protection Rules**
  + Require PR reviews for main/staging
  + Require status checks to pass
  + Dismiss stale reviews
  + Include administrators

### 2. Landing Page Enhancement 🏠

#### Current Issues:

* Placeholder “Integration and Tools” section
* Static featured species selection
* Generic content that doesn’t showcase real capabilities

#### Tasks:

* **Remove Placeholder Content**
  + Delete “Integration and Tools” section
  + Remove mock integration cards
* **Add Real Value Sections**
  + **Live Statistics Dashboard**
  + - Total species monitored  
    - Species by conservation status  
    - Recent CITES trade alerts  
    - Data last updated timestamp
  + **Research Highlights**
    - Featured research findings
    - Key conservation insights
    - Link to publications
  + **Data Sources & Partners**
    - CITES Trade Database
    - IUCN Red List
    - NAMMCO
    - Arctic Council affiliations
  + **Recent Updates Feed**
    - Latest species assessments
    - New data additions
    - System updates
* **Dynamic Featured Species**
  + Implement weighted selection based on:
    - Conservation priority (CR/EN species)
    - Recent data updates
    - Seasonal relevance
    - Research focus

### 3. Human Review Verification System ✅

#### Objectives:

* Clear indication of data verification status
* Transparency in data quality
* Build trust with researchers

#### Database Schema Updates:

-- Add to species table  
ALTER TABLE species ADD COLUMN review\_status VARCHAR(50) DEFAULT 'pending';  
ALTER TABLE species ADD COLUMN reviewed\_by UUID REFERENCES profiles(id);  
ALTER TABLE species ADD COLUMN reviewed\_at TIMESTAMP;  
ALTER TABLE species ADD COLUMN review\_notes TEXT;  
  
-- Create review history table  
CREATE TABLE species\_review\_history (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 species\_id UUID REFERENCES species(id),  
 reviewer\_id UUID REFERENCES profiles(id),  
 action VARCHAR(50), -- 'approved', 'rejected', 'flagged'  
 notes TEXT,  
 created\_at TIMESTAMP DEFAULT NOW()  
);

#### UI Implementation:

* **Species Cards**
  + Add verification badge (✓ Verified / ⚠️ Pending / ❌ Needs Review)
  + Show “Last reviewed by [Expert Name] on [Date]”
* **Species Detail Pages**
  + Prominent verification status banner
  + Review history timeline
  + Data quality indicators
* **Gallery Filters**
  + Add “Verification Status” filter
  + Option to show only verified species
* **Admin Panel**
  + Review queue for pending species
  + Batch approval interface
  + Review statistics dashboard

## 💡 Additional Suggestions

### 4. Data Quality Dashboard 📊

* Create dedicated page showing:
  + Data completeness metrics per species
  + Missing data fields analysis
  + Source attribution statistics
  + Update frequency tracking

### 5. API Development 🔌

* Public REST API for researchers
  + Rate-limited endpoints
  + API key management
  + Documentation with examples
  + Usage analytics

### 6. Progressive Web App (PWA) 📱

* Implement PWA features:
  + Offline species browsing
  + Install prompts
  + Push notifications for data updates
  + Responsive images with srcset

### 7. Export Functionality 📥

* Multiple export formats:
  + CSV for data analysis
  + PDF species reports
  + Citation-ready formats
  + Bulk data downloads

### 8. Community Features 👥

* Researcher profiles
* Data contribution system
* Species observation reports
* Discussion threads per species

### 9. Advanced Search 🔍

* Search improvements:
  + Search by geographic range
  + Trade route filtering
  + Multi-criteria search
  + Search history

### 10. Performance Monitoring 📈

* Implement analytics:
  + Google Analytics 4
  + Error tracking (Sentry)
  + Performance monitoring
  + User behavior insights

## 🏗️ Technical Debt to Address

1. **Code Splitting**
   * Current bundle size: 1.35 MB
   * Implement React.lazy() for routes
   * Dynamic imports for heavy components
2. **Testing Suite**
   * Add Jest configuration
   * Component unit tests
   * Integration tests for API calls
   * E2E tests with Playwright
3. **Documentation**
   * API documentation
   * Component storybook
   * Deployment guide
   * Contributing guidelines

## 📅 August 2025 Sprint Schedule

### Week 1 (Aug 1-7): Backend Data Processing

**Day 1-3**: Complete Species Data Processing - Identify remaining species - Process through Scite AI - Quality control checks

**Day 4-5**: Database Migration & Loading - Create comprehensive backups - Run data validation scripts - Execute migration to production

**Day 6-7**: Data Quality Inspection - Performance analysis - Integrity verification - Quality scoring implementation

### Week 2 (Aug 8-14): Backend Standards & Domain Setup

**Day 8-9**: Security & Monitoring - Implement security enhancements - Setup monitoring stack - Configure logging

**Day 10-11**: Documentation & Compliance - API documentation - Data dictionary - FAIR principles implementation

**Day 12-14**: Domain Migration - Register ArcticTracker.thearctic.is - Configure hosting and DNS - Test migration process

### Week 3 (Aug 15-21): Frontend Infrastructure

**Day 15-16**: CI/CD Pipeline - GitHub Actions setup - Environment configuration - Branch protection rules

**Day 17-19**: Landing Page Enhancement - Remove placeholder content - Add live statistics dashboard - Implement dynamic featured species

**Day 20-21**: Review System Backend - Database schema updates - Create review history tables - API endpoints for reviews

### Week 4 (Aug 22-31): Integration & Launch

**Day 22-24**: Review System Frontend - UI components for verification - Admin panel implementation - Gallery filters

**Day 25-26**: Testing & Optimization - Integration testing - Performance optimization - Security audit

**Day 27-28**: Production Deployment - Final domain migration - Deploy all features - Monitor for issues

**Day 29-31**: Documentation & Handoff - Update all documentation - Create maintenance guides - Sprint retrospective

## 🎯 Definition of Done

* All code reviewed and approved
* Tests written and passing
* Documentation updated
* Deployed to staging for testing
* No critical bugs
* Performance benchmarks met

## 📊 Success Metrics

* CI/CD pipeline running successfully
* Zero manual deployment steps
* 100% of species showing verification status
* Landing page load time < 2 seconds
* Positive user feedback on data transparency

## 🚀 Future Considerations

1. **Mobile App Development**
   * React Native implementation
   * Offline-first architecture
   * Field research tools
2. **Machine Learning Integration**
   * Species identification from photos
   * Trade pattern analysis
   * Population trend predictions
3. **Blockchain for Data Integrity**
   * Immutable audit trail
   * Decentralized verification
   * Research attribution

## 🔧 Backend Data Processing & Quality Sprint

### Sprint Overview

**Focus**: Complete species data processing, database migration, and establish data quality standards  
**Priority**: CRITICAL - Foundation for all frontend work  
**Duration**: Week 1-2 of August 2025 (Aug 1-14)

### 1. Complete Species Data Processing with Scite AI 🔬

#### Current Status:

* 14 species profiles completed and uploaded
* Remaining Arctic species need processing
* Scite AI workflow established but needs scaling

#### Tasks:

* **Identify Remaining Species**
* -- Query to find species without detailed profiles  
  SELECT s.id, s.scientific\_name, s.common\_name\_en  
  FROM species s  
  WHERE s.is\_arctic = true  
  AND (s.description IS NULL OR s.habitat IS NULL OR s.diet IS NULL)  
  ORDER BY s.conservation\_priority DESC;
* **Batch Processing Pipeline**
  + Create species processing queue
  + Implement rate limiting for Scite AI API
  + Add progress tracking and logging
  + Handle API failures gracefully
* **Data Collection Checklist per Species**
  + Scientific literature review (minimum 10 papers)
  + Population data and trends
  + Habitat and distribution maps
  + Diet and behavioral information
  + Conservation threats
  + Climate change impacts
  + Indigenous knowledge integration
  + Recent CITES trade patterns
* **Quality Control Process**
  + Cross-reference with IUCN Red List
  + Verify citation accuracy
  + Check for data inconsistencies
  + Validate geographic coordinates
  + Ensure minimum data completeness (80%)

### 2. Database Migration & Data Loading 📊

#### Pre-Migration Tasks:

* **Create Comprehensive Backup**
* # Full database backup with timestamp  
  pg\_dump -h [host] -U [user] -d [database] > arctic\_tracker\_backup\_$(date +%Y%m%d\_%H%M%S).sql  
    
  # Backup to cloud storage (S3/Google Cloud)  
  aws s3 cp arctic\_tracker\_backup\_\*.sql s3://arctic-tracker-backups/
* **Data Validation Scripts**
* # validate\_species\_data.py  
  - Check for NULL values in required fields  
  - Validate scientific name format  
  - Ensure IUCN status codes are valid  
  - Verify reference URLs are accessible  
  - Check image URLs return 200 status

#### Migration Process:

* **Staging Environment Setup**
  + Clone production database to staging
  + Test all migrations on staging first
  + Run performance benchmarks
* **Data Enhancement Migration**
* -- Add new fields for comprehensive profiles  
  ALTER TABLE species ADD COLUMN IF NOT EXISTS   
   morphology JSONB,  
   reproductive\_info JSONB,  
   migration\_patterns JSONB,  
   cultural\_significance TEXT,  
   research\_priorities TEXT[],  
   data\_quality\_score INTEGER DEFAULT 0,  
   last\_comprehensive\_update TIMESTAMP;
* **Upload Enhanced Species Data**
  + Use batch processing (500 records at a time)
  + Implement transaction rollback on errors
  + Log all changes for audit trail
  + Update search indexes

### 3. Data Architecture Quality Inspection 🔍

#### Database Health Checks:

* **Performance Analysis**
* -- Identify slow queries  
  SELECT query, calls, mean\_exec\_time, total\_exec\_time  
  FROM pg\_stat\_statements  
  WHERE mean\_exec\_time > 100  
  ORDER BY mean\_exec\_time DESC;  
    
  -- Check index usage  
  SELECT schemaname, tablename, indexname, idx\_scan  
  FROM pg\_stat\_user\_indexes  
  ORDER BY idx\_scan;
* **Data Integrity Verification**
  + Foreign key constraint validation
  + Check for orphaned records
  + Validate data type consistency
  + Ensure no duplicate species entries
* **Storage Optimization**
* -- Table size analysis  
  SELECT relname, pg\_size\_pretty(pg\_total\_relation\_size(relid))  
  FROM pg\_stat\_user\_tables  
  ORDER BY pg\_total\_relation\_size(relid) DESC;  
    
  -- Vacuum and analyze tables  
  VACUUM ANALYZE species;  
  VACUUM ANALYZE cites\_trade\_records;

#### Data Quality Metrics:

* **Implement Quality Scoring System**
* def calculate\_species\_quality\_score(species\_data):  
   score = 0  
   # Completeness checks (40 points)  
   if species\_data.description: score += 5  
   if species\_data.habitat: score += 5  
   if species\_data.diet: score += 5  
   if species\_data.conservation\_measures: score += 5  
   if species\_data.population\_data: score += 10  
   if species\_data.threat\_assessment: score += 10  
    
   # Data freshness (30 points)  
   last\_update = species\_data.last\_comprehensive\_update  
   if last\_update > (now - 6\_months): score += 30  
   elif last\_update > (now - 1\_year): score += 20  
   elif last\_update > (now - 2\_years): score += 10  
    
   # Reference quality (30 points)  
   if species\_data.reference\_count > 20: score += 30  
   elif species\_data.reference\_count > 10: score += 20  
   elif species\_data.reference\_count > 5: score += 10  
    
   return score

### 4. Backup Strategy & Disaster Recovery 💾

#### Comprehensive Backup Plan:

* **Automated Daily Backups**
* # backup-cron.yaml  
  schedule: "0 2 \* \* \*" # 2 AM daily  
  tasks:  
   - Full database backup  
   - Incremental file backups  
   - Upload to cloud storage  
   - Retention: 30 days daily, 12 months weekly
* **Backup Verification**
  + Weekly restore tests to dev environment
  + Checksum validation
  + Document restore procedures
  + Time-to-restore benchmarks
* **Multi-Region Redundancy**
  + Primary: Supabase automatic backups
  + Secondary: AWS S3 versioned bucket
  + Tertiary: Local encrypted backups

### 5. Industry Best Standards Implementation 🏆

#### Security Enhancements:

* **API Security**
  + Implement rate limiting per endpoint
  + Add request signing for sensitive operations
  + Enable audit logging for all data modifications
  + Implement field-level encryption for PII
* **Access Control**
* -- Row Level Security policies  
  CREATE POLICY species\_read\_public ON species  
   FOR SELECT USING (true);  
    
  CREATE POLICY species\_write\_admin ON species  
   FOR ALL USING (auth.jwt() ->> 'role' = 'admin');

#### Monitoring & Observability:

* **Setup Monitoring Stack**
  + Database metrics (connections, query time, storage)
  + API endpoint performance
  + Error tracking and alerting
  + Uptime monitoring
* **Logging Standards**
* // Structured logging format  
  logger.info({  
   action: 'species\_update',  
   species\_id: uuid,  
   user\_id: auth.user\_id,  
   changes: diff,  
   timestamp: new Date().toISOString()  
  });

#### Documentation:

* **API Documentation**
  + OpenAPI/Swagger specification
  + Authentication guide
  + Rate limit documentation
  + Example requests/responses
* **Data Dictionary**
  + Document all table schemas
  + Field descriptions and constraints
  + Relationships and dependencies
  + Data source attribution

#### Compliance & Standards:

* **FAIR Data Principles**
  + **F**indable: Persistent identifiers, rich metadata
  + **A**ccessible: Standard protocols, authentication
  + **I**nteroperable: Standard vocabularies, qualified references
  + **R**eusable: Clear licenses, provenance, domain standards
* **Conservation Data Standards**
  + Darwin Core compliance for biodiversity data
  + IUCN Red List Categories and Criteria
  + CITES trade data standards
  + ISO 8601 date/time formats

### 6. Performance Optimization 🚀

* **Query Optimization**
* -- Create composite indexes for common queries  
  CREATE INDEX idx\_species\_arctic\_status   
   ON species(is\_arctic, conservation\_status)  
   WHERE is\_arctic = true;  
    
  CREATE INDEX idx\_trade\_species\_year   
   ON cites\_trade\_records(species\_id, year DESC);
* **Caching Strategy**
  + Redis for frequently accessed species
  + CDN for static assets
  + Database query result caching
  + API response caching with ETags

### 7. Data Pipeline Automation 🔄

* **ETL Pipeline for Updates**
* # automated\_data\_updates.py  
  - Daily IUCN status checks  
  - Weekly CITES trade data sync  
  - Monthly literature review updates  
  - Automated quality score recalculation
* **Change Detection System**
  + Monitor source databases for updates
  + Automated notifications for significant changes
  + Version control for species data
  + Rollback capabilities

### Sprint Deliverables Checklist ✅

* All Arctic species processed through Scite AI
* Complete database with 100% species coverage
* Automated backup system operational
* Data quality scores for all species
* Performance benchmarks documented
* Security audit completed
* API documentation published
* Monitoring dashboards configured
* Disaster recovery plan tested
* Compliance documentation complete

### Success Metrics 📈

* **Data Completeness**: >95% of fields populated for all species
* **Quality Scores**: Average score >75/100 across all species
* **Performance**: All queries <100ms response time
* **Reliability**: 99.9% uptime target
* **Backup Success**: 100% successful daily backups
* **Security**: Zero vulnerabilities in security audit

### Post-Sprint Maintenance Plan 🛠️

1. **Weekly Tasks**
   * Review data quality reports
   * Check backup integrity
   * Monitor performance metrics
   * Process new literature updates
2. **Monthly Tasks**
   * Security patch updates
   * Performance optimization review
   * Data source synchronization
   * Quality score recalculation
3. **Quarterly Tasks**
   * Full security audit
   * Disaster recovery drill
   * Schema optimization review
   * Stakeholder data quality review

*Sprint Planning Date: July 18, 2025*  
*Sprint Duration: August 1-31, 2025*  
*Backend Focus: Week 1-2 (Aug 1-14)*  
*Frontend Focus: Week 3-4 (Aug 15-31)*  
*Sprint Retrospective: September 1, 2025*