GBIF Explorer - Project Re-engineering Plan

Project Overview

Transform `facesofplants` into a modern AI-powered biodiversity discovery platform that makes GBIF data accessible to both citizen scientists and researchers through natural language queries.

Technical Architecture

Frontend Stack (Next.js 14+)

```
gbif-explorer/
├── app/                    # App Router (Next.js 14+)
│   ├── (dashboard)/        # Dashboard routes
│   │   ├── discover/       # Main exploration interface
│   │   ├── collections/    # User saved collections
│   │   └── insights/       # Data analytics view
│   ├── api/                # API routes
│   │   ├── query/          # AI query processing
│   │   ├── gbif/           # GBIF API proxy
│   │   └── auth/           # Authentication
│   ├── globals.css
│   ├── layout.tsx
│   └── page.tsx
├── components/
│   ├── ui/                 # Reusable UI components
│   ├── exploration/        # Discovery interface components
│   ├── results/            # Results display components
│   └── shared/             # Common components
├── lib/
│   ├── ai/                 # AI/LLM integration
│   ├── gbif/               # GBIF API client
│   ├── aws/                # AWS SDK configurations
│   └── utils/              # Utility functions
├── hooks/                  # Custom React hooks
├── types/                  # TypeScript definitions
└── public/                 # Static assets
```

AWS Infrastructure

Core Services

1. **AWS Amplify** - Frontend hosting and CI/CD
2. **Amazon Bedrock** - AI/LLM services for query processing
3. **AWS Lambda** - Serverless backend functions

4. **Amazon API Gateway** - API management and rate limiting
5. **Amazon DynamoDB** - User data, query history, collections
6. **Amazon S3** - Static assets, cached results, species images
7. **Amazon CloudFront** - CDN for global performance
8. **Amazon Cognito** - User authentication and authorization

Advanced Services

1. **Amazon OpenSearch** - Enhanced search capabilities
2. **AWS Step Functions** - Complex query orchestration
3. **Amazon EventBridge** - Event-driven architecture
4. **Amazon SQS** - Queue management for heavy queries
5. **AWS X-Ray** - Performance monitoring and debugging

Migration Strategy

Phase 1: Foundation (Weeks 1-2)

- [ ] Initialize Next.js 14 project with TypeScript
- [ ] Set up AWS Amplify for hosting
- [ ] Configure basic AWS services (Cognito, DynamoDB)
- [ ] Implement basic UI components from the React prototype
- [ ] Set up GBIF API integration

Phase 2: Core Features (Weeks 3-4)

- [ ] Integrate Amazon Bedrock for AI query processing
- [ ] Implement dual-interface (citizen/researcher modes)
- [ ] Build natural language query parser
- [ ] Create results visualization components
- [ ] Add user authentication and profiles

Phase 3: Advanced Features (Weeks 5-6)

- [ ] Implement collections and saved queries
- [ ] Add geolocation-based suggestions
- [ ] Build data export functionality for researchers
- [ ] Integrate real-time GBIF data caching
- [ ] Performance optimization and caching strategies

Phase 4: Polish & Launch (Weeks 7-8)

- [ ] Mobile responsiveness and PWA features

- [ ] Advanced analytics and insights
- [ ] SEO optimization
- [ ] Documentation and user guides
- [ ] Beta testing and feedback integration

Key Components to Preserve from Prototype

1. User Interface Components

typescript

```typescript
// Preserve the dual-mode toggle
const UserTypeToggle = () => { /* ... */ }

// Natural language search interface
const SearchInterface = () => { /* ... */ }

// Results display for both user types
const CitizenResults = () => { /* ... */ }
const ResearcherResults = () => { /* ... */ }

// Feature cards and landing elements
const FeatureCard = () => { /* ... */ }
```

2. Design System

- Glassmorphism aesthetic with backdrop blur
- Green-to-blue gradient color scheme
- Card-based layout with rounded corners
- Smooth transitions and micro-interactions

3. User Experience Patterns

- Example queries for onboarding
- Progressive disclosure of complexity
- Context-aware suggestions
- Visual feedback for loading states

AWS Services Integration

Amazon Bedrock Integration

typescript

```typescript
// lib/ai/bedrock-client.ts
import { BedrockRuntimeClient, InvokeModelCommand } from "@aws-sdk/client-bedrock-runtime";

export class BiodiversityQueryProcessor {
  async processNaturalLanguageQuery(query: string, userType: 'citizen' | 'researcher') {
    // Convert natural language to GBIF API parameters
    // Handle user context and preferences
    // Return structured query results
  }
}
```

GBIF API Integration

typescript

```typescript
// lib/gbif/client.ts
export class GBIFClient {
  async searchOccurrences(params: GBIFSearchParams) {
    // Proxy GBIF API calls through Next.js API routes
    // Handle rate limiting and caching
    // Transform data for frontend consumption
  }
}
```

DynamoDB Schema

typescript

```typescript
// User Collections
interface UserCollection {
  userId: string;
  collectionId: string;
  name: string;
  description: string;
  queries: SavedQuery[];
  species: string[];
  createdAt: string;
  updatedAt: string;
}

// Query History
interface QueryHistory {
  userId: string;
  queryId: string;
  query: string;
  userType: 'citizen' | 'researcher';
  results: any;
  timestamp: string;
}
```

Environment Configuration

Local Development

bash

```bash
# .env.local
NEXT_PUBLIC_AWS_REGION=us-east-1
NEXT_PUBLIC_USER_POOL_ID=
NEXT_PUBLIC_USER_POOL_CLIENT_ID=
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
GBIF_API_URL=https://api.gbif.org/v1
BEDROCK_MODEL_ID=anthropic.claude-3-sonnet-20240229-v1:0
```

Production Deployment

- Use AWS Amplify environment variables
- Configure IAM roles for service access
- Set up CloudWatch monitoring
- Enable AWS X-Ray tracing

Performance Considerations

Caching Strategy

1. **CloudFront**: Static assets and API responses
2. **DynamoDB**: Frequently accessed species data
3. **S3**: Generated visualizations and reports
4. **Browser**: Query results and user preferences

Optimization Techniques

1. **Image Optimization**: Next.js Image component with S3 integration
2. **Code Splitting**: Dynamic imports for heavy components
3. **Streaming**: Server-side rendering with streaming
4. **Edge Functions**: Geolocation-based content delivery

Monitoring & Analytics

Application Monitoring

- AWS CloudWatch for infrastructure metrics
- AWS X-Ray for request tracing
- Custom metrics for user engagement
- Error tracking with AWS CloudWatch Logs

User Analytics

- Query pattern analysis
- Popular species and regions
- User journey optimization
- A/B testing for interface improvements

Security Considerations

Data Protection

- End-to-end encryption for user data
- GDPR compliance for EU users
- Rate limiting on API endpoints
- Input validation and sanitization

Access Control

- Role-based access (citizen vs researcher)
- API key management for GBIF access
- Resource-level permissions in AWS
- Audit logging for compliance

Budget Estimation (Monthly)

AWS Services (Estimated)

- **Amplify Hosting**: $15-30
- **Lambda Functions**: $20-50
- **DynamoDB**: $25-75
- **Bedrock API Calls**: $100-300
- **S3 Storage**: $10-25
- **CloudFront**: $15-40
- **Other Services**: $25-50

Total Estimated: $210-570/month (scales with usage)

Success Metrics

User Engagement

- Daily/Monthly Active Users
- Query completion rates
- Time spent on platform
- Collection creation and sharing

Technical Performance

- API response times (<2s for queries)
- Error rates (<1%)
- Uptime (99.9%+)
- Cache hit rates (>80%)

Scientific Impact

- Data downloads by researchers
- Citations in scientific papers
- Species discovery contributions
- Educational usage statistics

Migration Commands

1. Project Initialization

bash

```
# Clone and setup new project
git clone https://github.com/juserr/facesofplants.git gbif-explorer
cd gbif-explorer
git checkout -b rewrite-v2

# Initialize Next.js with TypeScript
npx create-next-app@latest . --typescript --tailwind --eslint --app
npm install @aws-sdk/client-bedrock-runtime @aws-sdk/client-dynamodb aws-amplify
```

2. AWS Setup

bash

```
# Install and configure Amplify CLI
npm install -g @aws-amplify/cli
amplify configure
amplify init
amplify add auth
amplify add storage
amplify add function
```

3. Component Migration

bash

```
# Copy and adapt React components from prototype
cp prototype-components/* components/exploration/
# Update imports and add Next.js specific optimizations
```

This architecture provides a solid foundation for transforming your project into a modern, scalable, and user-friendly biodiversity discovery platform while preserving the innovative UI/UX concepts from our prototype.