Data Engineering Portfolio Website - Project Requirement Document

Project Owner: Mohanasundaram Murugesan
Version: 1.0
Date: February 2026
Status: Planning Phase

Executive Summary

This PRD outlines the development of an elite data engineering portfolio website that showcases technical expertise, project work, and professional experience while demonstrating real data engineering capabilities through live pipeline monitoring and analytics dashboards. The website will serve as both a portfolio and a working demonstration of data engineering skills, setting it apart from traditional static portfolio sites.

Key Differentiator: Live data pipeline monitoring dashboard integrated into the portfolio, showing real-time metrics from actual ETL workflows (GitHub activity, website analytics, or curated data feeds).

Project Goals & Success Metrics

Primary Goals

1. **Showcase Professional Experience** - Present 4+ years of data engineering experience with quantified impact metrics
2. **Demonstrate Technical Depth** - Display mastery of modern data stack (PySpark, Airflow, Kafka, Snowflake, AWS)
3. **Stand Out to Recruiters** - Create memorable, interactive experience that generates callbacks
4. **Prove Current Skills** - Show active engagement with latest tools and best practices

Success Metrics

- **10+ recruiter views per week** within first month
- **3+ project deep-dives** by interested hiring managers (tracked via analytics)
- **2+ interview callbacks** attributable to portfolio within 60 days
- **95+ PageSpeed score** for performance
- **<2 second load time** for initial page
- **Mobile-responsive** design (100% compatibility)

Target Audience

Primary Audience

- **Data Engineering Hiring Managers** at mid-to-large tech companies
- **Technical Recruiters** specializing in data roles
- **Engineering Leads** evaluating senior data engineer candidates

Secondary Audience

- **Peers & Networking Contacts** in data engineering community
- **Conference/Meetup Organizers** reviewing speaker applications
- **Academic Advisors** for graduate program recommendations

Audience Needs

- Quick assessment of technical skills (scanning time: 30-60 seconds)
- Evidence of production-ready capabilities (not just tutorials)
- Clear communication of complex technical concepts
- Proof of modern tooling expertise
- Demonstration of impact through metrics

Technical Architecture

Frontend Stack

Framework: Hugo (Static Site Generator)

- **Rationale:** Fast build times, no runtime dependencies, excellent SEO, markdown-based content management
- **Theme:** Modified version of "PaperMod" or "Stack" theme (developer-focused, clean design)
- **Customization:** Custom CSS for brand colors, interactive components via vanilla JavaScript

Key Technologies:

- HTML5, CSS3 (with CSS Grid & Flexbox)
- Vanilla JavaScript (ES6+) for interactivity
- Chart.js or D3.js for data visualizations
- Responsive design with mobile-first approach

Backend & Data Engineering Components

Live Dashboard Data Source (The Data Engineering Touch):

Option 1: GitHub Activity Pipeline (Recommended)

- **Data Source:** GitHub API (personal profile, repositories, commits)
- **Pipeline:** Scheduled GitHub Actions workflow (daily runs)
- **Processing:** Python script extracts commit history, repo stats, language usage
- **Storage:** GitHub Pages JSON files (versioned, tracked in repo)

- **Frontend Display:** JavaScript fetches JSON, renders interactive charts

Option 2: Portfolio Analytics Pipeline

- **Data Source:** Cloudflare Analytics API or Google Analytics
- **Pipeline:** Serverless function (Cloudflare Workers) fetches data daily
- **Processing:** Aggregates page views, visitor geography, popular pages
- **Storage:** Static JSON hosted alongside site
- **Frontend Display:** Real-time dashboard showing portfolio engagement

Architecture Diagram (GitHub Pipeline):
GitHub API → GitHub Actions (Daily Cron) → Python Script
↓
Process & Transform (PySpark concepts, pandas)
↓
Generate JSON (Medallion-inspired: raw → cleaned → aggregated)
↓
Commit to Repo → Deploy via GitHub Actions → Hugo Build
↓
GitHub Pages (Free Hosting) → Live Portfolio with Data Dashboard

Hosting & Deployment

Primary Hosting: GitHub Pages (Free, Custom Domain Support, HTTPS)

- **URL Structure:** `mohanasundaram.dev` or `mohanasundaram.tech`
- **Build Process:** GitHub Actions CI/CD pipeline
- **Deployment Trigger:** Push to `main` branch auto-deploys

Alternative Options:

- **Vercel** (Free tier, excellent performance, automatic HTTPS)
- **Netlify** (Free tier, branch previews, form handling)
- **Cloudflare Pages** (Free, global CDN, edge functions)

Custom Domain: $10-15/year (Namecheap, Google Domains)

- Benefits: Professional appearance, memorable URL, full control

CI/CD Pipeline:

.github/workflows/deploy.yml

Trigger: Push to main
↓

1. Fetch GitHub Activity Data (Python script)
2. Build Hugo Site
3. Run Lighthouse CI (performance checks)
4. Deploy to GitHub Pages
5. Purge CDN cache (if using Cloudflare)

Core Features & Requirements

1. Landing Page / Hero Section

Must-Have Elements:

- **Professional Headshot** (high-quality, professional attire)
- **Name & Current Title:** "Data Engineer | MS Business Analytics @ CSUEB"
- **Tagline:** "Building production-grade data pipelines with PySpark, Airflow, and Snowflake"
- **CTA Buttons:**
    - "View Projects" (scroll to projects section)
    - "Download Resume" (PDF, ATS-friendly format)
    - "Contact Me" (email link or form)
- **Tech Stack Badge Cloud:** Visual icons for Python, PySpark, Airflow, Kafka, Snowflake, AWS, Docker

Design Approach:

- Clean, minimal design (avoid clutter)
- Strong visual hierarchy (name/title prominent)
- Dark mode toggle (shows attention to UX detail)
- Subtle animations (fade-in on load, hover effects)

2. About Me Section

Content:

- **Professional Summary:** 2-3 sentences highlighting 4+ years experience, key technical strengths, career focus
    - Example: "Data Engineer with 4+ years building scalable ETL pipelines and lakehouse architectures. Expertise in real-time streaming (Kafka + Spark), cloud data platforms (AWS, Snowflake), and workflow orchestration (Airflow). Currently pursuing MS in Business Analytics at CSUEB while seeking challenging data engineering roles."
- **Experience Highlights:**
    - **Bhramastra (Nov 2020 - Jun 2024):** Built Medallion Lakehouse reducing query time 30%, orchestrated Airflow pipelines improving reliability 40%
    - **Tata Elxsi (Dec 2019 - Sep 2020):** Optimized AI data pipelines for LiDAR datasets (50K+ samples), reduced latency 15%
- **Current Focus:**
    - Graduate studies in Business Analytics (graduating May 2026)
    - President of AI Club at CSUEB (leading 15+ student community)
    - Actively seeking Data Engineer positions in Bay Area
- **Certifications:** AWS Certified Cloud Practitioner (2024), SnowPro Associate (2025)

Interactive Element:

- **Skills Matrix:** Interactive grid showing proficiency levels
    - Data Engineering Tools: 5/5 (PySpark, Airflow, Kafka, Delta Lake)
    - Cloud Platforms: 4/5 (AWS, Snowflake)
    - Programming: 5/5 (Python, SQL)

    - Visualization: 4/5 (Tableau, Streamlit)

3. Projects Showcase (Core Focus)

Project Cards - Detailed Breakdown:

Project 1: ThreatIntel ETL Pipeline

Hero Image: Architecture diagram showing data flow (URLhaus → Kafka → PySpark → Delta → ML Model)

Quick Stats:

- 50K+ daily threat indicators processed
- 99%+ data quality score
- 20K+ domains enriched daily
- 95%+ enrichment coverage

Tech Stack Badges: PySpark | Apache Airflow | Kafka | Delta Lake | AWS

Description:
Built production-grade ETL pipeline consolidating 3+ threat intelligence feeds (URLhaus, OpenPhish, MISP) using medallion architecture. Implemented real-time enrichment with parallel DNS resolution, IP geolocation, and WHOIS lookups, reducing latency from hours to minutes.

Key Achievements:

- Designed Airflow DAGs for scheduled ingestion with data validation checks
- Implemented Kafka-based streaming layer for real-time threat updates
- Built Delta Lake bronze/silver/gold layers with schema enforcement
- Created data quality framework ensuring 99%+ accuracy for ML models

Technical Deep Dive (Expandable Section):

- **Architecture Diagram:** Visual showing pipeline flow
- **Code Snippets:**
    - Airflow DAG structure with task dependencies
    - PySpark transformation logic for data enrichment
    - Delta Lake merge operations for upserts
- **Challenges & Solutions:**
    - Challenge: Handling rate limits on external APIs
    - Solution: Implemented exponential backoff with retry logic and batch request pooling

Links:

- [View GitHub Repo] (primary CTA)
- [Read Technical Blog Post] (detailed write-up)

Project 2: Cloud Capacity Forecasting

Hero Image: Time series forecast chart showing predicted vs actual capacity

Quick Stats:

- 15% improvement over baseline Holt-Winters
- 100% data integrity achieved
- Automated R-based pipeline
- Hybrid Regression + ARIMA model

Tech Stack Badges: R | readr | forecast | ggplot2

Description:
Developed automated forecasting pipeline for cloud resource optimization using hybrid Two-Level model (Regression + ARIMA). Eliminated timestamp parsing errors through type-strict data validation, enabling reliable capacity planning for production workloads.

Key Achievements:

- Built type-safe data ingestion pipeline with readr
- Implemented hybrid forecasting model outperforming standard approaches
- Created automated reporting with ggplot2 visualizations
- Deployed as scheduled R script for weekly capacity predictions

Technical Deep Dive (Expandable Section):

- **Model Architecture:** Diagram showing two-level approach
- **Code Snippets:**
  - Data validation logic with readr
  - Regression + ARIMA model implementation
  - Forecast evaluation metrics
- **Results:** Before/after comparison charts

Links:

- [View GitHub Repo]
- [Interactive Dashboard] (if deployed with Shiny or Streamlit)

Project 3: Portfolio Website (Meta Project)

Hero Image: Screenshot of live pipeline monitoring dashboard

Quick Stats:

- 100% static, no backend required
- Real-time data from GitHub API
- <2 second load time
- 95+ PageSpeed score

Tech Stack Badges: Hugo | JavaScript | GitHub Actions | GitHub Pages

Description:
Built this portfolio website as a demonstration of data engineering principles applied to frontend development. Features automated data pipeline fetching GitHub activity,

processing with pandas-like logic, and rendering interactive charts—all within a static site architecture.

Key Achievements:

- Implemented CI/CD pipeline with GitHub Actions
- Created scheduled data ingestion workflow (GitHub API)
- Built JSON-based "data lake" for portfolio analytics
- Deployed with zero-cost hosting infrastructure

Technical Deep Dive (Expandable Section):

- **Architecture Diagram:** CI/CD flow and data processing
- **Code Snippets:**
  - GitHub Actions workflow YAML
  - JavaScript data fetching and chart rendering
  - Hugo templating for dynamic content
- **Performance Optimization:** Bundle splitting, lazy loading, CDN usage

Links:

- [View Source Code]
- [Read Build Log] (blog post documenting the process)

4. Live Data Dashboard (The Differentiator)

Dashboard Title: "Portfolio Activity & Metrics"

Metrics Displayed:

Section 1: GitHub Activity (Last 30 Days)

- **Commit Frequency Chart:** Bar chart showing commits per day
- **Language Distribution:** Pie chart showing Python (60%), SQL (25%), Other (15%)
- **Active Repositories:** List of top 3 repos by recent activity
- **Contribution Streak:** Current streak count with visual indicator

Section 2: Learning & Growth

- **Skills Acquisition Timeline:** Horizontal timeline showing certifications, courses completed
- **Recent Achievements:**
  - ✅ SnowPro Associate Certification (Jan 2025)
  - ✅ AWS Cloud Practitioner (2024)
  - ▯ Databricks Spark Certification (In Progress)

Section 3: Portfolio Analytics (If Implemented)

- **Visitor Count:** Total unique visitors this month
- **Popular Projects:** Ranking of most-viewed project pages
- **Geographic Distribution:** Map showing visitor locations

Implementation:
```
// Fetch GitHub activity JSON (generated by CI/CD pipeline)
fetch('/data/github-activity.json')
```

```
.then(response => response.json())
.then(data => {
renderCommitChart(data.commits);
renderLanguagePie(data.languages);
updateStreak(data.streak);
});
```

Technical Note Display:

> "This dashboard is powered by a scheduled GitHub Actions pipeline that fetches data from the GitHub API daily, processes it using Python, and generates static JSON files. The frontend renders this data using Chart.js—demonstrating ETL principles in a static site architecture."

5. Technical Skills Grid

Categories:

Data Engineering (Primary Focus):

- **Proficiency Bars with Icons:**
  - PySpark ██████████ Expert (4+ years)
  - Apache Airflow █████████░ Advanced (3+ years)
  - Apache Kafka ████████░░ Advanced (2+ years)
  - Delta Lake ████████░░ Advanced (2+ years)
  - Spark Streaming ███████░░░ Intermediate-Advanced

Cloud & Data Platforms:

- AWS (S3, Glue, Lambda, Redshift, EC2) █████████░ Advanced
- Snowflake ████████░░ Advanced (SnowPro Certified)
- Databricks ███████░░░ Intermediate-Advanced
- PostgreSQL / MySQL █████████░ Advanced

Programming & Scripting:

- Python ██████████ Expert
- SQL ██████████ Expert
- Bash ████████░░ Advanced
- R ███████░░░ Intermediate

Machine Learning & AI:

- TensorFlow / Keras ███████░░░ Intermediate
- Scikit-Learn ████████░░ Advanced
- Computer Vision ██████░░░░ Intermediate

DevOps & Tools:

- Docker ████████░░ Advanced
- Git / GitHub ██████████ Expert
- CI/CD Concepts ███████░░░ Intermediate-Advanced

Interactive Feature:

- Click skill name → Popup showing relevant project examples using that skill
- Filter projects by selected skills

6. Experience Timeline

Visual Timeline (Horizontal or Vertical):

2024 - Present: Graduate Studies & Leadership

- MS Business Analytics @ CSUEB (Graduating May 2026)
- President, Artificial Intelligence Club (15+ member community)
- Coursework: Data Warehousing, Big Data, ML, Database Management

2020 - 2024: Data Engineer @ Bhramastra (Chennai, India)

- Built Medallion Lakehouse (30% faster queries, 60% faster processing)
- Orchestrated Airflow pipelines (40% better recovery time)
- Developed Kafka streaming ingestion (80% fewer duplicates)

2019 - 2020: Associate AI Engineer @ Tata Elxsi (Bangalore, India)

- LiDAR data pipelines (50K+ samples, 22% fewer anomalies)
- Optimized object detection inference (15% latency reduction)
- Automated data quality monitoring (20% more edge cases detected)

2015 - 2019: Bachelor of Engineering @ Anna University (India)

- Computer Science & Engineering
- Foundation in algorithms, databases, software engineering

Interactive Elements:

- Hover over timeline points → Show detailed accomplishments
- Click timeline period → Expand to full description with metrics

7. Blog / Articles Section

Purpose: Demonstrate technical communication skills, thought leadership, SEO benefits

Article Ideas (3-5 Initial Posts):

1. **"Building a Production-Grade Threat Intelligence Pipeline with PySpark and Airflow"**
   - Deep dive into ThreatIntel project architecture
   - Code examples, architectural diagrams, lessons learned
   - Target audience: Mid-level data engineers
2. **"Medallion Architecture Explained: From Bronze to Gold with Delta Lake"**
   - Tutorial-style post with code snippets
   - Explains data quality tiers, schema evolution, performance optimization
   - Target audience: Junior-to-mid data engineers
3. **"How I Built a Self-Updating Portfolio Website Using GitHub Actions"**
   - Meta-article about this portfolio project
   - GitHub Actions workflow breakdown, static site data engineering
   - Target audience: Data engineers interested in portfolio building

4. **"AWS vs Snowflake vs Databricks: A Practical Comparison for Data Engineers"**
    - Hands-on comparison based on real project experience
    - Use cases, pricing considerations, tooling ecosystem
    - Target audience: Hiring managers, engineers evaluating platforms
5. **"Interview Prep: 10 PySpark Optimization Techniques I Wish I Knew Earlier"**
    - Practical optimization tips with before/after benchmarks
    - Broadcast joins, partition tuning, caching strategies
    - Target audience: Data engineers preparing for interviews

Blog Implementation:

- Hugo markdown files (easy to write and maintain)
- Syntax highlighting for code blocks
- Estimated reading time
- Social sharing buttons (LinkedIn, Twitter)
- Comment system (optional: Disqus or Utterances)

8. Resume / CV Section

Content:

- **Embedded PDF Viewer:** Display ATS-friendly resume inline
- **Download Button:** Prominent CTA for downloading PDF
- **Multiple Formats:**
    - Standard Resume (1-page, ATS-optimized)
    - Extended CV (2-page with publications/talks if applicable)

Resume Hosting Strategy:

- Store PDF in `/static/resume/` folder
- Version control resume updates in Git
- Use semantic versioning (Mohanasundaram_Resume_v2.1.pdf)

ATS Optimization Checklist:

- Plain text formatting (no fancy graphics)
- Standard section headers
- Keyword-rich descriptions (match job postings)
- Consistent date formatting
- PDF format (not Word doc)

9. Contact Section

Contact Methods:

Primary:

- **Email:** mohanasundaram012@gmail.com (clickable mailto link)
- **LinkedIn:** linkedin.com/in/Mohanasundaram (profile link)
- **GitHub:** github.com/mohanasundaramm1 (profile link)

Optional Contact Form:

- Name, Email, Message fields

- Powered by: Formspree (free tier, 50 submissions/month) or Netlify Forms
- Anti-spam: hCaptcha or reCAPTCHA
- Success message: "Thanks! I'll get back to you within 24 hours."

Additional Links:

- **Location:** San Francisco Bay Area (available for local/hybrid roles)
- **Status:** Open to full-time Data Engineer opportunities
- **Availability:** Available for interviews, can start June 2026 (post-graduation)

Call-to-Action:

> "I'm actively seeking Data Engineer roles in the Bay Area. If you're looking for someone who can build production-grade pipelines, optimize data infrastructure, and communicate complex technical concepts clearly—let's connect!"

10. Footer

Content:

- **Copyright:** © 2026 Mohanasundaram Murugesan
- **Social Links:** GitHub, LinkedIn, email icons
- **Quick Links:** About | Projects | Blog | Resume | Contact
- **Built With:** "Built with Hugo, deployed on GitHub Pages. [View source code]"
- **Last Updated:** Dynamically generated from Git commit date

---

Information Architecture

Site Map

```
Home (/)
├── About (#about)
├── Projects (#projects)
│   ├── ThreatIntel Pipeline (/projects/threatintel)
│   ├── Cloud Forecasting (/projects/cloud-forecasting)
│   └── Portfolio Website (/projects/portfolio)
├── Live Dashboard (#dashboard)
├── Skills (#skills)
├── Experience (#experience)
├── Blog (/blog)
│   ├── Article 1 (/blog/threatintel-pipeline)
│   ├── Article 2 (/blog/medallion-architecture)
│   └── Article 3 (/blog/portfolio-github-actions)
├── Resume (/resume)
└── Contact (#contact)
```

URL Structure

- Clean, semantic URLs: `/projects/threatintel` not `/project?id=1`
- Lowercase with hyphens: `/blog/medallion-architecture`
- No file extensions: `/resume` not `/resume.html`

Design System & Visual Guidelines

Color Palette

Primary Colors (Modern Data Engineering Brand):

- **Primary:** #20809D (Teal Blue - represents data flow, technology)
- **Secondary:** #5E5240 (Warm Gray - professional, grounded)
- **Accent:** #32B8C6 (Bright Teal - CTAs, highlights)

Neutrals:

- **Background Light:** #FCFCF9 (Cream White)
- **Background Dark:** #1F2121 (Charcoal - for dark mode)
- **Text Primary:** #134252 (Dark Slate)
- **Text Secondary:** #626C71 (Medium Gray)

Semantic Colors:

- **Success:** #21808D (Teal)
- **Warning:** #E68161 (Orange)
- **Error:** #C0152F (Red)

Dark Mode:

- Use CSS custom properties for theme switching
- Store preference in localStorage
- Toggle button in header

Typography

Font Families:

- **Headings:** 'Inter' or 'DM Sans' (clean, modern sans-serif)
- **Body:** 'Inter' or system font stack (-apple-system, BlinkMacSystemFont)
- **Code:** 'Fira Code' or 'JetBrains Mono' (monospace with ligatures)

Type Scale:

- **H1 (Hero):** 48px / 3rem (bold)
- **H2 (Section):** 36px / 2.25rem (semibold)
- **H3 (Subsection):** 28px / 1.75rem (semibold)
- **H4 (Card Title):** 20px / 1.25rem (medium)
- **Body:** 16px / 1rem (regular)
- **Small:** 14px / 0.875rem (regular)

Line Height:

- Headings: 1.2
- Body: 1.6 (improves readability)

Spacing System

- **Base Unit:** 8px
- **Scale:** 8px, 16px, 24px, 32px, 48px, 64px, 96px
- Consistent spacing creates visual rhythm

Components

Buttons:

- **Primary:** Filled with accent color, white text, hover darkens
- **Secondary:** Outlined with primary color, hover fills
- **Size:** 44px min height (touch-friendly)
- **Border Radius:** 8px (modern, friendly)

Cards (Project Cards):

- **Background:** White (light mode) / Dark Gray (dark mode)
- **Border:** 1px solid with subtle shadow
- **Padding:** 24px
- **Hover Effect:** Lift (translateY: -4px) + stronger shadow

Navigation:

- **Fixed Header:** Sticky navigation on scroll
- **Mobile:** Hamburger menu with smooth slide-in
- **Active State:** Underline or accent color indicator

Responsive Breakpoints

- **Mobile:** < 640px (single column layout)
- **Tablet:** 641px - 1024px (adaptive layout)
- **Desktop:** > 1024px (full multi-column layout)

Animations

- **Principle:** Subtle, purposeful, not distracting
- **Scroll Animations:** Fade-in on viewport entry (Intersection Observer)
- **Hover Effects:** 150-250ms transitions
- **Page Transitions:** Smooth, minimal

Content Strategy

Key Messages

Primary Message:
"Production-Ready Data Engineer with 4+ Years Building Scalable Pipelines"

Supporting Messages:

- "Expert in modern data stack: PySpark, Airflow, Kafka, Snowflake"
- "Proven track record of measurable impact (30-80% improvements)"
- "Strong communicator bridging technical and business stakeholders"

Content Tone & Voice

Tone:

- **Professional** but approachable
- **Technical** but clear (avoid unnecessary jargon)
- **Confident** without arrogance
- **Results-focused** with specific metrics

Voice Characteristics:

- Active voice preferred ("Built ETL pipeline" vs "ETL pipeline was built")
- Quantify achievements ("reduced latency by 20%" vs "improved performance")
- Balance technical depth with accessibility (explain acronyms on first use)

SEO Strategy

Target Keywords:

- Primary: "data engineer portfolio", "Mohanasundaram Murugesan"
- Secondary: "PySpark developer", "Airflow engineer", "Snowflake data engineer"
- Long-tail: "data engineer San Francisco", "real-time streaming engineer"

On-Page SEO:

- **Title Tags:** "Mohanasundaram Murugesan - Data Engineer | PySpark & Airflow Expert"
- **Meta Descriptions:** Compelling 150-160 character summaries
- **Header Hierarchy:** Proper H1 → H2 → H3 structure
- **Alt Text:** Descriptive alt text for all images
- **Schema Markup:** Person schema for rich snippets

Technical SEO:

- **Sitemap.xml:** Auto-generated by Hugo
- **Robots.txt:** Allow all, sitemap reference
- **Canonical URLs:** Prevent duplicate content
- **Open Graph Tags:** LinkedIn/Twitter card previews
- **Performance:** 95+ PageSpeed score (helps rankings)

Technical Implementation Plan

Phase 1: Foundation (Week 1)

Tasks:

1. **Domain & Hosting Setup**
   - Purchase domain (mohanasundaram.dev or .tech)
   - Set up GitHub Pages repository

   - Configure custom domain DNS (A records + CNAME)
   - Enable HTTPS (automatic via GitHub Pages)
2. **Hugo Setup**
   - Install Hugo (extended version for SCSS support)
   - Choose and install theme (PaperMod or Stack)
   - Configure `config.toml` (site title, baseURL, params)
   - Set up project structure (`content/`, `static/`, `layouts/`)
3. **Basic Content**
   - Write About Me section
   - Create placeholder project pages
   - Add resume PDF
   - Set up basic navigation

Deliverable: Live site with basic structure at custom domain

Phase 2: Content & Projects (Week 2)

Tasks:

1. **Project Deep Dives**
   - Write detailed ThreatIntel project page (1500+ words)
   - Write Cloud Forecasting project page (1000+ words)
   - Create architecture diagrams (draw.io or Excalidraw)
   - Add code snippets with syntax highlighting
2. **Experience & Skills**
   - Build interactive skills grid
   - Create experience timeline
   - Add certifications section
   - Write compelling professional summary
3. **Design Refinement**
   - Customize theme colors to brand palette
   - Add custom CSS for cards, buttons, layouts
   - Implement dark mode toggle
   - Add favicon and logo

Deliverable: Complete static content ready for review

Phase 3: Data Engineering Features (Week 3)

Tasks:

1. **GitHub Activity Pipeline**
   - Write Python script to fetch GitHub API data

     ## scripts/fetch_github_data.py

     ```
     import requests
     import json
     from datetime import datetime, timedelta
     def fetch_commits(username, days=30):
     # Fetch commit activity
     # Process and aggregate
     ```

```
# Generate JSON output
pass
def fetch_language_stats(username):
# Fetch repo language data
# Calculate percentages
pass
```

- Create GitHub Actions workflow

.github/workflows/update-data.yml

```
name: Update Portfolio Data
on:
schedule:
- cron: '0 6 * * *' # Daily at 6 AM UTC
workflow_dispatch:
jobs:
update-data:
runs-on: ubuntu-latest
steps:
- uses: actions/checkout@v3
- name: Set up Python
uses: actions/setup-python@v4
with:
python-version: '3.11'
- name: Fetch GitHub data
run: python scripts/fetch_github_data.py
env:
GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
- name: Commit data
run: |
git config user.name "GitHub Actions"
git config user.email "actions@github.com"
git add data/
git commit -m "Update GitHub activity data"
git push
```

- Store generated JSON in `/static/data/` folder
- Test pipeline locally first

2. **Dashboard Implementation**
   - Create dashboard section in homepage
   - Write JavaScript to fetch and render data

```
// assets/js/dashboard.js
async function loadGitHubActivity() {
const response = await fetch('/data/github-activity.json');
const data = await response.json();
renderCommitChart(data.commits);
renderLanguageChart(data.languages);
updateMetrics(data.stats);
}
function renderCommitChart(commits) {
// Chart.js implementation
}
```

    - Add Chart.js library (via CDN or npm)
    - Style dashboard cards and charts
    - Add loading states and error handling
3. **Performance Optimization**
    - Minify CSS and JavaScript
    - Optimize images (WebP format, lazy loading)
    - Enable Gzip compression
    - Set up CDN (Cloudflare free tier)

Deliverable: Fully functional live dashboard demonstrating data engineering skills

Phase 4: Blog & Polish (Week 4)

Tasks:

1. **Blog Setup**
    - Configure Hugo blog section
    - Create blog post template
    - Write first 2-3 articles (ThreatIntel, Medallion Architecture)
    - Add syntax highlighting theme
    - Implement reading time estimates
2. **SEO & Analytics**
    - Add Google Analytics or Plausible (privacy-friendly)
    - Generate sitemap.xml
    - Create robots.txt
    - Add Open Graph meta tags
    - Implement schema markup
3. **Final Polish**
    - Cross-browser testing (Chrome, Firefox, Safari, Edge)
    - Mobile responsiveness testing (iOS, Android)
    - Accessibility audit (WAVE tool, Lighthouse)
    - Performance testing (PageSpeed Insights, WebPageTest)
    - Copy editing and proofreading
4. **Launch Prep**
    - Create LinkedIn announcement post
    - Update LinkedIn profile to link to portfolio
    - Share in relevant communities (r/dataengineering, Twitter)
    - Email network about new portfolio

Deliverable: Production-ready portfolio website with marketing launch plan

---

Risk Management

Technical Risks

Risk 1: GitHub API Rate Limits

- **Likelihood:** Medium
- **Impact:** Medium (dashboard won't update)
- **Mitigation:** Use authenticated API calls (higher limits), implement caching, add fallback static data

Risk 2: Build Pipeline Failures

- **Likelihood:** Low
- **Impact:** High (site won't deploy)
- **Mitigation:** Extensive testing, error notifications, manual deployment fallback

Risk 3: Performance Issues

- **Likelihood:** Low
- **Impact:** Medium (slow load times hurt UX and SEO)
- **Mitigation:** Image optimization, code minification, CDN usage, regular performance audits

Content Risks

Risk 4: Overly Technical Content

- **Likelihood:** Medium
- **Impact:** Medium (recruiters may not understand value)
- **Mitigation:** Get feedback from non-technical friends, include executive summaries, use clear metrics

Risk 5: Outdated Information

- **Likelihood:** Medium (over time)
- **Impact:** Low-Medium (appears inactive)
- **Mitigation:** Set quarterly review reminders, update blog regularly, add "Last Updated" timestamps

Business Risks

Risk 6: Low Visibility

- **Likelihood:** High (competitive market)
- **Impact:** High (defeats purpose)
- **Mitigation:** Active LinkedIn promotion, SEO optimization, share in relevant communities, add to resume

---

Success Criteria & KPIs

Immediate Success (First Month)

- [x] **Site is live** with custom domain and HTTPS
- [x] **PageSpeed score 95+** on mobile and desktop
- [x] **All core sections complete:** About, Projects, Skills, Experience, Resume, Contact
- [x] **Live dashboard functional** with real GitHub data
- [x] **2+ blog posts published** with technical depth
- [x] **Shared on LinkedIn** with 50+ views/reactions

Short-Term Success (Months 2-3)

- [ ] **100+ unique visitors per month** (Google Analytics)
- [ ] **10+ recruiter profile views per week** (LinkedIn)
- [ ] **3+ project deep-dive sessions** (Analytics: >3 min time on project pages)
- [ ] **2+ interview callbacks** referencing portfolio
- [ ] **5+ blog posts published** covering key data engineering topics

Long-Term Success (Months 4-6)

- [ ] **Landed Data Engineer position** where portfolio was factor in hiring decision
- [ ] **Featured in community** (shared on r/dataengineering, LinkedIn influencer repost)
- [ ] **Blog ranking** on Google for "data engineering portfolio" or related terms
- [ ] **Portfolio template** used by peers (if open-sourced)

Maintenance Plan

Regular Updates (Monthly)

- **Content Review:** Check for outdated information, broken links
- **Blog Cadence:** Publish 1-2 new articles per month
- **Resume Updates:** Keep resume current with new skills, projects
- **Dashboard Monitoring:** Ensure data pipeline still running, fix breaks

Quarterly Reviews

- **Analytics Review:** Check visitor patterns, popular content, bounce rates
- **SEO Audit:** Review keyword rankings, update meta descriptions
- **Design Refresh:** Minor visual updates to keep modern appearance
- **Performance Audit:** Run PageSpeed, fix any regressions

Annual Overhaul (Optional)

- **Major Design Update:** Consider new theme or visual refresh
- **Architecture Review:** Evaluate if static site still meets needs
- **Content Audit:** Archive old posts, rewrite evergreen content
- **Feature Additions:** Add new interactive elements, tools

Budget Breakdown

One-Time Costs

Item	Provider	Cost
Domain Name (1 year)	Namecheap / Google Domains	$12-15
Professional Headshot	Local photographer (optional)	$0-100
Total One-Time		**$12-115**

Recurring Costs (Annual)

Item	Provider	Cost
Domain Renewal	Namecheap / Google Domains	$12-15/year
Hosting	GitHub Pages	**FREE**
CDN	Cloudflare (free tier)	**FREE**
SSL Certificate	GitHub Pages / Let's Encrypt	**FREE**
Analytics	Plausible / Google Analytics	**FREE**
Contact Form	Formspree (free tier)	**FREE**
Total Annual		**$12-15/year**

Total Cost (First Year): $24-130
Total Cost (Subsequent Years): $12-15/year

Cost Optimization Notes

- **GitHub Pages** provides free hosting for public repositories (including custom domains)
- **Cloudflare** free tier includes global CDN and DDoS protection
- **Let's Encrypt** SSL certificates auto-renew via GitHub Pages
- **Hugo** is open-source and free
- **No backend costs** since everything is static
- **Scalability:** Can handle 100K+ monthly visitors on free tier

---

Open Questions & Decisions Needed

Decision Points

1. Domain Name Selection

- **Option A:** mohanasundaram.dev (modern, developer-focused)
- **Option B:** mohanasundaram.tech (broader tech appeal)
- **Option C:** Use GitHub username: mohanasundaramm1.com
- **Recommendation:** mohanasundaram.dev (shorter, memorable)

2. Blog Depth

- **Option A:** Technical deep-dives (1500+ words, code-heavy)
- **Option B:** Shorter insights (500-800 words, accessible)
- **Option C:** Mix of both (some deep, some quick tips)
- **Recommendation:** Option C for broader audience appeal

3. Dashboard Scope

- **Option A:** GitHub activity only (simple, reliable)
- **Option B:** Add portfolio analytics (visitor stats, popular pages)
- **Option C:** Add external data source (e.g., Kaggle profile, LeetCode stats)
- **Recommendation:** Start with A, expand to B/C later

4. Project Repositories

- **Option A:** Keep projects private, show code snippets only
- **Option B:** Make all projects public to demonstrate transparency
- **Option C:** Create sanitized versions of work projects for public sharing
- **Recommendation:** Option C for ThreatIntel (already public?), B for new projects

Follow-Up Items

- [ ] **Finalize domain name** decision and purchase
- [ ] **Get professional headshot** or use high-quality existing photo
- [ ] **Review Hugo themes** and select final candidate
- [ ] **Outline first 3 blog posts** with titles and key points
- [ ] **Decide on GitHub repo visibility** for existing projects
- [ ] **Set up development environment** (Hugo installation, GitHub repo)

Appendix A: Competitive Analysis

Competitor Portfolio Analysis

Portfolio 1: [Example Data Engineer Portfolio on GitHub]

- **Strengths:** Clean design, multiple projects, active blog
- **Weaknesses:** Static content only, no live data, generic projects
- **Differentiation:** Our live dashboard and production metrics set us apart

Portfolio 2: [Industry Standard Portfolio]

- **Strengths:** Professional headshots, well-written content, good SEO
- **Weaknesses:** Outdated design, no interactivity, slow load times
- **Differentiation:** Modern design, fast performance, interactive elements

Key Takeaways:

- Most portfolios are static Markdown/HTML
- Few demonstrate real data engineering in portfolio itself
- Production metrics and quantified impact are rare
- Mobile-responsiveness often overlooked

Appendix B: Technical Resources

Development Tools

- **Hugo Installation:** https://gohugo.io/installation/
- **Hugo Themes:** https://themes.gohugo.io/
- **Chart.js Docs:** https://www.chartjs.org/docs/latest/
- **GitHub Actions:** https://docs.github.com/en/actions

Design Resources

- **Color Palette Tools:** Coolors.co, Adobe Color
- **Icon Libraries:** Lucide Icons, Font Awesome
- **Stock Photos:** Unsplash, Pexels
- **Diagram Tools:** Excalidraw, draw.io

Learning Resources

- **Hugo Tutorial:** https://gohugo.io/getting-started/quick-start/
- **Static Site SEO:** Moz Beginner's Guide
- **GitHub Actions Workflows:** Awesome GitHub Actions repo
- **Web Performance:** web.dev/measure

Appendix C: Example Code Snippets

GitHub Actions Workflow (Complete)

```
name: Deploy Portfolio with Data Update

on:
push:
branches: [main]
schedule:
- cron: '0 6 * * *' # Daily at 6 AM UTC
workflow_dispatch:

jobs:
build-deploy:
runs-on: ubuntu-latest
```

```
steps:
  - name: Checkout code
    uses: actions/checkout@v3
    with:
      submodules: true
      fetch-depth: 0

  - name: Set up Python
    uses: actions/setup-python@v4
    with:
      python-version: '3.11'

  - name: Install Python dependencies
    run: |
      pip install requests pandas

  - name: Fetch GitHub activity data
    run: python scripts/fetch_github_data.py
    env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      GITHUB_USERNAME: mohanasundaramm1

  - name: Set up Hugo
    uses: peaceiris/actions-hugo@v2
    with:
      hugo-version: 'latest'
      extended: true

  - name: Build Hugo site
    run: hugo --minify

  - name: Deploy to GitHub Pages
    uses: peaceiris/actions-gh-pages@v3
    with:
      github_token: ${{ secrets.GITHUB_TOKEN }}
      publish_dir: ./public
      cname: mohanasundaram.dev
```

Python Data Fetching Script (fetch_github_data.py)

```
import os
import requests
import json
from datetime import datetime, timedelta
from collections import defaultdict

GITHUB_TOKEN = os.environ.get('GITHUB_TOKEN')
GITHUB_USERNAME = os.environ.get('GITHUB_USERNAME')
HEADERS = {'Authorization': f'token {GITHUB_TOKEN}'}

def fetch_commit_activity(username, days=30):
"""Fetch commit activity for the last N days"""
```

url = f'https://api.github.com/users/{username}/events'
response = requests.get(url, headers=HEADERS)
events = response.json()

```
commit_counts = defaultdict(int)
cutoff_date = datetime.now() - timedelta(days=days)

for event in events:
    if event['type'] == 'PushEvent':
        event_date = datetime.strptime(event['created_at'], '%Y-%m-%dT%H:%M:%S
        if event_date >= cutoff_date:
            date_str = event_date.strftime('%Y-%m-%d')
            commit_counts[date_str] += len(event['payload']['commits'])

return dict(commit_counts)
```

def fetch_language_stats(username):
"""Aggregate language usage across all repos"""
url = f'https://api.github.com/users/{username}/repos'
response = requests.get(url, headers=HEADERS)
repos = response.json()

```
language_bytes = defaultdict(int)

for repo in repos:
    if not repo['fork']:  # Skip forked repos
        lang_url = repo['languages_url']
        lang_response = requests.get(lang_url, headers=HEADERS)
        languages = lang_response.json()

        for lang, bytes_count in languages.items():
            language_bytes[lang] += bytes_count

# Convert to percentages
total_bytes = sum(language_bytes.values())
language_percentages = {
    lang: round((bytes_count / total_bytes) * 100, 1)
    for lang, bytes_count in language_bytes.items()
}
```

```
# Sort by percentage descending
return dict(sorted(language_percentages.items(), key=lambda x: x[1], reverse=T
```

def calculate_streak(commit_counts):
"""Calculate current commit streak"""
today = datetime.now().date()
streak = 0

```
for i in range(365):  # Check last year
    check_date = today - timedelta(days=i)
    date_str = check_date.strftime('%Y-%m-%d')

    if date_str in commit_counts and commit_counts[date_str] > 0:
        streak += 1
    else:
        break  # Streak broken

return streak
```

def main():
print("Fetching GitHub activity data...")

```
commit_counts = fetch_commit_activity(GITHUB_USERNAME, days=30)
language_stats = fetch_language_stats(GITHUB_USERNAME)
streak = calculate_streak(commit_counts)

# Get top 3 active repos
url = f'https://api.github.com/users/{GITHUB_USERNAME}/repos'
response = requests.get(url, headers=HEADERS, params={'sort': 'updated', 'per_p
top_repos = [{'name': repo['name'], 'url': repo['html_url'], 'description': repo['des
             for repo in response.json() if not repo['fork']]

output_data = {
    'last_updated': datetime.now().isoformat(),
    'commits': commit_counts,
    'languages': language_stats,
    'streak': streak,
    'top_repos': top_repos
}

# Write to JSON file
output_path = 'static/data/github-activity.json'
os.makedirs(os.path.dirname(output_path), exist_ok=True)

with open(output_path, 'w') as f:
    json.dump(output_data, f, indent=2)

print(f"✅ Data written to {output_path}")
print(f"   - Total commits (30 days): {sum(commit_counts.values())}")
print(f"   - Current streak: {streak} days")
print(f"   - Top language: {list(language_stats.keys())[0]}")
```

if **name** == '**main**':
main()

JavaScript Dashboard Rendering (dashboard.js)

// Load and render GitHub activity dashboard
async function initDashboard() {
try {
const response = await fetch('/data/github-activity.json');
const data = await response.json();

```
renderCommitChart(data.commits);
renderLanguageChart(data.languages);
updateMetrics(data.streak, data.commits);
displayTopRepos(data.top_repos);
updateTimestamp(data.last_updated);
```

} catch (error) {
console.error('Error loading dashboard data:', error);
document.getElementById('dashboard').innerHTML = '
Dashboard temporarily unavailable

';
}
}

function renderCommitChart(commits) {
const ctx = document.getElementById('commitChart').getContext('2d');

```
// Sort dates chronologically
const sortedDates = Object.keys(commits).sort();
const commitCounts = sortedDates.map(date => commits[date]);

new Chart(ctx, {
type: 'bar',
data: {
labels: sortedDates,
datasets: [{
label: 'Commits',
data: commitCounts,
backgroundColor: '#32B8C6',
borderColor: '#20809D',
borderWidth: 1
}]
},
options: {
responsive: true,
plugins: {
title: {
display: true,
text: 'Commit Activity (Last 30 Days)'
}
},
scales: {
y: {
beginAtZero: true,
ticks: { stepSize: 1 }
}
}
}
});
}

function renderLanguageChart(languages) {
const ctx = document.getElementById('languageChart').getContext('2d');

// Get top 5 languages
const topLanguages = Object.entries(languages).slice(0, 5);
const labels = topLanguages.map(([lang]) => lang);
const percentages = topLanguages.map(([, pct]) => pct);

new Chart(ctx, {
type: 'doughnut',
data: {
labels: labels,
datasets: [{
data: percentages,
backgroundColor: ['#20809D', '#32B8C6', '#5E5240', '#E68161', '#626C71']
}]
},
options: {
```

```
responsive: true,
plugins: {
title: {
display: true,
text: 'Language Distribution'
},
legend: {
position: 'bottom'
}
}
}
});
}

function updateMetrics(streak, commits) {
const totalCommits = Object.values(commits).reduce((a, b) => a + b, 0);

document.getElementById('streak-count').textContent = streak;
document.getElementById('commit-count').textContent = totalCommits;
}

function displayTopRepos(repos) {
const container = document.getElementById('top-repos');
container.innerHTML = repos.map(repo => <div class="repo-card"> <h4><a href="${repo.url}" target="_blank">${repo.name}</a></h4> <p>${repo.description || 'No description'}</p> </div> ).join('');
}

function updateTimestamp(lastUpdated) {
const date = new Date(lastUpdated);
const formatted = date.toLocaleDateString('en-US', {
month: 'short',
day: 'numeric',
year: 'numeric'
});
document.getElementById('last-updated').textContent = Last updated: ${formatted};
}

// Initialize dashboard on page load
document.addEventListener('DOMContentLoaded', initDashboard);
```

Document Approval

Prepared By: Mohanasundaram Murugesan
Date: February 2026
Version: 1.0

Approval Status: ☐ Draft ☑ Ready for Development ☐ In Progress ☐ Complete

Next Steps:

1. Review and approve PRD

2. Purchase domain name
3. Begin Phase 1 implementation (Foundation setup)
4. Schedule weekly progress reviews

End of Document