Technical Development Brief for Daniel

Sentia Spirits Manufacturing Planning MVP

Project Overview

You will build a Manufacturing Planning Dashboard to optimize Sentia Spirits' working capital management and inventory planning. This application will provide sophisticated demand forecasting, stock level optimization, and working capital modeling for our product portfolio.

Timeline: 3 weeks to first week of September 2025

Your Role: Execute development prompts using vibe coding methodology to build Sentia's custom manufacturing planning solution

---

Development Methodology: Vibe Coding

Core Principle

You will receive **comprehensive sequential development prompts** that build the entire application using scaffolding and Lego block methodology. Your job is to:

1. **Execute each prompt exactly as specified**
2. **Maintain strict context references**
3. **Debug and fix any errors**
4. **Request additional prompts for gaps**

Critical Success Factor: Prevent AI Drift

- **Always reference specific context folders/files** in every interaction with Claude
- **Never search the entire codebase** - use structured context references
- **Validate context understanding** before proceeding with each prompt

---

Technical Stack

Development Environment

- **Primary IDE**: Cursor

- **AI Assistant**: Claude Code CLI
- **Version Control**: GitHub CLI
- **Database**: Neon PostgreSQL (vector database)
- **Hosting**: Railway
- **Backend**: Flask (Python)
- **Frontend**: HTML/CSS/JavaScript with responsive design

Branch Structure

Plain Text

```
development  → All development work (your primary branch)
test         → User acceptance testing
production   → Live application
```

Auto-sync: All branches automatically sync to respective Neon databases and Railway environments

---

Context Management System

Folder Structure (Critical for AI Drift Prevention)

Plain Text

```
/context/
├── business-requirements/     # Sentia's business needs and data
├── technical-specifications/  # System architecture and specs
├── database-schemas/          # Database models and relationships
├── api-documentation/         # API endpoints and integration specs
├── ui-components/             # Frontend components and layouts
├── business-logic/            # Forecasting algorithms and calculations
├── testing-scenarios/         # Test cases and validation criteria
└── deployment-configs/        # Infrastructure and deployment settings
```

Context Usage Rules

1. **Always specify exact context folder/file** when asking Claude for help
2. **Example**: "Using /context/database-schemas/inventory_models.md, create the Product model"

3. **Never say**: "Look at the codebase and figure out..."
4. **Keep context folders manageable** - break large files into focused smaller files

---

Your Development Process

Phase 1: Environment Setup

1. Install and configure Cursor, Claude Code CLI, GitHub CLI
2. Create GitHub repository with three branches
3. Set up Neon PostgreSQL databases for each branch
4. Configure Railway hosting environments
5. Create context folder structure

Phase 2: Execute Development Prompts

1. **Receive sequential development prompts**
2. **Read context files specified in each prompt**
3. **Execute prompt using Claude Code CLI in Cursor**
4. **Test the generated code**
5. **Debug any errors**
6. **Commit to development branch**
7. **Request next prompt as needed**

Phase 3: Quality Assurance

- **Double-check all generated code**
- **Run tests after each major component**
- **Fix bugs immediately**
- **Update context documentation**
- **Validate against business requirements**

Your Responsibilities

Primary Tasks

1. **Execute development prompts sequentially** - Don't skip ahead or deviate
2. **Maintain context integrity** - Always use proper context references
3. **Debug and fix errors** - Generated code may need refinement
4. **Test functionality** - Ensure each component works before proceeding
5. **Document issues** - Report problems and request additional prompts

Code Quality Standards

- **Follow Flask best practices** for backend development
- **Ensure responsive design** for frontend
- **Implement proper error handling**
- **Add appropriate logging**
- **Write clean, readable code**
- **Comment complex business logic**

Communication Protocol

- **Report completion** of each prompt execution
- **Document any errors** encountered and how you fixed them
- **Request clarification** if prompt instructions are unclear
- **Ask for additional prompts** to fill gaps or improve functionality

---

Business Context (Essential Understanding)

Sentia Spirits Challenge

- **3 products**: GABA Red, Black, Gold (50cl bottles)
- **3 markets**: UK, EU, USA (= 9 effective SKUs due to labeling)
- **5 sales channels**: Amazon UK/USA, Shopify UK/USA/EU
- **Seasonal variation**: 15x volume increase (June → Dec/Jan)
- **Current problem**: £1.5M working capital requirement unsustainable

Application Requirements

1. **Demand Forecasting**: Predict sales by product/region/channel
2. **Stock Level Optimization**: Calculate required inventory levels

3. **Working Capital Modeling**: Track cash requirements for inventory
4. **Scenario Planning**: Compare conservative vs aggressive stocking
5. **Dashboard Visualization**: Real-time charts and reporting

---

Success Metrics

Technical Success

- [ ] Application runs without errors in all environments
- [ ] All business requirements implemented correctly
- [ ] Responsive design works on desktop and mobile
- [ ] Data imports and calculations are accurate
- [ ] User authentication and permissions work properly

Process Success

- [ ] All development prompts executed successfully
- [ ] Context management prevents AI drift
- [ ] Code quality meets professional standards
- [ ] Documentation is comprehensive and accurate
- [ ] Deployment pipeline works smoothly

Emergency Protocols

If You Encounter Issues

1. **Stop and document the problem** - Don't continue with broken code
2. **Check context references** - Ensure you're using correct context files
3. **Debug systematically** - Use Cursor's debugging tools
4. **Request help from development team** - Ask for additional prompts or clarification
5. **Update context documentation** - Document solutions for future reference

Red Flags (Stop Immediately)

- Claude starts "hallucinating" or making assumptions not in context

- Code generation becomes inconsistent with previous components
- Business logic doesn't match Sentia's actual requirements
- Database operations fail or produce incorrect results

Final Notes

Remember

- **You're building Sentia's critical business solution** - The operations team depends on this working correctly
- **Quality over speed** - Better to build correctly than quickly
- **Context is everything** - Proper context management is critical for success
- **Ask questions** - Better to clarify requirements than assume

Your Success = Sentia's Success

This Manufacturing Planning Dashboard will directly impact Sentia's ability to optimize working capital and manage inventory effectively. Your careful execution of the development prompts and attention to detail will ensure the operations team has the tools they need to make informed decisions about production planning, stock levels, and cash flow management.

Ready to begin? Confirm your understanding and we'll start with the first development prompt.