Full-Stack Development Workflow: Vue/Quasar Frontend & Flask/AI Backend

This document outlines a comprehensive workflow for full-stack development, integrating a Vue/Quasar frontend with a Flask backend and an AI agent using the Metta framework (metta-lang.dev). The workflow is visually represented in the accompanying mindmap diagram.

Mindmap Diagram

Mindmap Diagram: Full-Stack Development Workflow

Detailed Workflow Outline

1. Project Setup & Initialization

- **Overall Project:** Monorepo vs. Polyrepo, Version Control (Git)
- **Frontend Setup:** Vue CLI, Quasar CLI, npm/yarn, `quasar.conf.js`, `vue.config.js`
- **Backend Setup:** Python, pip/conda, virtual environments, Flask, `config.py`

2. Core Concepts & Components

- **Frontend (Vue/Quasar):** Components, Props, Emits, Slots, Composition API, Quasar Components, Reusability
- **Backend (Flask):** Routing, Views, Templates (Jinja2), Flask Extensions (SQLAlchemy, RESTful/RESTX)

3. State & Data Management

- **Frontend State:** Pinia (Vue 3), Vuex (Vue 2), Local State

- **Backend Data:** Databases (PostgreSQL, MySQL), ORM (SQLAlchemy), Migrations (Flask-Migrate), Caching (Redis)

4. API Communication & Integration

- **Frontend to Backend:** Axios, Fetch API, JSON payloads
- **Backend to AI Agent:** Metta Framework (metta-lang.dev), Agent Design, API Endpoints for AI
- **Authentication & Authorization:** JWT, OAuth, Flask-Login, Flask-JWT-Extended

5. UI/UX & Styling (Frontend Focus)

- **Quasar Design System:** Material Design, Custom Themes
- **Styling:** SCSS/Sass, CSS Variables, Utility Classes
- **Responsiveness:** Quasar Flexbox, Grid, Breakpoints
- **Icons:** Quasar Icon Packs

6. Best Practices & Development Workflow

- **Code Structure:** Modularization, Separation of Concerns
- **Naming Conventions:** Consistent naming
- **Linting & Formatting:** ESLint, Prettier
- **Testing:** Unit Testing (Frontend: Vue Test Utils, Vitest/Jest; Backend: pytest), Integration Testing, E2E Testing (Cypress)
- **Performance Optimization:** Lazy loading, Code splitting, Image optimization
- **Accessibility (A11y):** Semantic HTML, ARIA attributes
- **Internationalization (i18n):** Vue I18n, Quasar I18n

7. Security

- **Input Validation:** Preventing injection attacks
- **Data Encryption:** Protecting sensitive data
- **CORS:** Cross-Origin Resource Sharing
- **CSRF Protection:** Flask-WTF

- **Logging & Monitoring:** Tracking security events

8. Deployment & Operations

- **Frontend Build:** Quasar CLI Build (PWA, SPA, SSR, Electron, Capacitor)
- **Backend Deployment:** Web Servers (Gunicorn, uWSGI), Reverse Proxy (Nginx)
- **Containerization:** Docker, Docker Compose
- **CI/CD:** GitHub Actions, GitLab CI, Jenkins
- **Hosting:** Netlify, Vercel, AWS, Google Cloud, Azure
- **Monitoring & Logging:** Prometheus, Grafana, ELK Stack

9. Modern Design Patterns

- **Microservices Architecture:** Breaking down monolithic applications
- **Event-Driven Architecture:** Asynchronous communication
- **Domain-Driven Design (DDD):** Focusing on core domain logic
- **Repository Pattern:** Abstracting data access
- **Factory Pattern:** Creating objects
- **Dependency Injection:** Managing component dependencies
- **CQRS (Command Query Responsibility Segregation):** Separating read and write operations
- **API Gateway:** Centralized entry point
- **Serverless Computing:** AWS Lambda, Google Cloud Functions