Below is a comprehensive MVP Development Document that details the nodes, sub-nodes, tech stack, file structure, and additional components required to build a no-code, drag-and-drop Agentic Workflow Builder Platform that you can run on your local host.

MVP Development Document: No-Code Agentic Workflow Builder Platform

1. Introduction

This document outlines the requirements, architecture, and implementation plan for an MVP of a no-code agentic workflow builder. The platform enables users to visually build workflows by dragging and dropping pre-defined nodes (as detailed in the Sub-Nodes document citeturn1file0), and later integrates additional nodes from the smol agent library. In Phase 1, the MVP will also include integration API widgets (starting with 50 popular integrations) and pre-built modules for advanced functions like image analysis and AI calling.

---

2. Architecture Overview

2.1 High-Level Design

- **Modular & Microservices-Based:**
  Separate the application into front-end and back-end services, allowing each node type (trigger, input/data source, transformation, etc.) to scale and evolve independently.

- **API-First Approach:**
  Design RESTful (or GraphQL) APIs for managing user workflows, node configurations, integrations, and pre-built modules.

- **Plugin/Connector Framework:**
  Develop a standardized connector interface that simplifies adding both core nodes and third-party integrations.

- **Local Development Focus:**
  Use Docker (and Docker Compose) to containerize services for easy local deployment and testing.

---

3. Node & Sub-Node Definitions

3.1 Node Categories and Their Sub-Nodes

A. Trigger Nodes

- **Webhook Listener:**
  • Listens for HTTP requests.
  • Configurable authentication and validation.

- **Event Listener:**
  • Subscribes to events from external systems (e.g., queues, database updates).
  • Supports filtering and processing.

- **Conditional Trigger:**
  • Activates workflows based on specific conditions in the payload, headers, or metadata.

B. Input/Data Source Nodes

- **API Connector:**
  • Fetches data from external REST or GraphQL APIs.
  • Handles authentication (OAuth, API keys), pagination, and rate limiting.

- **Database Connector:**
  • Connects to SQL/NoSQL databases.
  • Supports custom queries and indexing for optimization.

- **File Ingestor:**
  • Reads data from CSV, JSON, XML files in cloud storage (S3, Google Drive).
  • Supports file format transformation.

- **Streaming Data Listener:**
  • Consumes real-time data from Kafka, WebSockets, MQTT, etc.
  • Buffers or batches incoming messages.

- **Caching Layer:**
  • Temporarily stores fetched data (with TTL) to reduce redundant calls.

C. Data Mapping/Transformation Nodes

- **Field Mapper:**
  • Maps source fields to target fields with aliasing and nesting capabilities.

- **Data Type Converter:**
  • Converts between data types (e.g., string to integer).

- **Data Formatter:**
  • Applies formatting rules (e.g., date normalization, currency conversion).

- **Data Validator:**
  • Checks for required fields and validates formats.

- **Conditional Transformer:**
  • Transforms data conditionally (e.g., different formatting based on country code).

- **Aggregation & Calculation Node:**
  • Computes derived values (sums, averages, percentages).

- **Custom Scripting Node:**
  • Allows custom JavaScript or Python code for advanced transformations.

D. Conditional/Decision Nodes

- **Expression Evaluator:**
  • Evaluates logical expressions using operators like >, <, ==, &&, ||.

- **Pattern Matcher:**
  • Uses regex or predefined patterns to classify data.

E. Loop/Iteration Nodes

- **Iterator Controller:**
  • Manages looping through collections, with sequential or parallel execution.

- **Item Variable Mapper:**
  • Maps each item in a collection to an iteration variable.

F. Action/Integration Nodes

- **API Request Handler:**
  • Sends HTTP requests (GET, POST, etc.) with support for various authentication

methods.

- **Payload Formatter:**
  • Structures data into the required format (JSON, XML, etc.).

- **Rate Limiter:**
  • Controls request frequency and handles backoff strategies.

- **Batch Processor:**
  • Groups actions for batch API calls.

- **Response Validator:**
  • Checks API responses for success and manages error handling.

G. Output/Result Nodes

- **Notification Dispatcher:**
  • Sends notifications via email, SMS, Slack, push notifications.

- **Dashboard Updater:**
  • Updates analytics dashboards dynamically.

- **Data Storage Writer:**
  • Persists final results to databases or file storage.

H. Scheduler/Timer Nodes

- **Cron Scheduler:**
  • Triggers actions based on cron expressions.

- **Fixed Delay Timer:**
  • Adds a fixed delay between steps.

- **Dynamic Timer:**
  • Adjusts delays dynamically based on runtime conditions.

- **Recurring Timer:**
  • Repeatedly triggers actions at set intervals.

- **Event-Based Timer:**
  • Waits for specific events before proceeding.

- **Timeout Handler:**
  • Cancels workflows if actions exceed a specified time limit.

- **Time Zone Converter:**
  • Adjusts scheduled actions according to time zones.

I. Common Sub-Nodes (Applicable Across Nodes)

- **Retry & Fallback Mechanism:**
  • Automatically retries failed actions and falls back to alternative integrations if necessary.

- **Error Handling Node:**
  • Manages errors for each node and logs issues.

- **Cleanup Handler:**
  • Performs cleanup tasks after workflow execution.

Reference: Sub-Nodes document citeturn1file0

3.2 Integration API Widgets (Initial 50 Popular Integrations)

1. WhatsApp
2. Telegram
3. Instagram
4. LinkedIn
5. Facebook
6. Twitter
7. Slack
8. Microsoft Teams
9. Gmail
10. Google Calendar
11. Google Drive
12. Google Sheets
13. Zoom
14. Salesforce
15. HubSpot
16. Shopify
17. WooCommerce
18. Stripe
19. PayPal
20. QuickBooks
21. Xero

22. Trello
23. Asana
24. Monday.com
25. Jira
26. GitHub
27. GitLab
28. Bitbucket
29. Dropbox
30. Box
31. OneDrive
32. Amazon S3
33. Google Analytics
34. Mixpanel
35. Segment
36. Mailchimp
37. Constant Contact
38. SendGrid
39. Twilio
40. Zendesk
41. Freshdesk
42. Intercom
43. Pipedrive
44. Marketo
45. Adobe Creative Cloud
46. Notion
47. Airtable
48. Smartsheet
49. SurveyMonkey
50. Typeform

3.3 Pre-Built Modules

- **Image Analysis Module:**
  AI-powered object detection, facial recognition, and sentiment analysis.

- **Image Generation Module:**
  Uses generative AI to create images based on user prompts.

- **AI Calling Module:**
  Integrates with the Bland API (or similar) for AI-powered voice calling, transcription, and automated responses.

- **Workflow Logging Module:**
  Provides detailed logging and telemetry of workflow executions for debugging and

performance tracking.

---

4. Technology Stack

4.1 Front-End

- **Framework:** React.js
- **State Management:** Redux or Context API
- **UI Libraries:** Material-UI, Ant Design, or Bootstrap
- **Drag-and-Drop:** React Flow, React DnD
- **Routing:** React Router

4.2 Back-End

- **Server Environment:** Node.js with Express.js
- **API:** RESTful APIs (or GraphQL using Apollo Server)
- **Database:** MongoDB for flexibility or PostgreSQL for relational needs
- **Authentication:** JWT-based authentication
- **Message Queue:** RabbitMQ or Kafka for event-driven processing

4.3 AI/ML Components

- **Integration with Smol Agents:**
  Use Hugging Face's APIs to enable self-correcting logic, guided tours, and multi-agent orchestration.

- **Image/Video Processing Libraries:**
  TensorFlow.js or PyTorch (if needed for advanced processing)

4.4 DevOps & Infrastructure

- **Containerization:** Docker (with Docker Compose for local setup)
- **CI/CD:** GitHub Actions, Jenkins, or similar
- **Version Control:** Git
- **Local Development:** Node.js server running on localhost, optionally containerized via Docker

4.5 Logging & Testing

- **Logging:** Winston or Morgan (backend), browser console/logging libraries (frontend)

- **Testing:** Jest for unit tests, Cypress for end-to-end tests

---

5. File Structure & Code Organization

A recommended file structure for the project is as follows:

```
/no-code-agent-builder
├── /backend
│   ├── /src
│   │   ├── /controllers       # Business logic and API endpoints
│   │   ├── /models            # Database schemas and models
│   │   ├── /routes            # API route definitions
│   │   ├── /services          # Service layers (node execution, integrations, AI modules)
│   │   ├── /config            # Environment and configuration files (database, auth, etc.)
│   │   ├── /middlewares       # Express middlewares (error handling, logging, auth)
│   │   └── app.js             # Main Express application entry point
│   ├── /tests                 # Backend unit and integration tests
│   ├── package.json
│   └── Dockerfile             # Docker configuration for the backend service
├── /frontend
│   ├── /public                # Static assets (index.html, images, etc.)
│   ├── /src
│   │   ├── /components        # Reusable React components (node library, canvas, sidebar, toolbar)
│   │   ├── /pages             # Page components (editor, dashboard)
│   │   ├── /store             # Redux store and reducers (or other state management)
│   │   ├── /services          # API service calls to the backend
│   │   ├── /styles            # CSS/SCSS files for styling
│   │   ├── /utils             # Utility functions (drag-and-drop helpers, formatters)
│   │   └── index.js           # Main entry point for the React application
│   ├── package.json
│   └── Dockerfile             # Docker configuration for the frontend service
├── /docs
│   └── MVP_Development_Document.md  # This detailed development document
├── docker-compose.yml         # Compose file to run both frontend and backend services locally
└── README.md                  # Project overview and setup instructions
```

Explanation

- **Backend:**
  All business logic, data models, API endpoints, and services reside in the `/backend` folder.

- **Frontend:**
  Contains the React application, UI components, state management, and styling files.

- **Docs:**
  Additional documentation, guides, and architectural decisions.

- **Docker & Compose:**
  The Dockerfiles and docker-compose.yml allow for easy local deployment and containerized development.

---

6. MVP Implementation Roadmap

6.1 Phase 1 – Core MVP Development

1. **UI/IDE Core Implementation:**
    - Build a drag-and-drop interface using React and libraries like React Flow.
    - Develop the left sidebar (node library) that displays all the nodes as defined in the Sub-Nodes document.
    - Create the main canvas with grid, snap-to-grid, and connection lines.
2. **Backend & API Setup:**
    - Develop RESTful API endpoints in Node.js/Express to support node configuration, workflow execution, and integrations.
    - Implement database models (using MongoDB or PostgreSQL) to store user workflows and node configurations.
    - Create the plugin framework for node execution and integration management.
3. **Node Functionality:**
    - Implement basic functionality for all core node types (Triggers, Input/Data Source, Transformation, etc.) with error handling and configuration options.
    - Include common sub-nodes (Retry & Fallback, Error Handling, Cleanup).
4. **Integration API Widgets & Pre-Built Modules:**
    - Integrate initial 50 popular API widgets (e.g., WhatsApp, Telegram, Instagram, etc.).

    - Develop pre-built modules such as Image Analysis, Image Generation, AI Calling (via Bland API), and Workflow Logging.
5. **Smol Agent Node Exploration:**
    - Identify and prototype additional nodes from the smol agent library (e.g., guided tours, self-correcting logic, multi-agent orchestration).
6. **Testing & Feedback:**
    - Conduct internal user testing for the drag-and-drop interface and node configurations.
    - Collect feedback to refine node behaviors and integration functionalities.

6.2 Future Phases

- Expand the node library beyond the initial set.
- Increase the number of integrations from 50 to 500+.
- Enhance AI capabilities with more sophisticated decision nodes and multi-agent orchestration.
- Move toward production deployment on cloud infrastructure after successful local testing.

---

7. Running the MVP on Localhost

Prerequisites

- Node.js (with npm or yarn)
- Docker & Docker Compose (if containerizing)

Steps

Clone the Repository:

```
 git clone <repository_url>
cd no-code-agent-builder
```

1.
2. **Run the Backend:**

Without Docker:

```
 cd backend
npm install
npm start  # or node app.js
```

    -

With Docker:
 docker build -t agent-builder-backend .
docker run -p 5000:5000 agent-builder-backend

    -

3. **Run the Frontend:**

Without Docker:
 cd frontend
npm install
npm start # runs on localhost:3000 by default

    -

With Docker:
 docker build -t agent-builder-frontend .
docker run -p 3000:3000 agent-builder-frontend

    -

4. **Using Docker Compose:**

    - Ensure `docker-compose.yml` is correctly configured (as per the file structure).

Run:
 docker-compose up --build

    -

5. **Access the Application:**

    - Open a browser and navigate to `http://localhost:3000` for the frontend UI.
    - The backend API is accessible at `http://localhost:5000`.

---

8. Additional Considerations

- **Environment Management:**
  Use `.env` files to manage API keys, database URLs, and other sensitive configuration variables.

- **Testing:**
  Set up Jest for backend unit tests and Cypress for end-to-end testing of the frontend.

- **Documentation & CI/CD:**
  Maintain up-to-date documentation in the `/docs` folder and configure a CI/CD pipeline using GitHub Actions or a similar tool.

- **Version Control & Collaboration:**
  Use Git for source control and follow best practices for branching, code reviews, and collaboration.

---

9. Conclusion

This document provides a detailed blueprint for building an MVP of the No-Code Agentic Workflow Builder Platform that runs on localhost. By following this plan—covering nodes and sub-nodes, the chosen tech stack, file structure, and step-by-step deployment instructions—you will create a robust foundation that can be expanded upon to support thousands of integrations and advanced AI features in future iterations.

Happy coding!