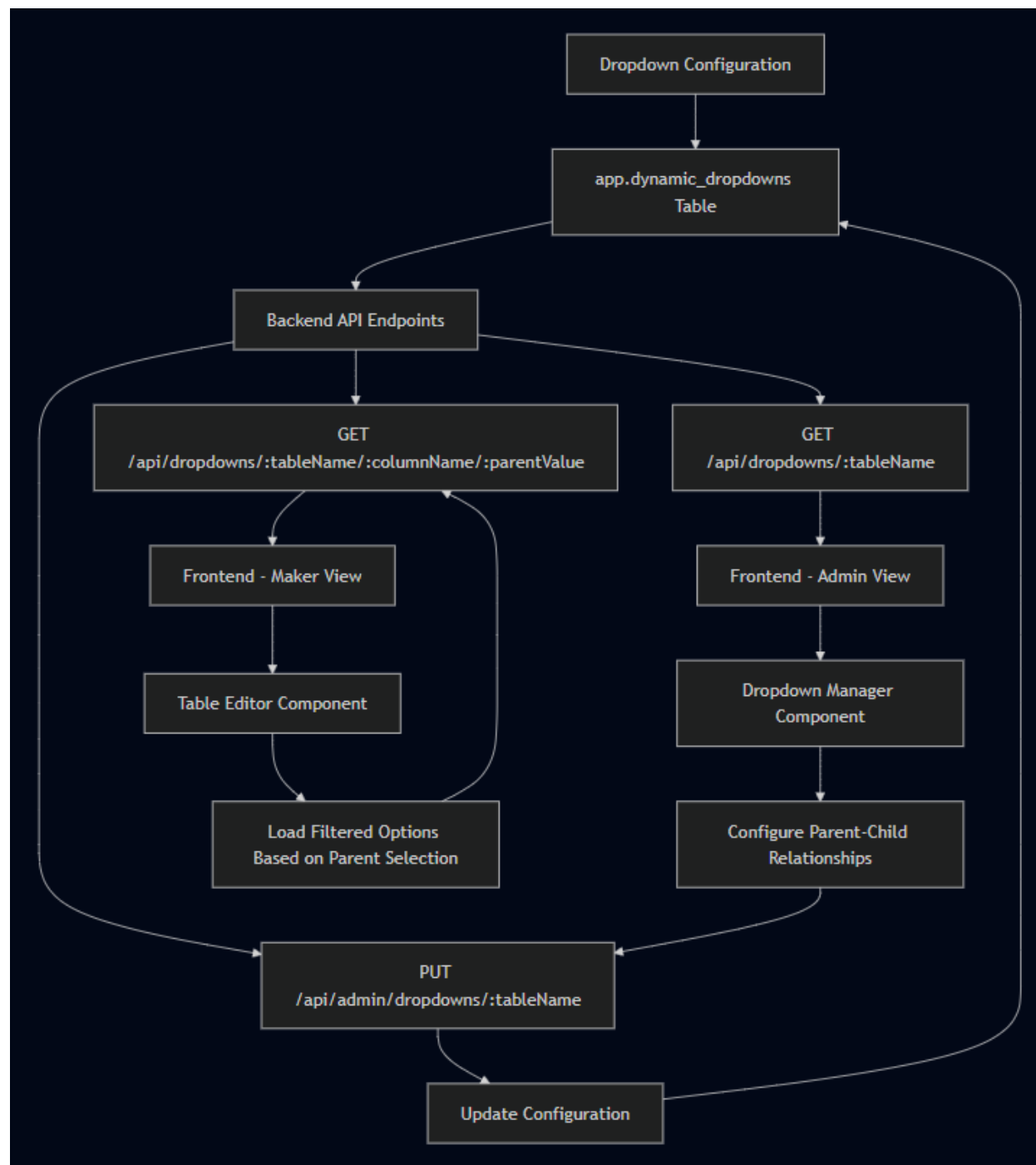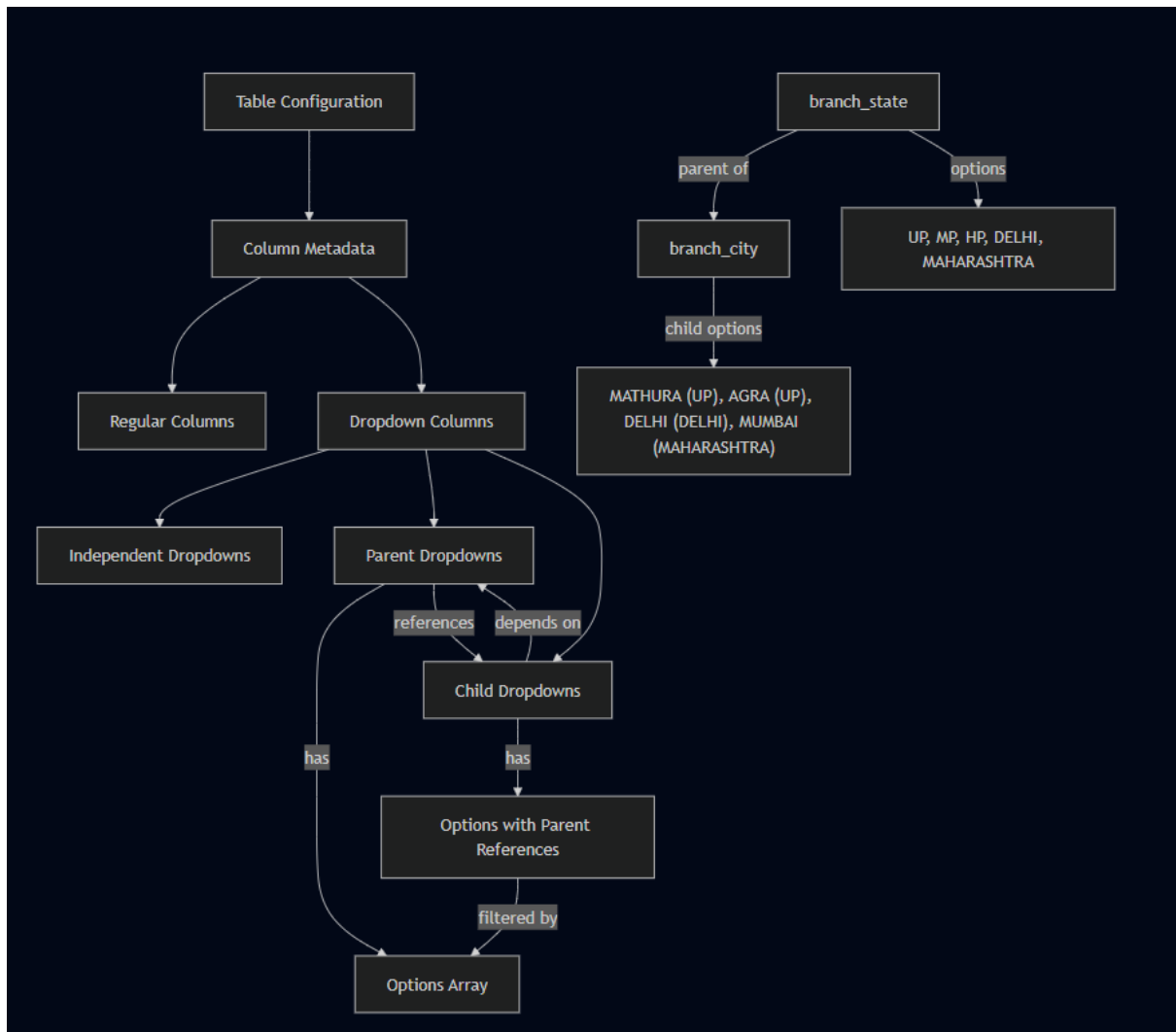# Building Dependent Dropdown Feature for Dropdown Manager
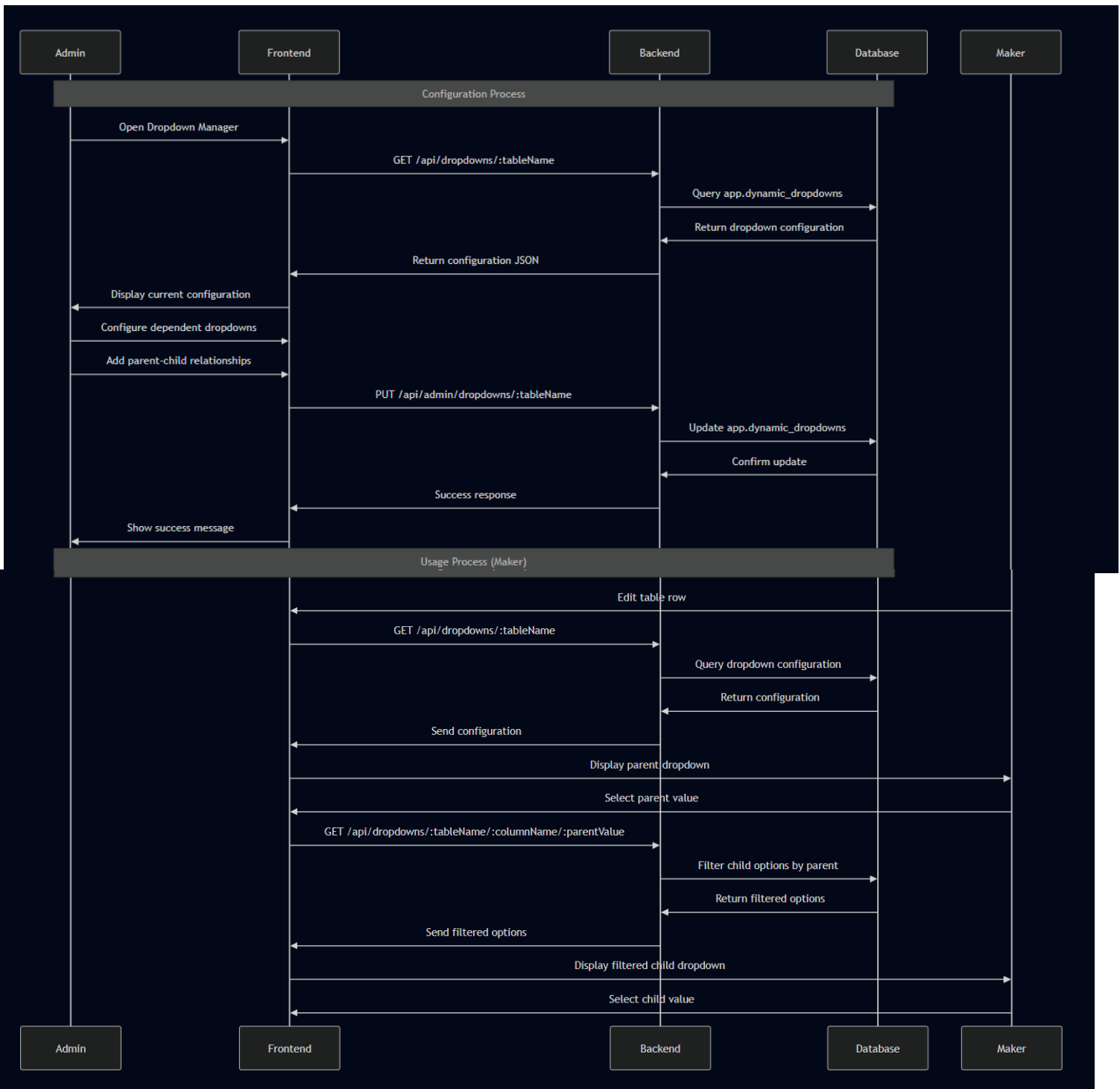
Overview

The dependent dropdown feature will allow admin users to:

1. Designate a column as dependent on another (parent) column

2. Define which parent values should display which child options

3. Save these configurations to the database

4. Retrieve filtered options based on selected parent values

```mermaid
graph TD
    A[Table Configuration] --> B[Column Metadata]
    B --> C[Regular Columns]
    B --> D[Dropdown Columns]
    D --> E[Independent Dropdowns]
    D --> F[Parent Dropdowns]
    D --> G[Child Dropdowns]
    F -- references --> G
    G -- depends on --> F
    F -- has --> H[Options Array]
    G -- has --> I[Options with Parent References]
    I -- filtered by --> H

    J[branch_state] -- parent of --> K[branch_city]
    J -- options --> L[UP, MP, HP, DELHI, MAHARASHTRA]
    K -- child options --> M[MATHURA (UP), AGRA (UP), DELHI (DELHI), MUMBAI (MAHARASHTRA)]
```

Table Configuration → Column Metadata

Column Metadata → Regular Columns, Dropdown Columns

Dropdown Columns → Independent Dropdowns, Parent Dropdowns

Parent Dropdowns — references → Child Dropdowns

Child Dropdowns — depends on → Parent Dropdowns

Parent Dropdowns — has → Options Array

Child Dropdowns — has → Options with Parent References

Options with Parent References — filtered by → Options Array

branch_state — parent of → branch_city

branch_state — options → UP, MP, HP, DELHI, MAHARASHTRA

branch_city — child options → MATHURA (UP), AGRA (UP), DELHI (DELHI), MUMBAI (MAHARASHTRA)

Data Structure

The current dropdown_options in app.dynamic_dropdowns table needs to be extended to support parent-child relationships:

[

 {

  "columnName": "branch_state",

```
    "options": ["UP", "MP", "HP", "DELHI", "MAHARASHTRA"],

    "dependentColumns": ["branch_city"]

  },

  {

    "columnName": "branch_city",

    "options": [

      {"value": "MATHURA", "parent": "UP"},

      {"value": "AGRA", "parent": "UP"},

      {"value": "ALIGARH", "parent": "UP"},

      {"value": "DELHI", "parent": "DELHI"},

      {"value": "MUMBAI", "parent": "MAHARASHTRA"}

    ],

    "parentColumn": "branch_state"

  }

]
```

Backend API Endpoints

1. GET /api/dropdowns/:tableName

2. GET /api/dropdowns/:tableName/:columnName/:parentValue

3. PUT /api/admin/dropdowns/:tableName

Flow Diagram

1. **Admin Flow:**

   - Admin opens dropdown management interface

   - Selects table and column to configure

   - Toggles "dependent dropdown" option if needed

   - If dependent, selects parent column

   - For each parent value, adds child options

   - Saves configuration

2. **Maker Flow:**

- Maker edits a row in a table

- System checks if column has dropdown options

- If column has a parent dependency, initially shows only parent dropdown

- When parent value is selected, child dropdown is populated with filtered options

- Maker selects appropriate value from filtered options

Edge Cases to Handle

1. **Parent value has no children:**

- Show an empty dropdown with a message "No options available for this parent value"

2. **Parent column is deleted:**

- Check all dependent columns and update their configuration

- Either convert them to regular dropdowns or remove dependency

3. **Parent option is removed:**

- Handle orphaned child options by either:

- Automatically removing them

- Moving them to an "Unassigned" category

4. **Multiple dependencies:**

- Support nested dependencies (3+ levels) if needed

- Support multiple parent columns for a single child column

5. **Default values:**

- Pre-select most common parent-child combinations when editing

6. **Adding new parent values:**

- When a new parent value is added, provide interface to quickly add child options

7. **Data consistency:**

- When loading existing data, verify parent-child relationships are valid

- Prevent saving inconsistent relationships