DataGrid Application using AG Grid

Aptitude Test - Web App Development Internship
BMW Battery Cell Competence Center

Jobin Roy

November 20, 2025

Repository: github.com/jobz3/AGGridProject

Agenda

Requirements Analysis

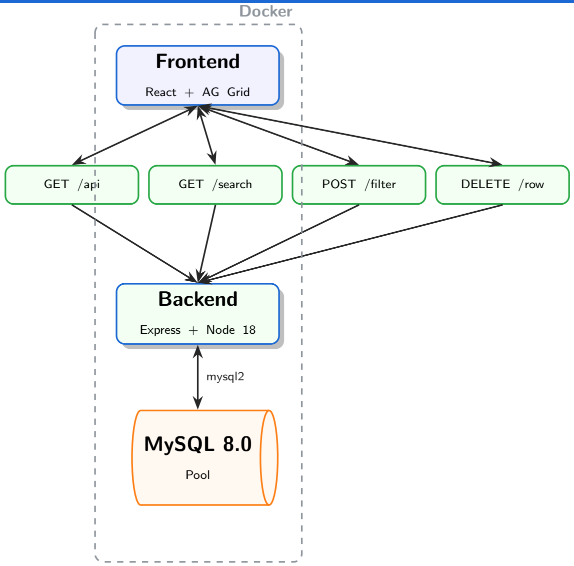
Core Objective

Build a **DataGrid Component** that handles any structural data with N columns, integrating frontend and backend services.

Required Features:

- ✓ N-column support
- ✓ Actions column (View/Delete)
- ✓ Backend-driven Search
- ✓ Backend-driven Filtering
- ✓ MySQL integration
- ✓ React MUI styling

High-Level Architecture



Technology Stack Deep Dive

Frontend

- **React 19.2.0**
  - Latest features
  - Context API
- **AG Grid 34.3.1**
  - Quartz theme
  - Material icons
- **MUI 7.3.5**
  - Components
  - Theming
- **Vite 6.0.1**
  - Fast builds
  - HMR

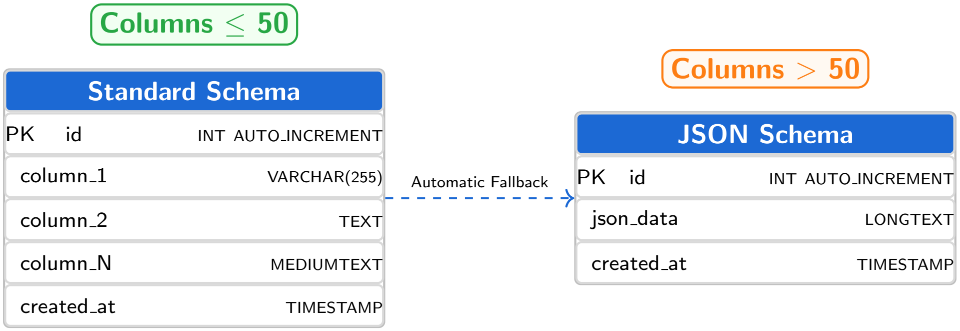
Backend

- **Node.js 18**
  - Alpine image
  - Production ready
- **Express.js 5.1.0**
  - RESTful APIs
  - Middleware
- **mysql2 3.15.3**
  - Connection pool
  - Promise API
- **csv-parser**
  - Stream processing

DevOps

- **Docker**
  - Multi-stage
  - Health checks
- **Docker Compose**
  - Orchestration
  - Networks
- **Nginx Alpine**
  - Static serving
  - SPA routing
- **MySQL 8.0**
  - Volumes
  - Adminer UI

Adaptive Database Schema Strategy



Why Two Schemas?

- **Standard (<50 cols):** Dedicated SQL columns for efficient querying and indexing.
- **JSON (>50 cols):** Prevents MySQL row size limits (65KB max). Data stored as JSON.

Feature 1: DataGrid Component

Dynamic Column Generation

- Reads first data row to extract columns
- Auto-generates `columnDefs` for AG Grid
- Transforms keys: `user_name` → `USER NAME`
- Adds pinned **Actions** column

AG Grid Features:

- Sorting on all columns
- Pagination (10/25/50/100 rows)
- Resizable columns
- Custom cell renderers

Code: DataGrid.jsx

```
const dynamicColDefs = keys
.filter(k => k !== 'id')
.map(key => ({
  field: key,
  headerName: key
    .replace(/_/g, ' ')
    .toUpperCase()
}));
dynamicColDefs.push({
  field: 'actions',
  cellRenderer: ActionCell,
  pinned: 'right'
});
```

Feature 2: View Action & Detail Page

Workflow:

1. User clicks **View** button
2. Navigates to `/detail`
3. Detail page displays all fields
4. Back button returns to DataGrid

ActionCell Component

```
const onView = (e) => {
  e.stopPropagation();
  context.navigate('/detail', {
    state: { row: data }
  });
};
<Button
  variant='contained'
  onClick={onView}
  startIcon={<Visibility />}
>
  View
</Button>
```

MUI Button with Material Icon

Feature 3: Backend-Driven Search

Requirement: Search across ALL columns via backend API

Frontend (DataGrid.jsx)

- TextField with search icon
- 500ms debounce to prevent API spam
- Calls `GET /api/search?query=...`
- Updates grid with filtered results

Backend (routes/data.js)

- Detects schema type
- **Column schema:** Dynamic `OR` clause
- **JSON schema:** Searches within JSON
- Returns matching rows with count

SQL Query Example (Column Schema)

```
SELECT * FROM data WHERE
    `name` LIKE '%search%' OR
    `email` LIKE '%search%';
```

Feature 4: Advanced Filtering System

10 Filter Operators Implemented:

1. **Contains** - Substring match
2. **Equals** - Exact match
3. **Starts With** - Prefix match
4. **Ends With** - Suffix match
5. **Is Empty** - NULL or empty
6. **Is Not Empty** - Has value
7. **Greater Than** - Numeric
8. **Less Than** - Numeric
9. **Greater Than or Equal**
10. **Less Than or Equal**

Multi-Condition Filtering

Users can add multiple filters with AND logic:
Example: `age > 25 AND country = 'Germany'`

UI: FilterModal.jsx (MUI Modal with dynamic filter rows)
API: POST /api/filter with {`filters: [{column, operator, value}]`}

Feature 5: Delete Functionality

Two Delete Entry Points:

1. From DataGrid Actions column
2. From Detail page

Safety Features:

- Confirmation modal
- Loading state during API call
- Success/error notifications
- Transaction-based deletion
- Row ID validation

Backend Implementation:

- `DELETE /api/delete-row`
- MySQL transaction ensures atomicity
- Returns affected row count

Bonus Feature: CSV Upload with Chunking

Problem: Large CSV files (>100MB) cause timeouts
Solution: Frontend splits data into chunks, uploads sequentially

Frontend (CSVUploadModal.jsx)

- Uses PapaParse for CSV parsing
- Splits rows into 5000-row chunks
- Shows progress bar during upload
- Uploads to `/api/push-data-chunked`

Backend (routes/data.js)

- First chunk: DROP and CREATE table
- Subsequent chunks: INSERT in batches
- Transaction per chunk
- Batch inserts (100-1000 rows)

Performance Optimization

Before: 100,000 rows = 100,000 INSERT queries (slow)
After: 100,000 rows = 100 batch INSERT queries (fast)

Docker Multi-Stage Build Architecture

Frontend Dockerfile (3 stages)

1. **Base:** Node 18 Alpine
2. **Builder:** npm ci + Vite build
3. **Runner:** Nginx Alpine
   - Copies `dist/` folder
   - Custom nginx config
   - SPA routing support
   - Proxy `/api` to backend

Image Sizes:

- With dev deps: 500MB
- Multi-stage optimized: 25MB

Backend Dockerfile (3 stages)

1. **Base:** Node 18 Alpine
2. **Deps:** Production deps only
3. **Runner:** Non-root user
   - Creates `nodeuser` (UID 1001)
   - Copies only needed files
   - Health check endpoint

Security Features:

- Non-root containers
- Alpine Linux (minimal)
- No dev dependencies

Code Quality & Best Practices

Frontend Best Practices:

- Component modularity
- Custom hooks
- Error boundaries
- Loading states
- Snackbar notifications
- Debounced search
- React Router navigation

Backend Best Practices:

- MySQL connection pooling
- Transaction management
- SQL injection prevention
  - Parameterized queries
  - Input sanitization
- Error handling & logging
- RESTful API design
- Batch operations
- Environment variables

Security Measures

1. Non-root containers **2.** Prepared statements **3.** CORS config **4.** Input validation

Performance Optimizations

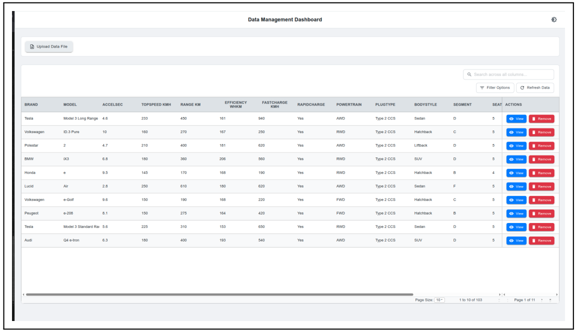
1. **Database Level:**
   - Connection pooling (max 10 connections)
   - Batch inserts (100-1000 rows per query)
   - Indexed `id` column (primary key)
   - Efficient queries with proper WHERE clauses
2. **Backend Level:**
   - Chunked CSV upload (prevents timeout)
   - Transaction-based operations
   - JSON parsing only when necessary
3. **Frontend Level:**
   - Vite for fast builds & HMR
   - Debounced search (500ms)
   - AG Grid virtual scrolling
   - Pagination (reduces DOM nodes)
   - React.memo for expensive components
4. **Docker Level:**
   - Multi-stage builds (smaller images)
   - Alpine Linux base (5MB vs 900MB)
   - nginx for static file serving

Testing & Validation

Manual Testing Performed:

- Addtional testing with CSV uploads (0.5K, 1K, 10K, 100K rows)
- All 10 filter operators
- Search across different column types
- View/Delete from grid and detail page
- Theme switching
- Docker Deployment

Application Screenshots - Core Features



Data Management Dashboard

Upload Data File

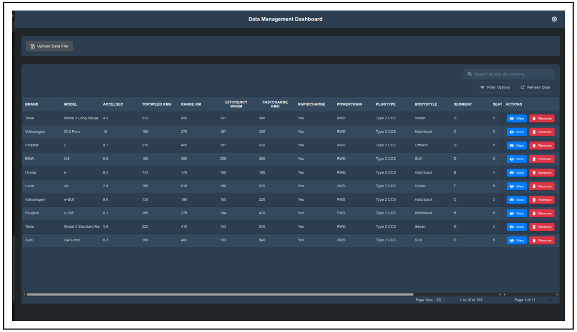
Search across all columns...

Filter Options | Refresh Data

BRAND	MODEL	ACCELSEC	TOPSPEED KMH	RANGE KM	EFFICIENCY WHKM	FASTCHARGE KMH	RAPIDCHARGE	POWERTRAIN	PLUGTYPE	BODYSTYLE	SEGMENT	SEAT	ACTIONS
Tesla	Model 3 Long Range	4.6	233	450	161	940	Yes	AWD	Type 2 CCS	Sedan	D	5	View Remove
Volkswagen	ID.3 Pure	10	160	270	167	250	Yes	RWD	Type 2 CCS	Hatchback	C	5	View Remove
Polestar	2	4.7	210	400	181	620	Yes	AWD	Type 2 CCS	Liftback	D	5	View Remove
BMW	iX3	6.8	180	360	206	560	Yes	RWD	Type 2 CCS	SUV	D	5	View Remove
Honda	e	9.5	145	170	168	190	Yes	RWD	Type 2 CCS	Hatchback	B	4	View Remove
Lucid	Air	2.8	250	610	180	620	Yes	AWD	Type 2 CCS	Sedan	F	5	View Remove
Volkswagen	e-Golf	9.6	150	190	168	220	Yes	FWD	Type 2 CCS	Hatchback	C	5	View Remove
Peugeot	e-208	8.1	150	275	164	420	Yes	FWD	Type 2 CCS	Hatchback	B	5	View Remove
Tesla	Model 3 Standard Ra	5.6	225	310	153	650	Yes	RWD	Type 2 CCS	Sedan	D	5	View Remove
Audi	Q4 e-tron	6.3	180	400	193	540	Yes	AWD	Type 2 CCS	SUV	D	5	View Remove

Page Size: 10 · 1 to 10 of 103 · Page 1 of 11

DataGrid - Light Mode



Data Management Dashboard

Upload Data File

Search across all columns...

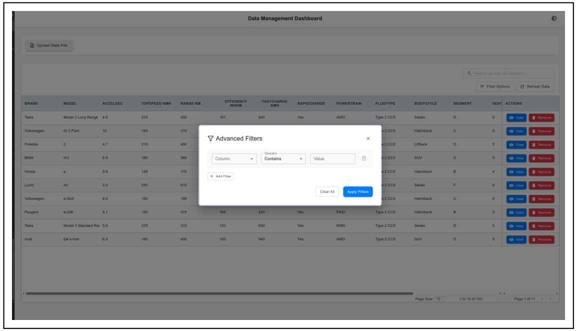
Filter Options | Refresh Data

BRAND | MODEL | ACCELSEC | TOPSPEED KMH | RANGE KM | EFFICIENCY WHKM | FASTCHARGE KMH | RAPIDCHARGE | POWERTRAIN | PLUGTYPE | BODYSTYLE | SEGMENT | SEAT | ACTIONS
Tesla | Model 3 Long Range | 4.6 | 233 | 450 | 161 | 940 | Yes | AWD | Type 2 CCS | Sedan | D | 5 | View Remove
Volkswagen | ID.3 Pure | 10 | 160 | 270 | 167 | 250 | Yes | RWD | Type 2 CCS | Hatchback | C | 5 | View Remove
Polestar | 2 | 4.7 | 210 | 400 | 181 | 620 | Yes | AWD | Type 2 CCS | Liftback | D | 5 | View Remove
BMW | iX3 | 6.8 | 180 | 360 | 206 | 560 | Yes | RWD | Type 2 CCS | SUV | D | 5 | View Remove
Honda | e | 9.5 | 145 | 170 | 168 | 190 | Yes | RWD | Type 2 CCS | Hatchback | B | 4 | View Remove
Lucid | Air | 2.8 | 250 | 610 | 180 | 620 | Yes | AWD | Type 2 CCS | Sedan | F | 5 | View Remove
Volkswagen | e-Golf | 9.6 | 150 | 190 | 168 | 220 | Yes | FWD | Type 2 CCS | Hatchback | C | 5 | View Remove
Peugeot | e-208 | 8.1 | 150 | 275 | 164 | 420 | Yes | FWD | Type 2 CCS | Hatchback | B | 5 | View Remove
Tesla | Model 3 Standard Ra | 5.6 | 225 | 310 | 153 | 650 | Yes | RWD | Type 2 CCS | Sedan | D | 5 | View Remove
Audi | Q4 e-tron | 6.3 | 180 | 400 | 193 | 540 | Yes | AWD | Type 2 CCS | SUV | D | 5 | View Remove

Page Size: 10 · 1 to 10 of 103 · Page 1 of 11

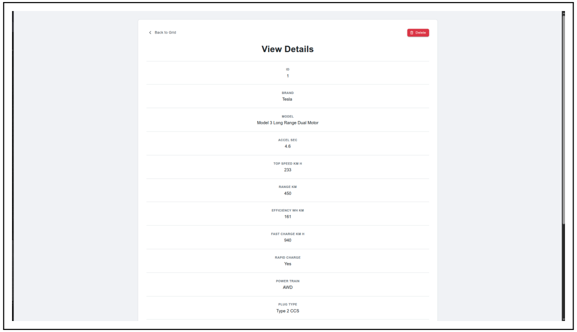
DataGrid - Dark Mode

Application Screenshots - Filtering & Details



BRAND	MODEL	ACCELSEC	TOPSPEED KMH	RANGE KM	EFFICIENCY WHKM	FASTCHARGE KMH	RAPIDCHARGE	POWERTRAIN	PLUGTYPE	BODYSTYLE	SEGMENT	SEAT	ACTIONS
Tesla	Model 3 Long Range	4.6	233	450	161	940	Yes	AWD	Type 2 CCS	Sedan	D	5	View Remove
Volkswagen	ID.3 Pure	10	160	270					2 CCS	Hatchback	C	5	View Remove
Polestar	2	4.7	210	400					2 CCS	Liftback	D	5	View Remove
BMW	iX3	6.8	180	360					2 CCS	SUV	D	5	View Remove
Honda	e	9.5	145	170					2 CCS	Hatchback	B	4	View Remove
Lucid	Air	2.8	250	610					2 CCS	Sedan	F	5	View Remove
Volkswagen	e-Golf	9.6	150	190					2 CCS	Hatchback	C	5	View Remove
Peugeot	e-208	8.1	150	275	164	420	Yes	FWD	Type 2 CCS	Hatchback	B	5	View Remove
Tesla	Model 3 Standard Ra	5.6	225	310	153	650	Yes	RWD	Type 2 CCS	Sedan	D	5	View Remove
Audi	Q4 e-tron	6.3	180	400	193	540	Yes	AWD	Type 2 CCS	SUV	D	5	View Remove

Filter Modal

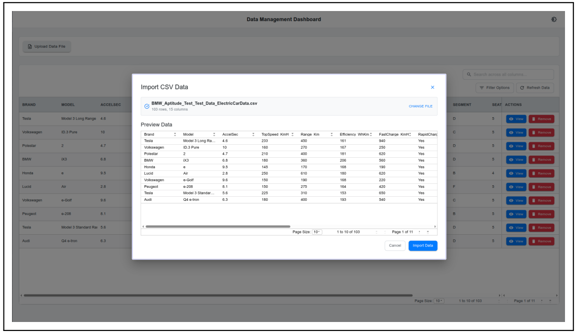


View Details

ID	1
BRAND	Tesla
MODEL	Model 3 Long Range Dual Motor
ACCEL SEC	4.6
TOP SPEED KM H	233
RANGE KM	450
EFFICIENCY WH KM	161
FAST CHARGE KM H	940
RAPID CHARGE	Yes
POWER TRAIN	AWD
PLUG TYPE	Type 2 CCS

Detail Page

Application Screenshots - Data Management



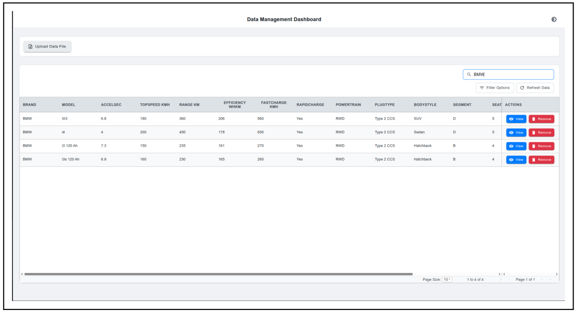
Import CSV Data

BMW_Aptitude_Test_Test_Data_ElectricCarData.csv
103 rows, 15 columns

Preview Data

Brand	Model	AccelSec	TopSpeed KmH	Range Km	Efficiency WhKm	FastCharge KmH	RapidCharg
Tesla	Model 3 Long Ra...	4.6	233	450	161	940	Yes
Volkswagen	ID.3 Pure	10	160	270	167	250	Yes
Polestar	2	4.7	210	400	181	620	Yes
BMW	iX3	6.8	180	360	206	560	Yes
Honda	e	9.5	145	170	168	190	Yes
Lucid	Air	2.8	250	610	180	620	Yes
Volkswagen	e-Golf	9.6	150	190	168	220	Yes
Peugeot	e-208	8.1	150	275	164	420	Yes
Tesla	Model 3 Standar...	5.6	225	310	153	650	Yes
Audi	Q4 e-tron	6.3	180	400	193	540	Yes

Page Size: 10 1 to 10 of 103 Page 1 of 11



BRAND	MODEL	ACCELSEC	TOPSPEED KMH	RANGE KM	EFFICIENCY WHKM	FASTCHARGE KMH	RAPIDCHARGE	POWERTRAIN	PLUGTYPE	BODYSTYLE	SEGMENT	SEAT	ACTIONS
BMW	iX3	6.8	180	360	206	560	Yes	RWD	Type 2 CCS	SUV	D	5	View Remove
BMW	i4	4	200	450	178	650	Yes	RWD	Type 2 CCS	Sedan	D	5	View Remove
BMW	i3 120 Ah	7.3	150	235	161	270	Yes	RWD	Type 2 CCS	Hatchback	B	4	View Remove
BMW	i3s 120 Ah	6.9	160	230	165	260	Yes	RWD	Type 2 CCS	Hatchback	B	4	View Remove

Page Size: 10 1 to 4 of 4 Page 1 of 1

CSV Upload Modal **Search Functionality**

Video Demo: Link to demo

Key Learnings & Takeaways

1. **Database Design Matters:**
   - Early consideration of schema limits saved time
   - Flexible architecture allows for edge cases
2. **User Experience is Critical:**
   - Loading states prevent user confusion
   - Clear error messages improve debugging
3. **Performance Optimization:**
   - Batch operations dramatically reduce latency
   - Debouncing prevents API overload
4. **Modern React:**
   - Context API sufficient for simple state
   - Hooks make code cleaner and reusable

Conclusion

Project Summary

Successfully delivered a **production-ready**, **scalable**, and **maintainable** DataGrid application.

Key Achievements:

- Component handling any large number of columns
- Complete backend API with searching and filtering
- UI/UX with theme support
- Docker deployment

Thank You!

Looking forward to discussing this project
and exploring next steps.

Backup: API Endpoints Reference

Method	**Endpoint**	**Description**
GET	/api/	Retrieve all data from table
GET	/api/search?query=X	Search across all columns
POST	/api/filter	Apply advanced filters
POST	/api/push-data	Upload CSV (single request)
POST	/api/push-data-chunked	Upload CSV (chunked)
DELETE	/api/delete-row	Delete rows by ID

Backup: Environment Variables

Variable	**Default**	**Description**
PORT	3000	Backend server port
DB_ROOT_PASS	-	MySQL root password
DB_NAME	project	Database name
DB_USER	-	MySQL user
DB_PASS	-	MySQL password
DB_PORT	3306	MySQL port
TABLE_NAME	data	Target table name
BACKEND_URL	localhost:3000/api	API base URL