Full Stack Application — Registration Status Dashboard

A full stack application with an **Angular frontend**, **Node.js/Express backend**, and **MySQL database** to track and visualize registration status counts with optional date filtering.

Prerequisites

- Node.js and npm installed
- MySQL installed and running
- Angular CLI installed globally:

```
npm install -g @angular/cli
```

Setup Instructions

1. Database Setup

1. Start MySQL.
2. Create a new database:

```
CREATE DATABASE test_db;
```

3. Update credentials in `/backend/.env`.

2. Backend Setup

1. Navigate to the backend folder:

```
cd backend
```

2. Install dependencies:

```
npm install
```

3. Create a `.env` file:

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=yourpassword
DB_NAME=test_db
PORT=3000
CSV_PATH=./data/TABLES(REGISTRATION STATUS HISTORY).csv
```

4. Seed database with CSV data:

```
npm run seed
```

5. Start backend server:

```
npm run dev
```

Backend runs at: `http://localhost:3000`

3. Frontend Setup

1. Navigate to the frontend folder:

```
cd frontend
```

2. Install dependencies:

```
npm install
```

3. Start frontend server:

```
ng serve --proxy-config proxy.conf.json
```

Frontend runs at: `http://localhost:4200`

Project Structure

```
registration-dashboard/
├── backend/
│   ├── src/
│   │   ├── app.js          # Main Express server
│   │   ├── db.js           # Database connection
│   │   └── seed.js         # CSV seeder
│   ├── data/
│   │   └── TABLES(REGISTRATION STATUS HISTORY).csv
│   ├── .env
│   ├── package.json
│   └── README.md
└── frontend/
    ├── src/
    │   ├── app/
    │   │   ├── components/
    │   │   │   ├── dashboard.component.ts
    │   │   │   ├── status-card.component.ts
    │   │   │   ├── search-bar.component.ts
    │   │   │   └── navbar.component.ts
    │   │   ├── services/
    │   │   │   └── status.service.ts
    │   │   └── app.config.ts
    │   ├── index.html
    │   └── styles.css
    ├── proxy.conf.json
    ├── package.json
    ├── angular.json
    └── README.md
```

🗄️ Database Design

Table: `registration_status_history`

Column Name	Data Type	Constraints	Description
REGISTRATION_STATUS_ID	BIGINT	PRIMARY KEY	Unique identifier for status history
REGISTRATION_ID	BIGINT		Reference to registration record
STATUS	VARCHAR(255)		Current status of registration
DATE_CREATED	DATE		Date when status was recorded

SQL Table Creation

```
CREATE TABLE IF NOT EXISTS registration_status_history (
  REGISTRATION_STATUS_ID BIGINT PRIMARY KEY,
  REGISTRATION_ID BIGINT,
  STATUS VARCHAR(255),
  DATE_CREATED DATE
);
```

🔌 API Endpoints

1. Health Check

GET `/api/test`
Response:

```
{
  "message": "Backend is working!"
}
```

2. Get Status Counts (Fixed)

GET `/api/status-counts?date=YYYY-MM-DD`

Parameters:

- `date` (optional): Filter results for a specific date.

Example Response:

```
[
  { "status": "Documents Received", "count": 2 },
  { "status": "Send Docs to TTG", "count": 3 },
  { "status": "TTG sent to county", "count": 3 }
]
```

Backend SQL Fix:

```
SELECT STATUS as status, COUNT(*) as count
FROM registration_status_history
WHERE DATE_CREATED = ? -- optional filter
GROUP BY STATUS
ORDER BY count DESC;
```

Backend Code Example (Express.js):

```
app.get("/api/status-counts", async (req, res) => {
  const { date } = req.query;
  let query = `
    SELECT STATUS as status, COUNT(*) as count
    FROM registration_status_history
  `;
  const params = [];
  if (date) {
    query += " WHERE DATE_CREATED = ?";
    params.push(date);
  }
  query += " GROUP BY STATUS ORDER BY count DESC";

  try {
```

```
    const [rows] = await db.execute(query, params);
    res.json(rows);
  } catch (err) {
    console.error(err);
    res.status(500).json({ error: "Internal Server
Error" });
  }
});
```

🎨 Frontend Components

- **DashboardComponent**: Displays status cards and date filtering.
- **StatusCardComponent**: Displays individual status count cards with color coding.
- **SearchBarComponent & NavbarComponent**: Navigation and search functionality.

🔄 Development Workflow

Start backend:

```
cd backend
npm run dev
```

Start frontend:

```
cd frontend
ng serve --proxy-config proxy.conf.json
```

🐛 Troubleshooting

- **MySQL Connection Error**: Verify MySQL is running.
- **Port Already in Use**: Kill process or change port in `.env`.
- **CSV File Not Found**: Ensure path in `.env` is correct.
- **CORS Issues**: Ensure proxy and backend CORS setup is correct.