🌤️Weather Dashboard Full Project – Task Dependency Diagram

Legend

- → = depends on / must be completed after
- **(parallel)** = can be developed concurrently

Phase 1 – Backend Foundation (Epic 2: Backend/ETL)

1. **Setup Flask App Factory and Environment (.env, Config)**
2. **Database Modeling (PostgreSQL / SQLite)** → 1
3. **Alembic or SQLAlchemy Table Creation** → 2
4. **Basic API Routes: /summary_data, /graph_data** → 3
5. **CORS & Axios Connection Testing** → 4 (parallel with Frontend Layout)

Phase 2 – Data Pipeline / ETL (Epic 2 Continued)

1. **Raw Data Fetch Scripts (Weather API / JSON)** → 2
2. **Data Cleaning and Normalization (process_weather_data.py)** → 6
3. **Aggregation Scripts (hourly/daily)** → 7
4. **Pipeline Runner (run_pipeline.py)** → 8
5. **Log Management + Error Handling** → 9

Phase 3 – Frontend Framework (Epic 1: WeatherDashboard v2)

1. **Layout.js and Navbar.js Base Components** → (parallel with 1–3)
2. **React Router Setup (App.js, Routes)** → 11
3. **Page Creation (WeatherDashboard.js, Results.js, About.js)** → 12
4. **Axios Base URL & .env Integration** → 5 + 12
5. **Responsive Grid + Tailwind/Framer Motion Styling** → 13
6. **Deploy Test Build to Vercel (.env.production)** → 15

Phase 4 – Visualization Features (Epic 1 Continued)

1. **Wind Speed API Integration (/graph_data?metric=wind_speed_avg)** → 4 + 14
2. **WindLineChart Component (Recharts or D3)** → 17
3. **Directional Arrows Overlay (WindDir Layer)** → 18
4. **Summary Cards (Temp/Wind/Humidity)** → 18 (parallel)
5. **Table Data Integration (/table_data)** → 4 + 14
6. **Frontend Data Validation / Error States** → 20 + 21

Phase 5 – Reliability, Monitoring & Deployment (Epic 3)

1. **ETL Logging + Data Quality Checks** → 10
2. **Caching + API Response Optimization** → 23
3. **Results Page Analytics Dashboard (Radar/Charts)** → 22 + 24
4. **Deployment to Render + Vercel Linked Build** → 25
5. **Automated Jobs / Cron Integration (Vercel Cron or Task Scheduler)** → 26
6. **Final Lighthouse & CORS Testing** → 26

🪱 Parallel Tracks Summary

Track	Focus	Runs Parallel To
Backend Core	DB + Flask routes	Frontend layout setup
ETL Chain	Fetch → Process → Aggregate	API route setup
Frontend UI	Layout → Routing → API binding	Once /api endpoints are functional
Reliability Layer	Logging, caching, and QA	After data & API stability

---

🐈 Overall Flow (Simplified)

```
Backend Config → DB Model → ETL → API → Frontend Layout → Charts → Reliability → Deployment
```

Would you like a **visual (graphical)** version next — e.g., a dependency flowchart or Gantt-style timeline (SVG or PNG)?