Cloudline Weather Dashboard (MERN) - Complete Project Summary

1. Project Overview
Cloudline Weather is a MERN full-stack weather dashboard with:
- Personalized city weather search and favorites
- Live location weather support (browser geolocation)
- Interactive precipitation map (Leaflet + OpenWeather tiles)
- Premium plans (Ads Free, Basic, Pro) with Razorpay checkout
- User authentication (email/password + Google sign-in)
- User-specific favorites in MongoDB
- Advanced weather cards and animated background themes based on weather and tim
e

2. Tech Stack
Frontend
- React (Vite)
- Chart.js + react-chartjs-2
- Leaflet + react-leaflet
- Tailwind CSS (configured)
- Custom CSS animations (aurora, stars, clouds, rain/snow canvas)

Backend
- Node.js + Express
- MongoDB + Mongoose
- Axios (external API calls)
- JWT authentication
- bcryptjs password hashing
- Google OAuth token verification
- Razorpay payment integration

APIs Used
- OpenWeather (Current + Forecast + Geocoding + Weather Map Tiles)
- ExchangeRate-API (USD to INR conversion for pricing)
- Razorpay (payment order and verification)
- Google Identity (OAuth login)

3. Core Features Implemented
Weather Features
- City search weather (`/api/weather?city=...`)
- Hourly and daily forecast cards
- Dynamic weather icons (amCharts icon pack)
- Weather-dependent animated background
- Live location weather (`/api/weather?lat=...&lon=...`)

User Features
- Email/password signup and login
- Google login
- Profile menu with active plan status and logout
- User-specific favorites (save/load/delete)

Premium & Payments
- Monthly and Annual plan toggle
- Plan activation with expiry tracking
- Pro upgrade discount if Basic is already active
- Pro auto-includes Basic benefits
- Razorpay order creation + payment signature verification

Pro-only Data/UI
- Precipitation summary with mm + POP
- Real precipitation map (Leaflet + OpenWeather tile overlay)
- Moon information cards

4. Database Models
User Model
- `email` (unique)
- `passwordHash`

Favorite Model
- `city`
- `cityNormalized`
- `user` (ObjectId reference)
- Unique compound index: `(user, cityNormalized)`

5. Final Frontend Dependencies
From `frontend/package.json`:
- chart.js: `^4.5.1`
- leaflet: `^1.9.4`
- react: `^19.2.0`
- react-chartjs-2: `^5.3.1`
- react-dom: `^19.2.0`
- react-leaflet: `^5.0.0`

Dev dependencies:
- @vitejs/plugin-react: `^5.1.1`
- vite: `^7.3.1`
- tailwindcss: `^3.4.17`
- postcss: `^8.5.6`
- autoprefixer: `^10.4.24`
- eslint + related plugins

6. Final Backend Dependencies
From `backend/package.json`:
- axios: `^1.13.5`
- bcryptjs: `^3.0.3`
- cors: `^2.8.6`
- dotenv: `^17.2.4`
- express: `^5.2.1`
- google-auth-library: `^10.5.0`
- jsonwebtoken: `^9.0.3`
- mongoose: `^9.2.0`
- razorpay: `^2.9.6`

7. Important Environment Variables
Backend (`backend/.env`)
- `OPENWEATHER_API_KEY`
- `MONGODB_URI`
- `JWT_SECRET`
- `RAZORPAY_KEY_ID`
- `RAZORPAY_KEY_SECRET`
- `GOOGLE_CLIENT_ID`
- `EXCHANGE_RATE_API_KEY`

Frontend (`frontend/.env`)
- `VITE_API_BASE_URL`
- `VITE_RAZORPAY_KEY_ID`
- `VITE_GOOGLE_CLIENT_ID`

8. Key Terminal Commands Used During Development
Project setup
```bash
npm init -y
npm install express dotenv axios cors mongoose razorpay jsonwebtoken bcryptjs go
ogle-auth-library
```

Run backend
```bash
cd backend
npx nodemon server.js
```

Run frontend
```bash
cd frontend
npm install
npm run dev
```

Build frontend
```bash
npm run build
npm run preview
```

Tailwind setup
```bash
npm install -D tailwindcss@3.4.17 postcss autoprefixer
npx tailwindcss init -p
```

Git/GitHub
```bash
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin <repo-url>
git push -u origin main
```

Remove dist from repo (if needed)
```bash
git rm -r --cached frontend/dist
echo "frontend/dist" >> .gitignore
git add .gitignore
git commit -m "Remove dist from repository and ignore it"
git push
```

9. Deployment Summary
Frontend Deployment (Netlify)
- Site URL: `https://cloudlineweather.netlify.app/`
- Build command: `npm run build`
- Publish directory: `dist`
- Root directory: `frontend`

Backend Deployment (Render)
- Start command: `node server.js` (or `npm start` if script provided)
- Build command: `npm install`
- Root directory: `backend`

Deployment checklist
1. Deploy backend first
2. Set backend env vars in Render
3. Deploy frontend with Netlify env vars
4. Update backend CORS allowlist for Netlify domain

5. Configure Google OAuth authorized origin
6. Configure Razorpay production/test keys and allowed domains
7. Verify full production flow

10. Common Issues Faced and Resolutions
- OpenWeather `401 invalid key` -> waited for key activation, corrected endpoint usage
- One Call 3.0 restriction errors -> used supported free endpoints where needed
- Razorpay `key_id mandatory` -> missing env vars fixed
- `cityName.trim is not a function` -> corrected event/string handling
- Duplicate city across users -> migrated to user-scoped favorites index
- Render `ERR_MODULE_NOT_FOUND` -> filename import case mismatch fixed
- Atlas connection/auth errors -> IP allowlist + DB user/password + URI format corrected
- `Tooltip already declared` -> aliased Leaflet Tooltip import

11. Final Architecture (High-level)
- Frontend (React/Vite) consumes backend REST APIs
- Backend (Express) integrates OpenWeather, ExchangeRate-API, Razorpay
- MongoDB Atlas stores users and favorites
- JWT secures user-scoped endpoints
- Premium activation state stored per-user in frontend local storage and UI logic

12. Suggested Next Improvements
- Move premium activation persistence fully to backend DB (single source of truth)
- Add refresh token/session strategy
- Add retry and caching layer for external weather APIs
- Add test coverage (unit + integration)
- Add CI/CD pipeline for lint/build/deploy checks

Prepared for: Cloudline Weather project