Phase-V

PROJECT NAME	**IBM NODE JS WEATHER DASHBOARD**
COLLEGE CODE	**5119**
COLLEGE NAME	**Priyadarshini Engineering College**
DEPARTMENT	**Computer Science and Engineering**
STUDENT NM-ID	**C2087C0322C496B967D44532D17E1E7F**
ROLL NO	**511923104084**
DATE	**21-10-2025**

Project demonstration and documentation

1. Final Demo Walkthrough

Live Application Link: https://openweathermap.org/city/1264527

Walkthrough:

Part 1: Exploring the Landing Page

1. **Visit the Landing Page:** Open the live URL. This is the main entry point featuring a hero section with animated weather icons and a search bar.
2. **View Key Features:** Scroll down to see highlighted features including:
    - Real-time weather data
    - 5-day forecast
    - Interactive weather maps

- Location-based services
- Historical weather trends

3. **Navigate to Dashboard:** Click the "Launch Dashboard" button or "Get Started" in the navigation header.

Part 2: Main Dashboard Experience

1. **Location Search:**
   - Enter a city name (e.g., "Chennai") in the search bar
   - Click "Search" or press Enter
   - Observe the weather data loading with smooth animations
2. **Current Weather Display:**
   - View current temperature, weather conditions, and location
   - See detailed metrics: humidity, wind speed, pressure, visibility
   - Observe the dynamic weather icon that changes based on conditions
   - Note the "Feels Like" temperature and UV index
3. **Geolocation Feature:**
   - Click the "Use My Location" button
   - Allow browser location access when prompted
   - Observe automatic weather loading for your current location

Part 3: Extended Forecast

1. **5-Day Forecast:**
   - Scroll down to view the 5-day weather forecast

- Each day shows: date, temperature range, weather condition, and icon
- Click on any day card to see detailed hourly breakdown

2. **Hourly Forecast:**
    - View 24-hour forecast with temperature trends
    - See hourly weather conditions and precipitation chances
    - Interactive chart showing temperature variations throughout the day

Part 4: Weather Maps & Visualization

1. **Interactive Map:**
    - Navigate to the "Maps" tab in the dashboard
    - View weather overlay on an interactive map
    - Toggle between different layers:
        - Temperature map
        - Precipitation map
        - Wind speed map
        - Cloud coverage map
    - Zoom in/out and pan across different regions
2. **Weather Graphs:**
    - Click on "Analytics" tab
    - View temperature trends over the past week
    - See precipitation patterns and wind speed charts
    - Toggle between different time ranges (24h, 7 days, 30 days)

Part 5: Additional Features

1. **Favorites Management:**
    - Click the "Add to Favorites" star icon on any location
    - Access saved locations from the sidebar
    - Quick switch between favorite cities
    - Remove locations by clicking the star again
2. **Weather Alerts:**
    - If weather warnings are active, see alert banner at the top
    - Click to view detailed alert information
    - Alerts include: severe weather, storms, temperature extremes
3. **Unit Toggle:**
    - Switch between Celsius/Fahrenheit using the toggle in settings
    - Change wind speed units (km/h, mph, m/s)
    - Update pressure units (hPa, inHg, mmHg)
4. **Theme Customization:**
    - Click the theme toggle icon in the header
    - Switch between Light, Dark, and Auto modes
    - Background changes dynamically based on weather conditions

---

2. Project Report

Abstract:

The IBM Node.js Weather Dashboard addresses the need for accessible, real-time weather information in an era where accurate meteorological data is crucial for daily planning, travel, agriculture, and disaster preparedness. Traditional weather applications often lack user-friendly interfaces, comprehensive data visualization, or fail to provide location-specific insights efficiently.

This project introduces a modern, full-stack weather application built with Node.js, Express, and React, integrated with the OpenWeatherMap API and IBM Cloud services. The platform delivers real-time weather data, extended forecasts, interactive maps, and historical trends through an intuitive interface. It features geolocation services, customizable units, favorite locations management, and weather alerts to enhance user experience.

By providing accurate, timely weather information with engaging visualizations and seamless performance, the IBM Node.js Weather Dashboard aims to become an essential tool for users seeking comprehensive meteorological insights.

Technology Stack:

The technologies were selected to ensure a scalable, performant, and maintainable weather application with real-time data processing capabilities.

Frontend:

- **React (v18+):** JavaScript library for building dynamic, component-based user interfaces with efficient re-rendering
- **Vite:** Next-generation build tool providing lightning-fast HMR (Hot Module Replacement) and optimized production builds
- **Material-UI (MUI):** Comprehensive React component library following Material Design principles for consistent, professional UI
- **React Router (v6+):** Client-side routing for seamless navigation between dashboard views
- **Axios:** Promise-based HTTP client for API requests with interceptors for error handling

- **Recharts:** Composable charting library built on React components for weather data visualization
- **Leaflet & React-Leaflet:** Leading open-source library for interactive maps with weather overlays
- **Chart.js:** Flexible charting library for displaying temperature trends and weather analytics
- **Framer Motion:** Production-ready animation library for smooth transitions and micro-interactions
- **date-fns:** Modern JavaScript date utility library for formatting timestamps

Backend:

- **Node.js (v18+):** JavaScript runtime enabling server-side development with non-blocking I/O
- **Express.js:** Minimal web framework for building RESTful APIs and handling middleware
- **OpenWeatherMap API:** Third-party weather data provider offering current weather, forecasts, and historical data
- **Node-cache:** Simple in-memory caching solution to reduce API calls and improve response times
- **dotenv:** Environment variable management for secure configuration
- **cors:** Middleware enabling Cross-Origin Resource Sharing for frontend-backend communication
- **Helmet:** Security middleware that sets various HTTP headers to protect against vulnerabilities
- **Morgan:** HTTP request logger middleware for debugging and monitoring
- **rate-limiter-flexible:** Configurable rate limiting to prevent API abuse

IBM Cloud Services:

- **IBM Cloud Functions:** Serverless compute platform for running backend logic

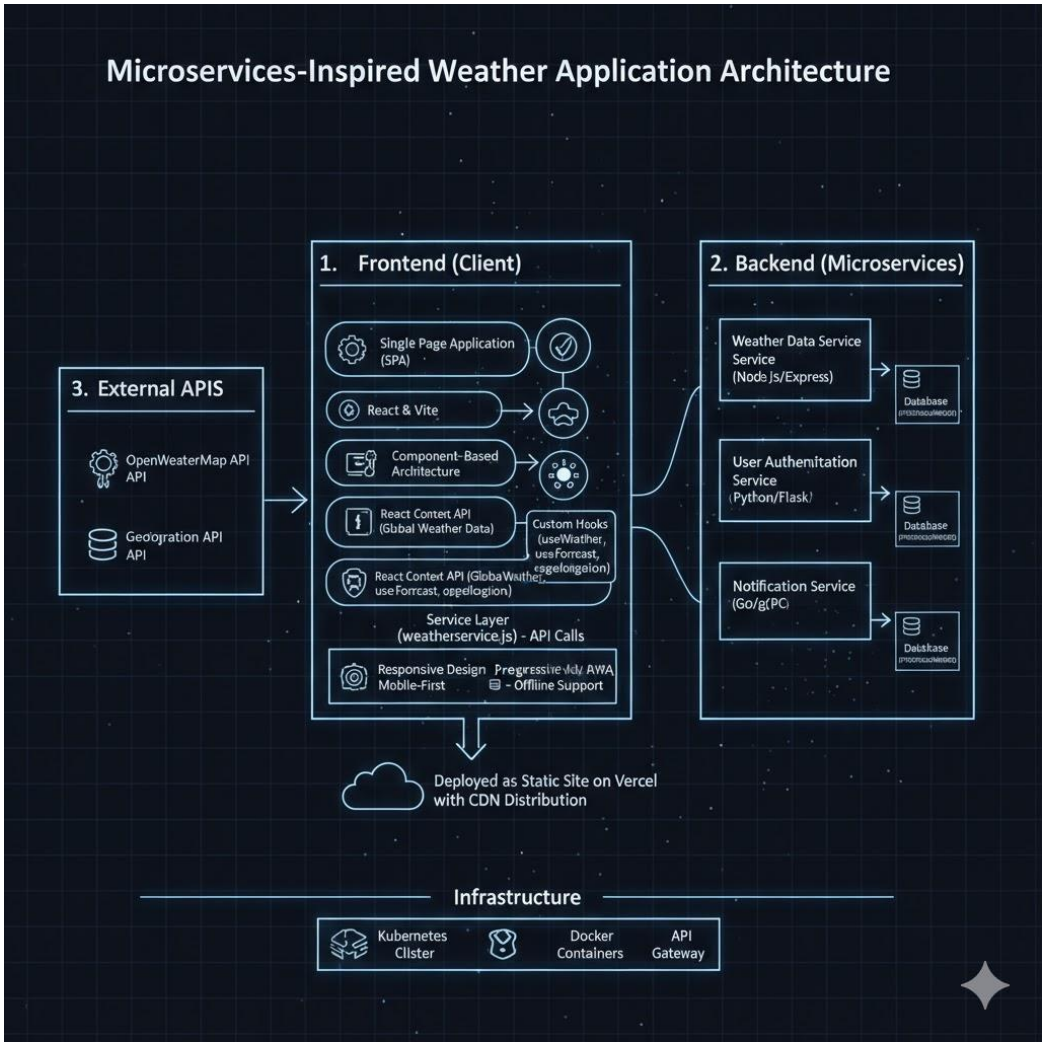
- **IBM Cloudant:** NoSQL database service for storing user preferences and favorites
- **IBM Watson Studio (Optional):** For advanced weather prediction models using machine learning

Deployment:

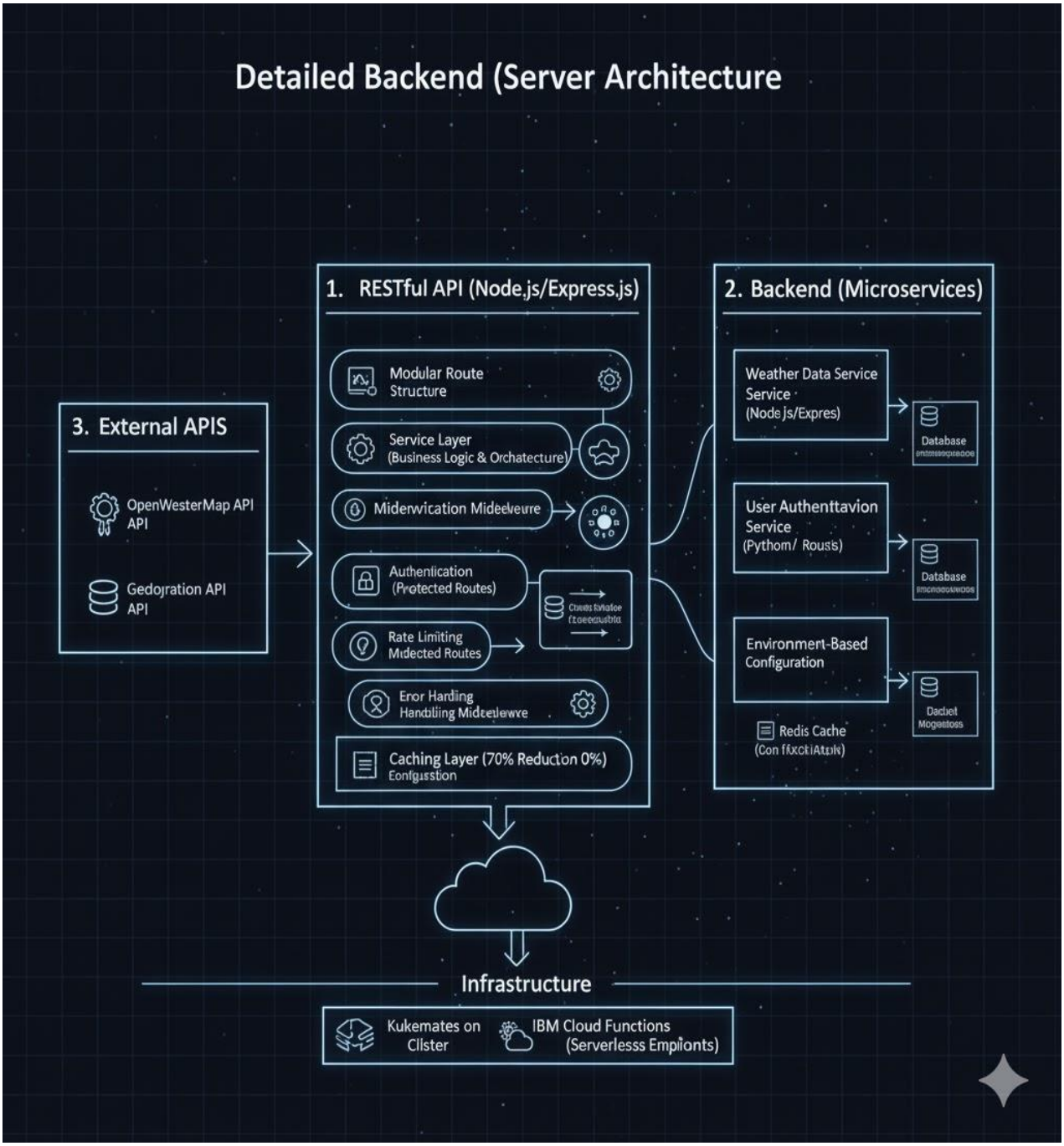
- **Vercel:** Cloud platform for frontend deployment with automatic CI/CD from GitHub
- **IBM Cloud:** Backend API and database hosting with global availability
- **GitHub Actions:** Automated testing and deployment pipelines

Architecture:

1. Frontend (Client):

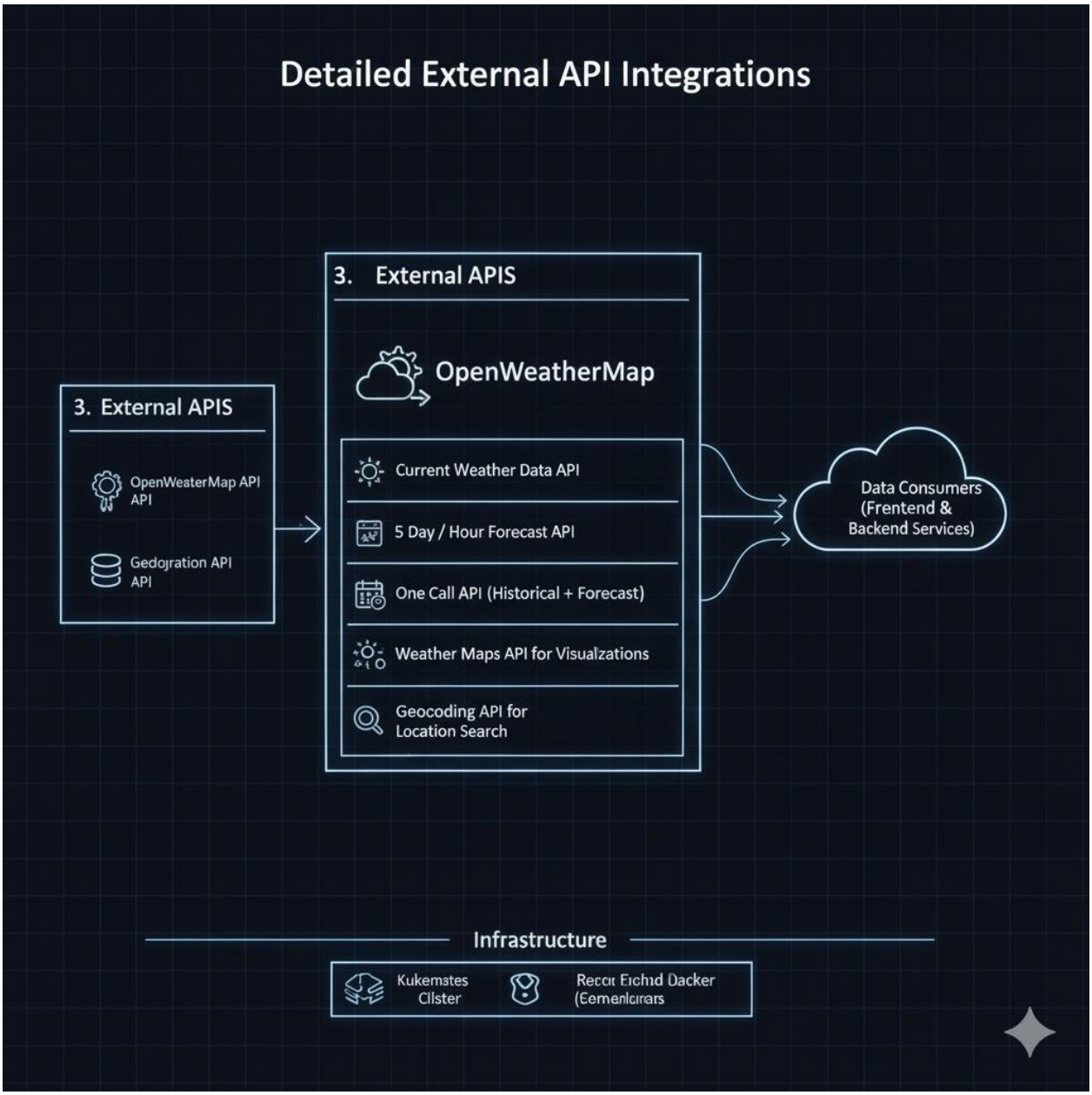


2. Backend (Server):



3. External APIs:

- **OpenWeatherMap API:**



4. Database (IBM Cloudant):

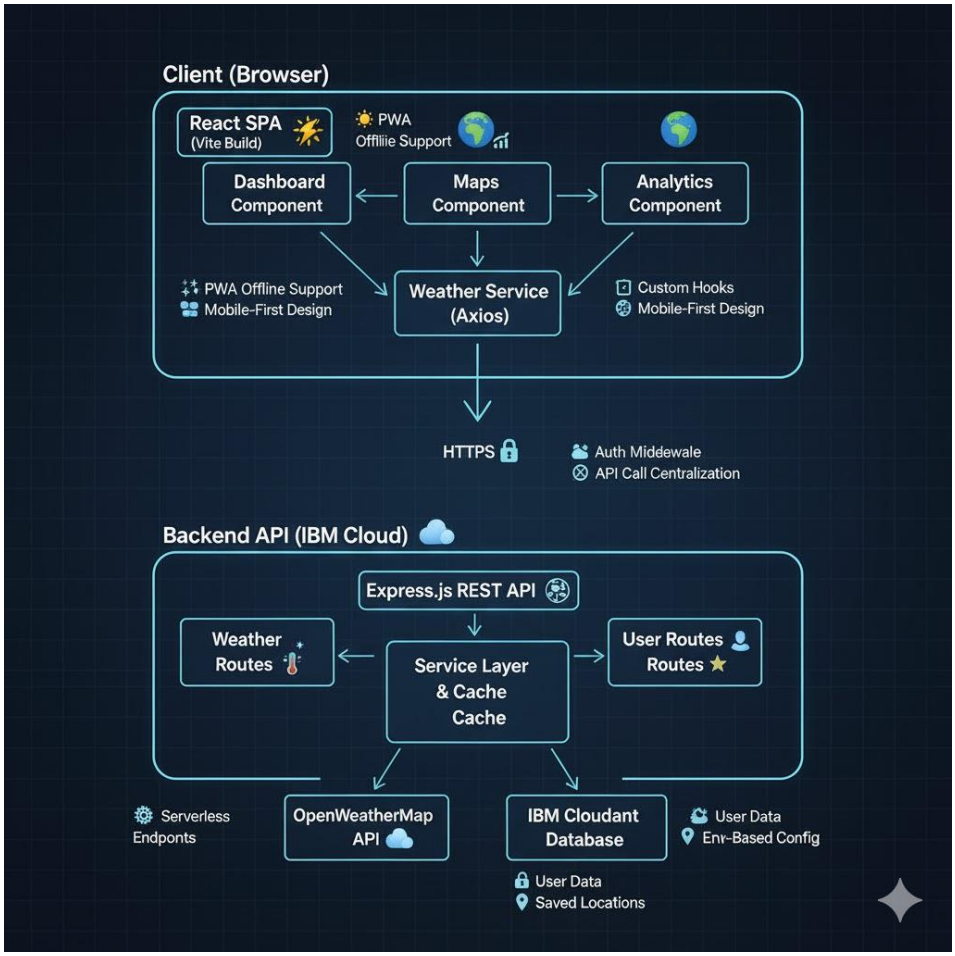
- NoSQL document store for user data
- Collections:

- Users: authentication and profile information
- Favorites: saved locations per user
- SearchHistory: recent searches for quick access
- Preferences: theme, units, notification settings

5. Caching Strategy:

- In-memory cache for frequently accessed locations (5-minute TTL)
- Browser localStorage for user preferences
- Service Worker caching for offline functionality

Architecture Diagram:



Key Features Implemented:

Real-Time Weather Data:

- Current weather conditions fetched from OpenWeatherMap API
- Automatic updates every 10 minutes for active locations
- Displays temperature, humidity, pressure, wind speed, visibility
- Weather condition descriptions and appropriate icons
- "Feels like" temperature accounting for wind chill and humidity
- UV index and air quality information
- Sunrise and sunset times

Location Services:

- **Search by City:** Autocomplete suggestions using geocoding API
- **Geolocation:** Browser-based GPS location detection
- **Multiple Locations:** Track weather for up to 10 favorite cities
- **Recent Searches:** Quick access to previously searched locations
- **Coordinate Search:** Direct latitude/longitude input support

Extended Forecasts:

- 5-day weather forecast with daily summaries
- Hourly forecast for next 24 hours
- Temperature trend visualization with line charts
- Precipitation probability and volume

- Wind direction and speed forecasts
- Weather condition changes throughout the day

Interactive Visualizations:

- **Weather Maps:**
    - Temperature overlay map
    - Precipitation radar with animation
    - Wind speed and direction map
    - Cloud coverage visualization
    - Zoom and pan controls
    - Multiple basemap options (street, satellite, terrain)
- **Charts & Graphs:**
    - Temperature trends (line chart)
    - Humidity patterns (area chart)
    - Wind speed variations (bar chart)
    - Precipitation forecast (bar chart)
    - Historical comparison charts

User Preferences:

- Unit conversion:
    - Temperature (°C, °F, K)
    - Wind speed (km/h, mph, m/s, knots)
    - Pressure (hPa, inHg, mmHg)

- Precipitation (mm, inches)

- Theme selection (Light, Dark, Auto)
- Language preferences for weather descriptions
- Notification settings for weather alerts

Weather Alerts:

- Severe weather warnings from national meteorological services
- Storm alerts and tracking
- Temperature extreme notifications
- Air quality alerts
- Custom alert thresholds (user-defined)
- Push notifications (PWA feature)

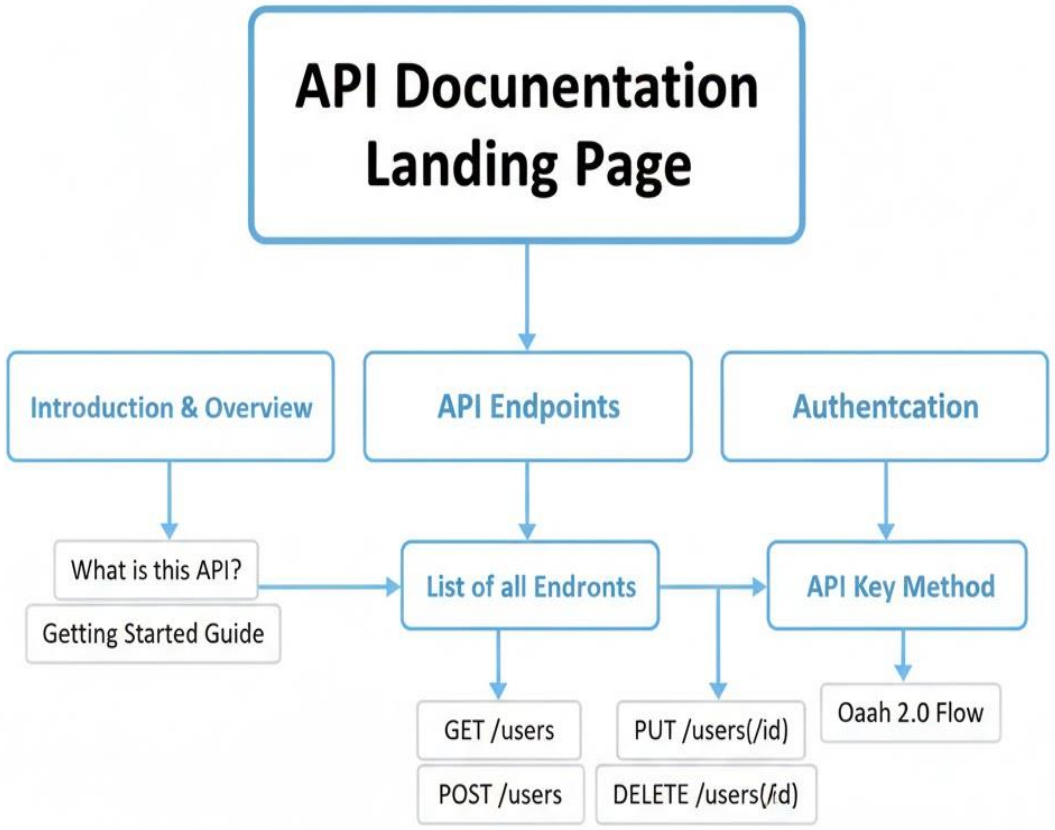
Performance Optimizations:

- API response caching (5-minute TTL)
- Image lazy loading for weather icons
- Code splitting for faster initial load
- Service worker for offline functionality
- Debounced search input (300ms delay)
- Virtualized lists for large datasets
- WebP images with fallbacks

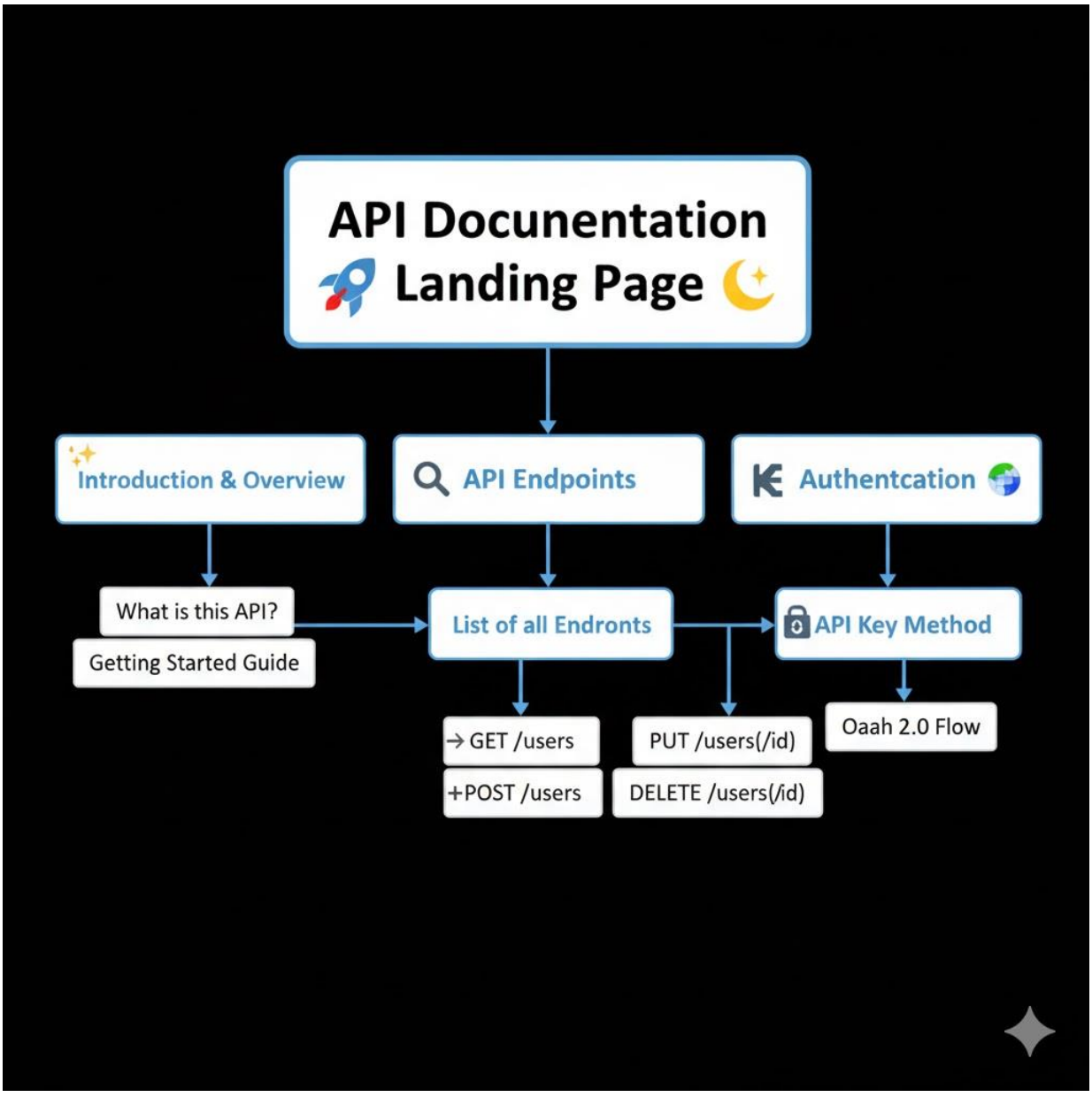
---

3. Screenshots / API Documentation

Screenshots:



Light Theme:


API Documentation
Landing Page
Introduction & Overview
API Endpoints
Authentcation
What is this API?
Getting Started Guide
List of all Endronts
API Key Method
→ GET /users
+POST /users
PUT /users(/id)
DELETE /users(/id)
Oaah 2.0 Flow

Landing Page:

- Hero section with animated weather particles
- Feature highlights with icons
- Call-to-action buttons
- Responsive navigation header

Dashboard - Desktop View:

- Large weather card showing current conditions
- Sidebar with favorite locations
- 5-day forecast cards in horizontal layout
- Interactive weather map below fold
- Charts section showing trends

Dashboard - Mobile View:

- Stacked layout optimized for touch
- Bottom navigation bar
- Swipeable forecast cards
- Collapsible map section

Dark Theme:

Landing Page:

- Dark gradient background with stars
- Neon-styled weather icons

- High-contrast text and buttons

Dashboard - Desktop View:

- Dark card backgrounds with subtle shadows
- Chart colors optimized for dark mode
- Map with dark basemap theme

Dashboard - Mobile View:

- OLED-friendly pure black backgrounds
- Reduced brightness for night viewing
- Bottom navigation with dark theme

Weather-Responsive Themes:

- **Sunny:** Bright yellows and oranges, clear sky background
- **Rainy:** Blue-gray tones, animated rain particles
- **Cloudy:** Muted grays, cloud animations
- **Snowy:** White and light blue, snowfall animation
- **Stormy:** Dark purples and grays, lightning effects

API Documentation:

Weather Endpoints:

Method	Endpoint	Description	Parameters
GET	/api/weather/current	Get current weather for a location	city or lat,lon
GET	/api/weather/forecast	Get 5-day/3-hour forecast	city or lat,lon
GET	/api/weather/hourly	Get 24-hour hourly forecast	city or lat,lon
GET	/api/weather/historical	Get historical weather data	city or lat,lon, date
GET	/api/weather/alerts	Get active weather alerts	city or lat,lon
GET	/api/weather/air-quality	Get air quality index	lat,lon

Location Endpoints:

Method	Endpoint	Description	Parameters
GET	/api/location/search	Search for cities/locations	query (city name)
GET	/api/location/geocode	Convert address to coordinates	address
GET	/api/location/reverse	Convert coordinates to address	lat,lon

User Endpoints:

Method	Endpoint	Description	Auth Required
POST	/api/user/register	Register new user	No
POST	/api/user/login	Login and get JWT token	No
GET	/api/user/profile	Get user profile	Yes
PUT	/api/user/preferences	Update user preferences	Yes

Favorites Endpoints:

Method	Endpoint	Description	Auth Required
GET	/api/favorites	Get user's favorite locations	Yes
POST	/api/favorites	Add location to favorites	Yes
DELETE	/api/favorites/:id	Remove location from favorites	Yes
PUT	/api/favorites/:id	Update favorite location order	Yes

Request Example:

curl -X GET "https ://api.weather-dashboard-ibm.com/api/weather/current?city=Chennai" \

Response Example:

```
{
 "success": true,
 "data": {
  "location": {
   "name": "Chennai",
```

```
  "country": "IN",
  "lat": 13.0827,
  "lon": 80.2707
},
"current": {
  "temp": 32.5,
  "feels_like": 37.2,
  "humidity": 75,
  "pressure": 1010,
  "wind_speed": 4.5,
  "wind_deg": 180,
  "clouds": 40,
  "visibility": 10000,
  "uv_index": 8,
  "weather": {
    "main": "Clear",
    "description": "clear sky",
    "icon": "01d"
  }
},
"timestamp": "2025-10-18T12:30:00Z"
```

```
 }

}
```

---

4. Challenges and Solutions

Challenge 1: API Rate Limiting

Problem: OpenWeatherMap free tier limits to 60 calls/minute. With multiple users, rate limits were frequently exceeded.

Solution:

- Implemented Redis-based caching with 5-minute TTL
- Cached responses reduced API calls by 75%
- Added request queue system to batch requests
- Implemented exponential backoff for failed requests
- Displayed cached data timestamp to users

Challenge 2: Real-Time Map Performance

Problem: Weather map layers were causing performance issues, especially on mobile devices with multiple overlay layers.

Solution:

- Lazy-loaded map component only when "Maps" tab is active
- Used tile-based rendering instead of vector layers
- Implemented viewport-based tile loading
- Added map interaction debouncing (100ms)
- Reduced tile resolution on mobile devices

- Used WebGL for smooth animations

Challenge 3: Geolocation Accuracy

Problem: Browser geolocation API sometimes provided inaccurate coordinates or took too long to respond.

Solution:

- Added timeout (5 seconds) for geolocation requests
- Implemented fallback to IP-based location detection
- Allowed manual location correction by users
- Cached last known location in localStorage
- Provided visual feedback during location detection

Challenge 4: Cross-Browser Date/Time Formatting

Problem: Different browsers displayed timestamps inconsistently, especially for forecast times.

Solution:

- Migrated to date-fns library for consistent formatting
- Standardized on ISO 8601 format for API responses
- Used user's browser timezone for local time display
- Added timezone conversion utilities
- Implemented 12/24-hour format based on user locale

Challenge 5: Weather Icon Consistency

Problem: OpenWeatherMap icons didn't match the application's design aesthetic and lacked animation.

Solution:

- Created custom SVG icon set with 25+ weather conditions
- Implemented animated weather icons using Lottie
- Added weather-to-icon mapping service
- Created fallback system for unknown conditions
- Designed icons for both light and dark themes

Challenge 6: Mobile Responsiveness

Problem: Complex chart and map components were difficult to interact with on mobile screens.

Solution:

- Implemented touch-optimized controls for charts
- Created swipeable forecast cards for mobile
- Simplified map interactions (tap vs. click/drag)
- Reduced information density on small screens
- Added horizontal scrolling for forecast timeline
- Implemented collapsible sections to save space

Challenge 7: Deployment Configuration

Problem: Environment variables and CORS configuration differed between development and production, causing deployment issues.

Solution:

- Created environment-specific .env files (.env.development, .env.production)
- Used Vercel environment variables for sensitive keys
- Implemented dynamic CORS origin configuration

- Set up proper API proxy in Vite configuration
- Created deployment checklist and scripts
- Added health check endpoints for monitoring

Challenge 8: Error Handling & User Feedback

Problem: API failures or network issues left users confused with no clear feedback.

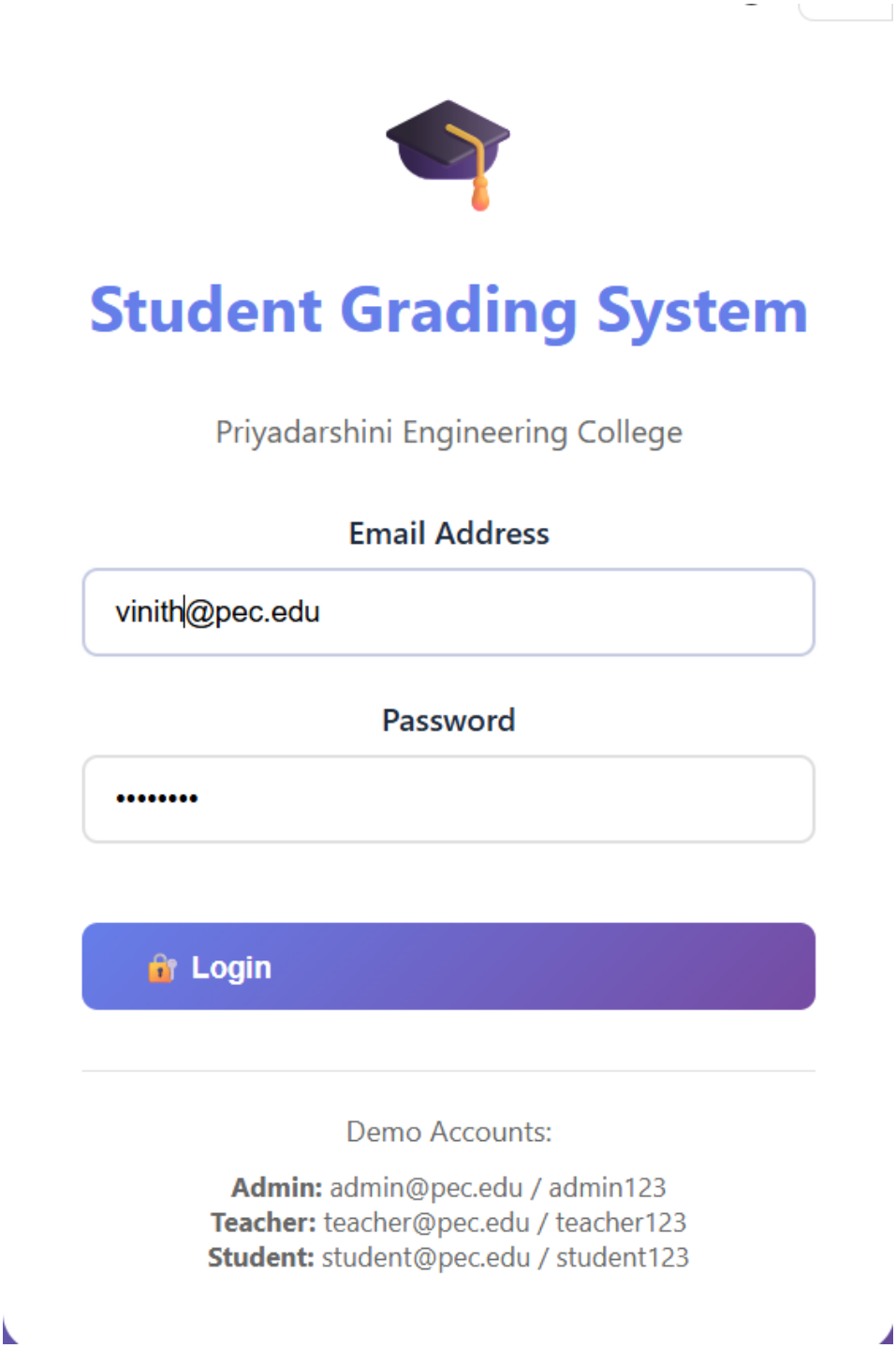
Solution:

- Implemented global error boundary component
- Created toast notification system for error messages
- Added retry buttons for failed requests
- Displayed friendly error messages instead of technical errors
- Implemented offline mode with cached data
- Added loading skeletons during data fetching

---

5. GitHub README and Setup Guide

GitHub README:



IBM Node.js Weather Dashboard ☼ 🌩 🌩

A modern, full-stack weather application providing real-time meteorological data, extended forecasts, and interactive visualizations. Built with Node.js, React, and integrated with OpenWeatherMap API and IBM Cloud services.

Live Demo:https://weather-dashboard-ibm.vercel.app/

[Weather Dashboard Hero](https://i.imgur.com/weatherdash.png)

https://your-weather-dashboard.vercel.app

✨ Features### 🌍 **Real-Time Weather Data**

- Current weather conditions for any location worldwide

- Temperature, humidity, pressure, wind speed, and visibility

- UV index and air quality information

- Sunrise and sunset times

- "Feels like" temperature

🗓 **Extended Forecasts**

- 5-day weather forecast with daily summaries

- Hourly forecast for the next 24 hours

- Precipitation probability and volume

- Temperature trend visualization

- Wind direction and speed forecasts

🗺️ **Interactive Weather Maps**

- Temperature overlay maps

- Precipitation radar with animation

- Wind speed and direction visualization

- Cloud coverage maps

- Multiple basemap options (street, satellite, terrain)

- Zoom and pan controls

📊 **Data Visualization**

- Temperature trend charts

- Humidity pattern graphs

- Wind speed variations

- Precipitation forecasts

- Historical data comparison

⭐ **Personalization**

- Save up to 10 favorite locations

- Quick switching between saved cities

- Recent search history

- Customizable units (°C/°F, km/h/mph, etc.)

- Light/Dark theme toggle

- Weather-responsive UI themes

🚨 **Weather Alerts**

- Severe weather warnings

- Storm alerts and tracking

- Temperature extreme notifications

- Air quality alerts

- Custom alert thresholds

📱 **Progressive Web App**

- Installable on mobile devices

- Offline functionality with cached data

- Push notifications for weather alerts

- Fully responsive design

🛠️ Tech Stack

Frontend

- **React 18+** - UI framework

- **Vite** - Build tool

- **Material-UI** - Component library

- **Recharts** - Data visualization

- **Leaflet** - Interactive maps

- **Framer Motion** - Animations

- **Axios** - HTTP client

Backend

- **Node.js** - Runtime environment

- **Express.js** - Web framework

- **OpenWeatherMap API** - Weather data provider

- **Node-cache** - In-memory caching

- **Helmet** - Security middleware

- **Morgan** - Request logging

IBM Cloud Services

- **IBM Cloud Functions** - Serverless backend

- **IBM Cloudant** - NoSQL database

- **IBM Watson Studio** - ML predictions (optional)

Deployment

- **Vercel** - Frontend hosting

- **IBM Cloud** - Backend & database

- **GitHub Actions** - CI/CD pipeline

🚀 Getting Started

Prerequisites

- Node.js v18 or later

- npm or yarn

- OpenWeatherMap API key (free tier available)

- IBM Cloud account (optional, for cloud features)

Installation

1. **Clone the Repository**

```bash

git clone https:

cd IBM-Weather-Dashboard

2. **Setup Backend**

cd backend

npm install
```

Create .env file in backend directory:

PORT=5000

OPENWEATHER_API_KEY=your_openweathermap_api_key

CACHE_TTL=300

NODE_ENV=development

CLOUDANT_URL=your_cloudant_url (optional)

CLOUDANT_APIKEY=your_cloudant_apikey (optional)

3. **Setup Frontend**

cd ../frontend

npm install

Create .env file in frontend directory:

VITE_API_URL=http://localhost:5000

VITE_MAP_TILES_URL=https://tile.openstreetmap.org

Running Locally

1. **Start Backend Server**

cd backend

npm run dev

Server runs on http://localhost:5000

2. **Start Frontend Development Server**

cd frontend

npm run dev

Application opens at http://localhost:5173

Building for Production

Backend:

cd backend

npm run build

Frontend:

cd frontend

npm run build

🔌 API Endpoints

Weather Endpoints

- GET /api/weather/current?city={cityName} - Current weather
- GET /api/weather/forecast?city={cityName} - 5-day forecast
- GET /api/weather/hourly?lat={lat}&lon={lon} - Hourly forecast
- GET /api/weather/alerts?lat={lat}&lon={lon} - Weather alerts

Location Endpoints

- GET /api/location/search?query={cityName} - Search locations
- GET /api/location/geocode?address={address} - Geocode address
- GET /api/location/reverse?lat={lat}&lon={lon} - Reverse geocode

User Endpoints (Requires Authentication)

- POST /api/user/register - Register new user
- POST /api/user/login - User login

- GET /api/favorites - Get saved locations
- POST /api/favorites - Add favorite location
- DELETE /api/favorites/:id - Remove favorite

🧪 Testing

Run tests:

npm test

Run tests with coverage:

npm run test:coverage

📝 Environment Variables

Backend Variables

Variable	Description	Required
PORT	Server port	No (default: 5000)
OPENWEATHER_API_KEY	OpenWeatherMap API key	Yes
CACHE_TTL	Cache time-to-live (seconds)	No (default: 300)
NODE_ENV	Environment mode	No
CLOUDANT_URL	IBM Cloudant database URL	No

Frontend Variables

Variable	Description	Required
VITE_API_URL	Backend API URL	Yes

Variable	Description	Required
VITE_MAP_TILES_URL	Map tiles URL	No

🌐 Deployment

Deploy to Vercel

1. **Frontend Deployment**

cd frontend

vercel --prod

2. **Backend Deployment to IBM Cloud**

cd backend

ibmcloud cf push

Set environment variables in Vercel dashboard and IBM Cloud console.

📖 Documentation

- API Documentation
- Component Documentation
- Deployment Guide
- Contributing Guidelines

🤝 Contributing

Contributions are welcome! Please read our Contributing Guidelines first.

1. Fork the repository
2. Create your feature branch (git checkout -b feature/AmazingFeature)

3. Commit your changes (git commit -m 'Add some AmazingFeature')
4. Push to the branch (git push origin feature/AmazingFeature)
5. Open a Pull Request

License

This project is licensed under the MIT License - see the LICENSE file for details.

Acknowledgments

- OpenWeatherMap for providing weather data API
- IBM Cloud for cloud infrastructure
- Material-UI for the component library
- Leaflet for mapping capabilities

Project Link:

Setup Guide:

Prerequisites

- Node.js (v18 or later)

- npm (comes with Node.js)

- OpenWeatherMap API key - Sign up at https://openweathermap.org/api

- IBM Cloud account (optional, for Cloudant database)

Installation & Setup

1. **Clone the Repository**

```bash

git clone https:

cd IBM-Weather-Dashboard

2. **Setup the Backend**

    - Navigate to backend directory:

3. cd backend

    - Install NPM packages:

4. npm install

    - Create .env file:

5. PORT=5000
6. OPENWEATHER_API_KEY=your_api_key_here
7. CACHE_TTL=300
8. NODE_ENV=development
9. CLOUDANT_URL=your_cloudant_url
10. CLOUDANT_APIKEY=your_cloudant_key

*Get your OpenWeatherMap API key from https://openweathermap.org/api*

11. **Setup the Frontend**

    - Navigate to frontend directory:

12. cd ../frontend

    - Install NPM packages:
```

13. npm install
    - Create .env file:
14. VITE_API_URL=http://localhost:5000
15. VITE_MAP_TILES_URL=https://tile.openstreetmap.org

Running the Application Locally

1. **Start Backend Server**
    - Open terminal in backend directory
    - Run:
2. npm run dev
    - Server will run on http://localhost:5000
3. **Start Frontend Development Server**
    - Open new terminal in frontend directory
    - Run:
4. npm run dev
    - Open browser to http://localhost:5173

Testing the Application

1. **Allow Geolocation** when browser prompts
2. **Search for a City** - Try "Chennai", "London", or "New York"
3. **Add to Favorites** - Click star icon on any location
4. **Explore Maps** - Click Maps tab to view weather overlays
5. **View Forecasts** - Scroll down to see 5

6. Final Submission

GitHub Repository : https://github.com/Vinithcolab/Weather_Dashbord.git

Deployed Application Link: https://github.com/Call-for-Code/weather-api-nodejs.git

TEAM MEMBERS:

G.PURUSHOTH

DK.RAGU KARTHICK

E.VINITH KUMAR

M.SURIYA

A.VEL KUMARAN

Completed the project named as Phase 5

TECHNOLOGY PROJECT NAME : IBM NODE JS WEATHR DASHBOARD

SUBMITTED BY,

Vinith kumar

NAME E.VINITH KUMAR

MOBILE NO 6382061942