raindrop

Technical Documentation

Jakub Jagodziński

Version 1.0
July 7, 2025

Contents

1 Introduction

1.1 Project Objective

Raindrop is a full-stack web application that allows users to search, view, and store 5-day weather forecasts based on city input. It integrates external APIs to provide detailed hourly predictions, and enables users to save and manage weather history for later reference.

The primary goals of the system are to:

- Allow searching for cities with autocomplete support,
- Fetch and display 5-day hourly weather forecasts,
- Persist selected forecasts in a local database for future viewing,
- Provide smooth and responsive user experience using modern frontend and backend frameworks.

1.2 Scope of the Application

The application is focused on the following functionalities:

- Fetching city data using OpenWeather Geocoding API,
- Pulling hourly forecast data using OpenWeatherMap API,
- Storing user-selected forecasts in MongoDB via backend API,
- Providing RESTful endpoints via FastAPI for managing forecast history,
- Offering a rich frontend interface built with React and Tailwind CSS.

2 System Overview

2.1 Architecture

Raindrop follows a classic client-server architecture with clearly separated frontend and backend components. Communication between layers is handled via RESTful HTTP API.

- **Frontend:** built with React (Vite) and styled with Tailwind CSS. Handles user interactions and displays data fetched from backend and external APIs.
- **Backend:** written in Python using FastAPI framework. Acts as a bridge between the frontend, MongoDB database, and third-party APIs.
- **Database:** MongoDB is used for storing saved forecasts and historical user interactions.

Docker and Docker Compose are used to containerize and orchestrate the entire system, making local deployment and development simple and reproducible.

2.2 Third-Party Integrations

The application relies on two external APIs:

- **OpenWeatherMap API:** used to fetch hourly weather data.
- **REST Countries API:** used to supplement geographic data such as country flags or codes.

3 Application Features

3.1 Forecast Search

Users can search for cities using a responsive input field that supports autocomplete functionality based on OpenWeather Geocoding API. Once a city is selected, the system retrieves a 5-day forecast with 3-hour intervals.

3.2 Hourly View

Weather forecasts are displayed in hourly resolution with visual grouping by day. Users can scroll through different days and view temperature, weather condition, and other metadata.

3.3 Saving Forecasts

Each forecast can be saved to the user's local history with a single click. Saved forecasts are stored in MongoDB and displayed on a separate screen for future review.

3.4 History Management

Users can:

- View a list of previously saved forecasts,
- Navigate between days within each saved forecast,
- Remove specific entries from the database.

All of these operations are handled through RESTful API endpoints provided by the backend.

4 Technology Stack

4.1 Frontend

- React (with Vite)
- Tailwind CSS

4.2 Backend

- FastAPI (Python 3.10+)

4.3 Database

- MongoDB

4.4 DevOps and Deployment

- Docker
- Docker Compose

4.5 External APIs

- OpenWeatherMap API
- REST Countries API

5 Running the Application

5.1 Configuration

Before running the application, ensure that you have created the following environment configuration files:

- `.env` in the root directory
- `.env` in the backend directory

Base templates for these files are provided as `.env.example`.

5.2 Launching with Docker Compose

To start all services (frontend, backend, and MongoDB), run the following command:

```
docker compose up
```

Once running, the services will be available at:

- Frontend: `http://localhost:3000`
- Backend API (Swagger UI): `http://localhost:8000/docs`

6 License and Author

6.1 License

This project is distributed under the MIT License.

6.2 Author

Jakub Jagodziński