Cloud Development Microservice Application Design Document

Project Name: Holiday Query and Data Storage Service (Holiday Cache Service)
Group: r

1. Application Purpose and External Data Source Usage

1.1 Application Purpose:
This application adopts a three-part collaborative architecture to provide an efficient Chinese public holiday query service1. The core function involves the Logic Processing Component calling an External Data Source (Third-Party API) and utilising the Data Storage Component to cache holiday data, thus speeding up user queries.
1.2 External Data Source (Third-Party API):

- **Name:** Nager.Date Public Holidays API
- **Purpose:** The Logic Processing Component calls this interface (/api/v3/PublicHolidays/{year}/{countryCode}) to retrieve public holiday data for a specified year and country.
- **Data Processing:** During the data processing stage, the Logic Processing Component is responsible for extracting date , countryCode, month, localName, name (English name), and cache_time (storage time) from the raw data returned by the interface, to suit the Data Storage Component's storage and the user interface display.

2. Data Storage Component Design (Database Service Design)

- **Responsibility:** Acts as the Data Cache Area for the Logic Processing Component 66, providing only the interface for saving and retrieving data.
- **Implementation:** A small application based on the Flask app(r_db_service.py)

2.1 Database Table (DB Table):

Table Name	Column Name	Data Type	Description	Constraint
public_holidays	date	VARCHAR(10)	Holiday Date (Format e.g. "YYYY/MM/DD")	Not Null, Primary Key (Composite)

	countryCode	VARCHAR(2)	Country Code (e.g. CN, IE)	Not Null, Primary Key (Composite)
	month	VARCHAR(2)	Month of the holiday (e.g. 01, 12)	Not Null
	localName	Text	Chinese/Local Name of the Holiday	Not Null
	name	Text	English Name of the Holiday	Not Null
	cache_time	Date/Time	Cache Time (Used by **Logic Processing Component** to determine expiry)	Not Null

2.2 DB Service Endpoints:

Storage and retrieval functions are divided by country, which facilitates calling and debugging by the Logic Service.

Interface Address (Endpoint)	**Request Method**	**Responsibility**

/db/save-china	POST	**Save** Chinese cache data: Receives a list of Chinese holiday records from the **Logic Processing Component** and writes them to the database for caching.
/db/get-china	GET	**Return** Chinese cache data: Receives an optional year parameter, **retrieves** and returns matching Chinese cache data from the database.
/db/save-ireland	POST	**Save** Irish cache data: Receives a list of Irish holiday records from the **Logic Processing Component** and writes them to the database for caching.
/db/get-ireland	GET	**Return** Irish cache data: Receives an optional year parameter, **retrieves** and returns matching Irish cache data from the database.

3. Logic Processing Component Design (Logic Service Design)

Responsibility: Handles business rules, calls the External Data Source, performs data format conversion, and manages the data storage policy.
Implementation: Based on the Flask app (r_logic_service.py).

3.1 Logic Service Endpoints:

ID	Function	Interface Address (Endpoint)	Request Method	Description (for Gateway Component)

endpoint	General Holiday Data Processing	/logic/holidays	GET	Receives parameters (country, year, month, name) passed by the **Gateway Component**, executes the complete business rules, and returns the filtered results.

The **Logic Service** only needs 1 core endpoint. It uses the country field in the URL or query parameters to decide whether to process the business rules for China or Ireland.

3.2 Logic Service Design Flow (endpoint)

Flow: /logic/holidays (GET)

Step	**Action**	**Implementation Details (within Logic Service)**
1. Receive and Extract	Receives and extracts parameters (country, year, etc.) passed by the **Gateway Service**.	

2. Determine **Data Storage/External Data Source** Target	Determines the **DB Service *endpoint*** and the **External Data Source** URL to be called based on the country parameter.	China (CN): Data Storage-Query = /db/get-china; Data Storage-Save = /db/save-china; External Data Source URL = https://date.nager.at/api/v3/PublicHolidays/2026/CN
		Ireland (IE): Data Storage-Query = /db/get-ireland; Data Storage-Save = /db/save-ireland; External Data Source URL = https://date.nager.at/api/v3/PublicHolidays/2026/IE
3. **Cache** Check	Calls the determined Data Storage-Query interface address.	**Advantage:** The **Logic Service** only needs to call /db/get-china?year=2026, without passing the country code in the URL; the **DB Service** will **automatically process** the country filtering.
4. Data Retrieval and Processing	If the **cache** misses or is expired:	
	a. Call **External Data Source**: The **Logic Service** calls the API URL determined in Step 2 to retrieve the raw data.	
	b. Data Processing: Processes the raw data returned by the API (cleansing, adding country code, calculating	This is the core responsibility of **Data processing** undertaken by the **Logic Service**.

	and extracting the month field, adding storage time).	
5. **Cache** Storage	Calls the determined Data Storage-Save interface address with the processed data.	
6. Final Filtering	Filters the retrieved data (**cached** data or newly fetched data) based on the month and name parameters passed by the **Gateway Service**.	
7. Return Result	Returns the finally filtered data to the **Gateway Service**.	

4. Gateway Service Design

Responsibility: Serves as the user-facing public request Endpoints, receiving user requests and forwarding them to the Logic Service.
Implementation: Based on the Flask app(r_gateway_service.py).

4.1 Gateway Service Endpoints:

Interface Address (Endpoint)	Request Method	Responsibility	Flow Description (User Interface)

endpoint 1 (China Full Year)	GET /api/v1/holidays/CN/2026	Queries all public holidays for China in 2026: Retrieves all public holiday data for China for that year.	The user calls this endpoint. The **Gateway endpoint** calls the **Logic Service's core endpoint** /logic/holidays?country=CN&year=2026 and returns the result to the user.
endpoint 2 (Ireland Full Year)	GET /api/v1/holidays/IE/2026	Queries all public holidays for Ireland in 2026: Retrieves all public holiday data for Ireland for that year.	The user calls this endpoint. The **Gateway endpoint** calls the **endpoint**/logic/holidays?country=IE&year=2026 and returns the result to the user.
endpoint 3 (China Specific Month)	GET /api/v1/holidays/CN/2026/{month}	Queries Chinese holidays for a specific month in 2026: Filters Chinese holidays based on the month parameter ({month}) in the URL.	The user calls this endpoint. The **Gateway endpoint** extracts {month} from the URL, calls the **Logic Service's core endpoint** /logic/holidays?country=CN&year=2026&month={month}, and returns the filtered results.
endpoint 4 (Ireland Specific Month)	GET /api/v1/holidays/IE/2026/{month}	Queries Irish holidays for a specific month in 2026: Filters Irish holidays based on the month	The user calls this endpoint. The **Gateway endpoint** extracts {month} from the URL, calls the **Logic**

		parameter ({month}) in the URL.	**Service's core endpoint** /logic/holidays?country=IE&year=2026&month={month}, and returns the filtered results.
endpoint 5 (China Specific Holiday Time)	GET /api/v1/holidays/CN/2026/name/{name}	Queries the date of a specific Chinese holiday in 2026: Finds the corresponding date based on the holiday name ({name}).	The user calls this endpoint. The **Gateway endpoint** extracts {name} from the URL, calls the **Logic Service's core endpoint** /logic/holidays?country=CN&year=2026&name={name}, and returns the matching result.
endpoint 6 (Ireland Specific Holiday Time)	GET /api/v1/holidays/IE/2026/name/{name}	Queries the date of a specific Irish holiday in 2026: Finds the corresponding date based on the holiday name ({name}).	The user calls this endpoint. The **Gateway endpoint** extracts {name} from the URL, calls the **Logic Service's core endpoint** /logic/holidays?country=IE&year=2026&name={name}, and returns the matching result.

5. Development and Deployment Guidelines

Development Sequence: The sequence must be strictly followed: **DB Service→ Logic Service → Gateway Service**. It is essential to ensure that the previous endpoint is fully functional and working correctly before developing the next endpoint that depends on it.

Caching Policy: When processing a query request, the **Logic Service** will first call the **DB Service** to check if the data has been stored. The **Logic Service** will only call the third-party API if the cache is expired or does not exist, in order to reduce external dependency and improve response speed.