Got it. Let me break down and outline what you've described for your dashboard application:

1. *Purpose of the Dashboard Application:*

   - A unified dashboard to access and view information from multiple applications (e.g., LinkedIn, Gmail, Outlook, Google, Slack).

   - Users can see all messages and notifications in one place without switching between apps.

2. *Main Components:*

   - *Dashboard Interface:* The front-end that displays data from all integrated applications.

   - *API Communication Program:* A backend program that communicates with various APIs of the applications.

   - *JSON Config Files:* Configuration files for each application's API detailing common and custom parameters.

   - *Common Code Base:* A set of code that processes the JSON configs and makes API calls.

3. *Steps to Create the Application:*

   - *API Integration:*

     - Develop a program to communicate with the APIs of LinkedIn, Gmail, Outlook, Slack, etc.

     - Ensure the program can handle different API structures by abstracting common and custom parameters.

   - *Strategy Pattern Implementation:*

     - Use the strategy pattern to manage differences and similarities across APIs.

     - Create JSON config files for each API, which include common keys and custom values.

     - Ensure these configs are designed to allow the common code to access all necessary parameters.

   - *JSON Configuration:*

     - Define key-value pairs in JSON files for each application's API.

     - Include both common keys (shared across all APIs) and custom values (specific to each API).

   - *Common Code Development:*

     - Write common code that reads and processes the JSON config files.

     - Implement dependency injection to dynamically inject parameters at runtime.

     - Ensure the code can make requests to the different APIs using the parameters from the JSON configs.

4. *High-Level Diagram:*

   - *Dashboard Interface*: Visual representation of messages and notifications from various applications.

   - *API Communication Layer*:

     - Each Application API (LinkedIn, Gmail, etc.)

     - Common Program to interact with APIs

   - *JSON Config Files*:

     - Separate JSON file for each application's API

- Common and Custom parameters within each JSON

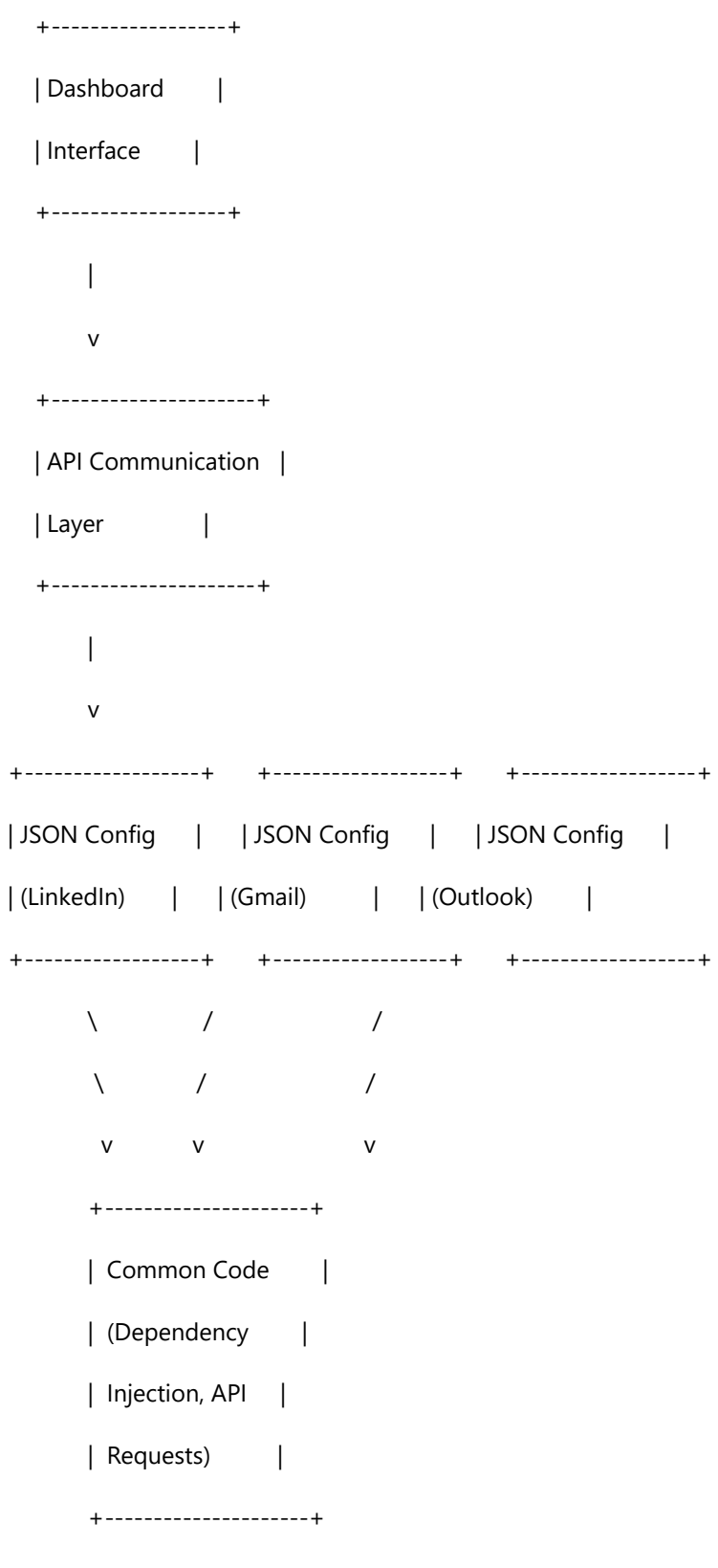
- *Common Code*:

- Reads JSON files

- Performs dependency injection

- Makes API requests

Here is a high-level diagram representation:



In this diagram:

- The *Dashboard Interface* is the front-end that users interact with.

- The *API Communication Layer* handles communication with various application APIs.

- The *JSON Config Files* for each application (LinkedIn, Gmail, Outlook, etc.) contain the necessary configuration details.

- The *Common Code* processes the JSON config files, performs dependency injection, and makes the API requests to gather data from the different applications.

There is another unit or service that will use the CAL to get the urls for the emails and other details. Then after the urls are gotten, they are pushed into a queue and there is another service that will take each url from the queue and again pass it to the CAL and ask it to get the message from each url. Then after the message is gottten, it will store it into the database.