**Declare packages and dependencies**

* Use a [pubspec.yaml](vscode-file://vscode-app/c:/Users/cashamil/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) file that defines the project name as “catalyst\_center\_app” a description, and version “1.0.0”.
* Specify the Dart SDK environment to be “>=2.17.0 <3.0.0”.
* Under dependencies, include the Flutter SDK and the HTTP package version ^0.13.3 and intl package version ^0.18.0. Under dev\_dependencies add the test package version ^1.16.0.
* Ensure that the “flutter” section has “uses-material-design: true”.

**Build Project and Define Code**

**NOTE:** The directory structure is already built out. Do not output any code or instructions regarding directory structure—provide only the source code for the files as specified. The files that need updating are **main.dart**, **my\_dart\_app.dart**, **services/api\_service.dart**., and **config.dart**. " You are an expert in software development, network engineering, and Cisco Catalyst Center. Build a complete Flutter application that implements the following requirements exactly as specified below. Do not help me create a directory structure. Just give me code for each file.

**Project Overview and Directory Structure**

• Name the project **catalyst\_cli\_executor**.  
• The project root must include a **pubspec.yaml** file and a **lib** folder.  
• Under **lib**, create exactly the following files (and no additional files such as models.dart):  
 – **main.dart**  
 – **my\_dart\_app.dart**  
 – A folder **services** containing **api\_service.dart**  
• Also include desktop support directories (windows, linux, macos) as needed.

**File: main.dart requirements**

1. **HTTP Overrides for Certificate Validation**  
    • Create a class named **MyHttpOverrides** that extends **HttpOverrides**.  
    • Override **createHttpClient** so that its **badCertificateCallback** always returns true (trusting all certificates).  
    • Include a comment making it clear that this override is for development only.
2. **Main Entry Point**  
    • In the **main()** function, set **HttpOverrides.global** to an instance of **MyHttpOverrides**.  
    • Then call **runApp()** with an instance of **MyApp** (from my\_dart\_app.dart).
3. **Platform Integration**  
    • Ensure that any necessary platform integration code is included for desktop builds.

**File: my\_dart\_app.dart requirements**

1. **StatefulWidget Implementation**  
    • Implement a StatefulWidget called **MyApp** as the main UI.  
    • Define (or import) the **Device** model in one place (for example, in my\_dart\_app.dart) so that both the UI and ApiService share the same definition. Do not use any extra file (like models.dart).
2. **UI Theme and Styling**  
    • Use a dark theme with:  
     – Brightness set to dark  
     – Scaffold background color: **#121212**  
     – Primary color: **#1E88E5**  
     – Secondary (accent) color: **#9C27B0**  
     – Text (including labels etc.): similar to **#E0E0E0**
3. **AppBar and Title**  
    • The AppBar must have the title **“Catalyst CLI Executor”** and match the dark styling.
4. **Issue Banners Section**  
    • Below the AppBar, include a section (using Containers with an amber background) to display multiple issue banners with each showing:  
     – Issue priority  
     – Issue description (from the “name” field returned by the issues API)  
     – Affected device name (extracted with a regex from the description)  
     – A formatted timestamp in HH:MM:SS format
5. **Device Dropdown**  
    • Include a dropdown widget (without underline) populated with the list of network devices received from the API.  
    • Each dropdown item must show “DeviceName - IP.”  
    • Automatically select the first device (if available) on startup.
6. **CLI Command Input and Execution**  
    • Include a text field labeled **“Enter CLI Command”** for user command input.  
    • Add an ElevatedButton labeled **“Run”** styled with:  
     – Purple background: **Color.fromARGB(255, 79, 29, 136)**  
     – Text in a color similar to **#E0E0E0**  
    • When pressed, if a device is selected and the CLI command is not empty, call **ApiService.runCommand** passing the selected device’s uuid and the command.  
    • The API flow must:  
     – Submit the command to get a task ID  
     – Poll the task API until a file ID is available (up to 20 attempts with 2-second intervals)  
     – Download the file content that contains the CLI output  
    • Display the returned CLI output in a scrollable area with a background color **#333333** using a monospace font.
7. **Initialization and Error Handling**  
    • In **initState()**, first authenticate, then load network devices and issues.  
    • Log (using print statements) the retrieved devices (their count, names, and IPs) and select the first available device.  
    • While network requests are pending, display a **CircularProgressIndicator**.  
    • If an error occurs, display the error message in red text.

**File:**[api\_service.dart](vscode-file://vscode-app/c:/Users/cashamil/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)**requirements**

1. **Class Implementation**  
    • Implement a class called **ApiService** to handle all networking calls.  
    • Do not import or use any models.dart; use only the Device and Issue models as defined within the project.
2. **API Base URL**  
    • Hardcode the base URL as **“**[**https://198.18.129.100”**](vscode-file://vscode-app/c:/Users/cashamil/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).
3. **Authentication Method**  
    • Create a method that retrieves credentials (username and password) from **config.dart**.  
    • Send a POST request to **[baseUrl]/dna/system/api/v1/auth/token** using an auth header constructed via Base64 encoding (“Basic username:password”).  
    • Include a placeholder for AES support if a boolean flag (useAES) is true (with header prefix **“CSCO-AES-256 credentials=”**).  
    • If the status is 200 and the JSON response contains a “Token” field, return the token. Otherwise, throw exceptions as needed (e.g., “Invalid credentials” for status 401).
4. **getDevices Method**  
    • Implement a method that sends a GET request to **[baseUrl]/dna/intent/api/v1/network-device**.  
    • In the response, use the “uuid” field (or fall back to “id”) for the device identifier.  
    • Create Device objects (using the shared model; ensure that uuid, name, and IP are non-null) and log a warning if the identifier is missing.  
    • Return the list of devices.
5. **getIssues Method**  
    • Implement a method that calculates a 24-hour time window (using milliseconds since epoch) and sends a GET request to **[baseUrl]/dna/intent/api/v1/issues** with the start and end times.  
    • Parse the returned JSON (from “response”) to create Issue objects with:  
     – Priority  
     – Description (from “name”)  
     – Affected device name (extracted via regex)  
     – Last occurrence time converted to a DateTime

• Return the list of issues.

1. **runCommand Method**  
    • Implement a method that sends a POST request to **[baseUrl]/dna/intent/api/v1/network-device-poller/cli/read-request** with a JSON payload that includes:  
     – A timeout (default 300 seconds)  
     – Description “Execute CLI command”  
     – Name “runCLICommand”  
     – A list named “commands” with the entered command  
     – A list named “deviceUuids” with the selected device’s uuid  
    • If the response status is 200, 201, or 202, parse the JSON for a “taskId” from the “response.” If no taskId is found, throw an exception.  
    • Call a helper (\_pollTaskResult) that polls **[baseUrl]/dna/intent/api/v1/task/[taskId]** every 2 seconds (up to 20 attempts) for a “progress” field that, when decoded, contains a “fileId.”  
    • Once a fileId is obtained, call another helper (\_getFileContents) to GET **[baseUrl]/dna/intent/api/v1/file/[fileId]**.  
    • From the file response (a list), extract the “commandResponses” map, then retrieve the output under the “SUCCESS” key for the command.  
    • Return the CLI output as a string (or an error message if not found).

**Additional High-Level Behaviors**

• In **MyApp**’s **initState()**, perform authentication, then load devices and issues, logging their details and selecting the first device by default.  
• If network calls are in progress, show a **CircularProgressIndicator**.  
• Display errors (from authentication, device, issue, or command API calls) in red text.  
• Throughout, use print statements to log debugging information.  
• Use the specified API endpoints and JSON structures exactly, with the hardcoded IP of [**https://198.18.129.100**](vscode-file://vscode-app/c:/Users/cashamil/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).  
• All UI elements (AppBar, issue banners, dropdown, CLI input, run button, CLI output area) must use the provided color scheme:  
 – Dark background: **#121212**  
 – Primary: **#1E88E5**  
 – Secondary/accent: **#9C27B0**  
 – Text: **#E0E0E0**  
 – CLI output area: **#333333**

-Text for the banner events: 000000

**Summary**

Do not help me create a directory structure. This is important. Just give me code for each file. Build a complete Flutter application for **catalyst\_center\_app** that:

• Overrides certificate validation in **main.dart** with **MyHttpOverrides** (including a development-only warning).  
 • Implements a single, consistent StatefulWidget (**MyApp**) in **my\_dart\_app.dart** with an exact dark theme, an AppBar titled **“Catalyst CLI Executor”**, issue banners, a dropdown for network devices, a text field for CLI commands, and a properly styled run button.  
 • Uses **ApiService** in **services/api\_service.dart** to handle API calls (authentication, retrieving devices, retrieving issues, and running CLI commands via a multi-step API flow) exactly as specified.  
 • Ensures that the Device model is defined in one place (do not import or reference models.dart) so there are no type mismatches.  
 • Handles loading states, error displays, and logs debugging information throughout.  
 • Uses all API endpoints and JSON structures exactly as documented, with the hardcoded IP [**https://198.18.129.100**](vscode-file://vscode-app/c:/Users/cashamil/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).