Complete Flow For API Integration: From Login to Home Screen

---

1. Login Flow

Step 1: User Opens the App

- First Screen:
  - The user is shown a Login Screen where they can enter their mobile number.

Step 2: OTP Flow

- Send OTP Request:
  - After entering the mobile number, the user clicks on “Send OTP.”
  - If the request is successful, navigate to the Enter OTP Screen.
  - Add a 30-second timer for OTP resend.
- Verify OTP:
  - User enters the OTP.
  - Make a Verify OTP API call.
  - If OTP is correct, the response will include:
    - Token (store securely using libraries like flutter_secure_storage).
    - Profile Updated Key (true/false).

---

2. Onboarding Flow

If Profile Updated = true

- Show Home Screen: #### If Profile Updated = false
- Show Onboarding Screen:
  - Fetch dropdown data (e.g., college, university) and slider data using the Dropdown API.
  - Store this data in Hive for offline use.
- User Inputs Information:
  - Allow the user to select their college/university using a dropdown (filterable).
  - Collect other required profile information.
- Device Token:
  - Retrieve the device’s FCM Token for push notifications.
- Update Profile API:
  - Send a PATCH request with the user’s inputs and FCM token.
  - If successful, navigate to the Select Subject Screen.

Caching Strategy for Onboarding:

- Dropdown Data:
  - Use Dio for API calls and cache data in Hive.
  - Update dropdown data every 7 days in the background.

---

3. Select Subject Flow

Step 1: Fetch Subjects

- Call the Fetch Subject API with the token in the header.
- Parse the response to separate:
  - Selected Subjects: Show these in the “Selected Subjects” list.
  - Unselected Subjects: Show these in the complete subject list.

Step 2: User Selects Subjects

- Allow the user to filter the subject list using a search bar.

Step 3: Save Selection

- On clicking “Save,” send the selected subjects to the backend via the API.
- Navigate to the Home Screen.

Caching Strategy for Subjects:

- Use Dio for API calls.
- Cache the subject data temporarily but avoid saving it to Hive.

---

4. Home Screen Flow

Step 1: Fetch APIs for Home Screen

- Call these APIs in parallel using Dio:
  1. User Profile API:
     - Fetch branch ID, university ID, college ID, etc.
     - Use this data to display the user’s profile.
  2. App Info API:
     - Fetch the marketingAssets (banners, images, videos).
     - Display these assets in the banner/slider section.
  3. Free Resources API:
     - Fetch data for the “Free Resources” section.

Step 2: Use Caching for Instant Load

- User Profile Data: Cache in Hive for offline access.

- Marketing Assets: Cache in Hive for instant banner loading.
- Free Resources:
    - Cache in Hive.
    - Show cached data initially and update in the background.

Step 3: Background Updates

- While APIs are being fetched, show a progress bar under “Free Resources” with a label like “Updating data…”
- Display old data from Hive until new data is fetched.

---

5. Handling Free Resources

Step 1: Nested Structure

- The Free Resources section can have a nested folder structure.
- Handle dynamic nesting and support file types like:
    - PDFs
    - YouTube videos
    - Self-hosted videos

Step 2: Caching and Updating

- On the first fetch:
    - Save the free resources data in Hive.
- For subsequent updates:
    - Fetch the data in the background.
    - Replace old data only after a successful API response.

---

6. Change Course Flow

Step 1: Update Free Resources

- On course change, fetch new free resources using the Free Resources API.
- Do not delete old data before fetching.
- Replace Hive data with new data after a successful API fetch.

UI Update:

- Reflect the new course name and free resources dynamically using state management (Provider, Bloc, or Riverpod).

---

7. Background Workers

Use workmanager to set up periodic tasks for background updates.

Background Update Tasks

- Update Dropdown Data: Every 7 days, fetch and update dropdown data in Hive.
- Update Free Resources: Trigger this task daily or when a silent push notification is received.

WorkManager Setup

dart import ‘package:workmanager/workmanager.dart’;

void callbackDispatcher() { Workmanager().executeTask((taskName, inputData) async { switch (taskName) { case ‘updateDropdownData’: // Fetch and update dropdown data in Hive break; case ‘updateFreeResources’: // Fetch and update free resources in Hive break; default: break; } return Future.value(true); }); }

void setupWorkManager() { Workmanager().initialize(callbackDispatcher); Workmanager().registerPeriodicTask(‘updateDropdownData’, ‘updateDropdownData’, frequency: const Duration(days: 7),); Workmanager().registerPeriodicTask(‘updateFreeResources’, ‘updateFreeResources’, frequency: const Duration(hours: 24),); }

---

8. Error Handling

- For all API calls, implement retries using Dio’s interceptor.
- Show error messages in the UI if an API call fails, while still displaying cached data.
- Log errors for debugging using services like Sentry or Firebase Crashlytics.

Caching Summary

Data	Caching Method	Update Frequency
Token	Secure Storage	Only when user logs in.
Dropdown Data	Hive	Every 7 days in the background.
User Profile	Hive	On every app open.

Data	Caching Method	Update Frequency
Free Resources	Hive	On course change or daily updates.
Marketing Assets	Hive	On every app open.

End-to-End Flow

1. Login Screen → Enter OTP → Verify OTP → Check profile_updated.
2. Onboarding Screen → Input Data → Update Profile API → Select Subject Screen.
3. Select Subject Screen → Fetch Subjects → Save → Home Screen.
4. Home Screen → Fetch APIs (User Profile, App Info, Free Resources) → Cache and update data.
5. Change Course → Update Free Resources → Reflect dynamically in UI.
6. Background Updates → Periodically refresh dropdown data and free resources.

Error Handling Flow: Including 401, 500 & others Handling and No Data Scenarios

---

Global Error Handling

1. 401 Unauthorized Response

- Behavior:
  - If any API responds with a 401 Unauthorized, it indicates the token is invalid or expired.
  - Action:
    - Clear the user session by:
      1. Deleting the stored token from flutter_secure_storage.
      2. Clearing user-related data from Hive (like dropdown data, free resources, etc.).
    - Redirect the user to the Login Screen with a message: “Session expired. Please log in again.”

Code Example for Global Interceptor

Use Dio interceptors to handle 401 globally:
dart import ‘package:flutter_secure_storage/flutter_secure_storage.dart’;
import ‘package:hive/hive.dart’;

class DioInterceptor extends Interceptor { final FlutterSecureStorage storage;

DioInterceptor(this.storage);

@override void onResponse(Response response, ResponseInterceptorHandler handler) { if (response.statusCode == 401) { _handleLogout(); } else { handler.next(response); // Continue for other responses } }

@override void onError(DioError err, ErrorInterceptorHandler handler) { if (err.response?.statusCode == 401) { _handleLogout(); } else { handler.next(err); // Continue for other errors } }

Future _handleLogout() async { // Clear token and user data await storage.delete(key: 'authToken'); await Hive.deleteBoxFromDisk('userProfile'); await Hive.deleteBoxFromDisk('freeResources'); // Navigate to Login Screen navigatorKey.currentState?.pushNamedAndRemoveUntil('/login', (route) => false); } }

2. Other API Errors

- Behavior:
  - For errors like 500 Internal Server Error or network issues, show a user-friendly error message without crashing the app.
- Action:
  - Display a toast/snackbar with the error: "Something went wrong. Please try again."
  - Keep displaying cached data from Hive if available.

Code Example for Error Messages

dart import 'package:flutter/material.dart';

void showError(BuildContext context, String message) { ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text(message)),); }

UI Design for No Data Scenarios

For every screen or section, implement a fallback UI for "No Data" situations with messages or graphics.

1. Login Screen

- No Data Scenario: Missing or invalid OTP.
- UI Fallback:
  - Show a message: "Invalid OTP. Please try again."
  - Allow the user to resend OTP after 30 seconds.

2. Onboarding Screen

- No Data Scenario: Dropdown API fails to load colleges or universities.
- UI Fallback:
  - Display a graphic (like a sad face) with a message:
    “Unable to load colleges. Please check your connection.”
  - Provide a retry button to re-fetch data.

3. Select Subject Screen

- No Data Scenario: No subjects available or API fails.
- UI Fallback:
  - Show a message: “No subjects available at the moment.”
  - Add a graphic and a retry button to fetch subjects again.

4. Home Screen

- No Data Scenarios:
  1. Free Resources: If no data is available, show a placeholder message:
     “No resources available. Please try again later.”
  2. Banner/Slider: If marketing assets fail to load:
     “Unable to load banners.”
- UI Fallback:
  - Add a retry button or periodically retry in the background.

5. Free Resources Section

- No Data Scenario: No resources for the selected course.
- UI Fallback:
  - Display a message: “No free resources available for this course.”
  - Use a placeholder graphic for visual appeal.

Error Handling Summary

Error Type	Action	UI Fallback
401 Unauthorized	Logout user, clear cache, and redirect to Login.	Show: “Session expired. Please log in.”
API Failures	Display cached data if available. Retry fetching or show: “Something went wrong.”	Retry button and user-friendly message.

Error Type	Action	UI Fallback
No Data	Show placeholder UI with a message and retry option.	Graphic + Message: “No data available.”

Code Example for No Data UI

dart import ‘package:flutter/material.dart’;

Widget noDataWidget(String message, VoidCallback onRetry) { return Column(mainAxisAlignment: MainAxisAlignment.center, children: [Icon(Icons.warning, size: 80, color: Colors.grey), SizedBox(height: 20), Text(message, style: TextStyle(fontSize: 18, color: Colors.grey)), SizedBox(height: 20), ElevatedButton(onPressed: onRetry, child: Text(“Retry”),),],); }

Flow Recap

1. 401 Handling: Log out the user and clear data.
2. Error Handling: Gracefully show errors and retain cached data.
3. No Data Scenarios: Use meaningful messages and retry buttons.
4. Dynamic Updates: Ensure every screen updates when new data is fetched successfully.

After Login User Open the App:

- Directly Redirect to Home Screen