PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)

Academic Year: 2024-25
Semester: VI
Class / Branch / Div: TE- IT A/B/C
Subject: MAD & PWA Lab
Name of Instructor: Manjusha Kashilkar

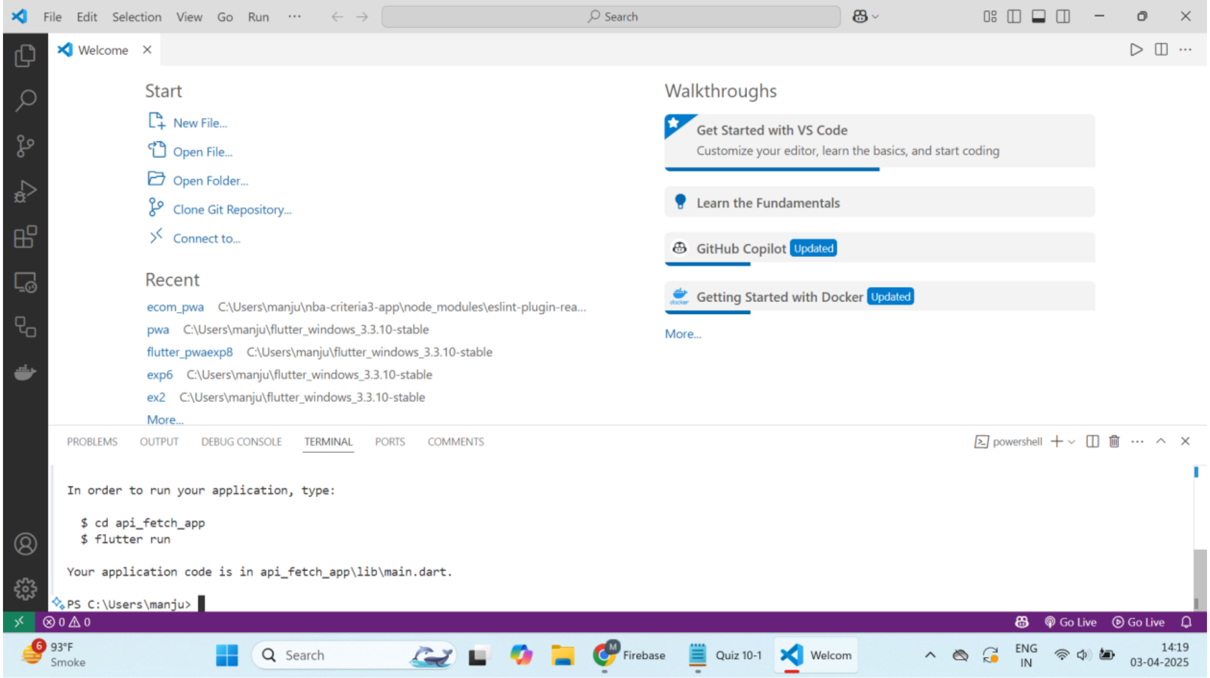
Name of Student: Chirag Malde
Student ID: 22104186
Roll No. 17
Date of Submission: 07/04/25

Experiment No-12

1-Create a New Flutter Project in VS Code

1. Open **VS Code**.
2. Open **Terminal** (Ctrl + ~ in VS Code).
3. Run the following command:

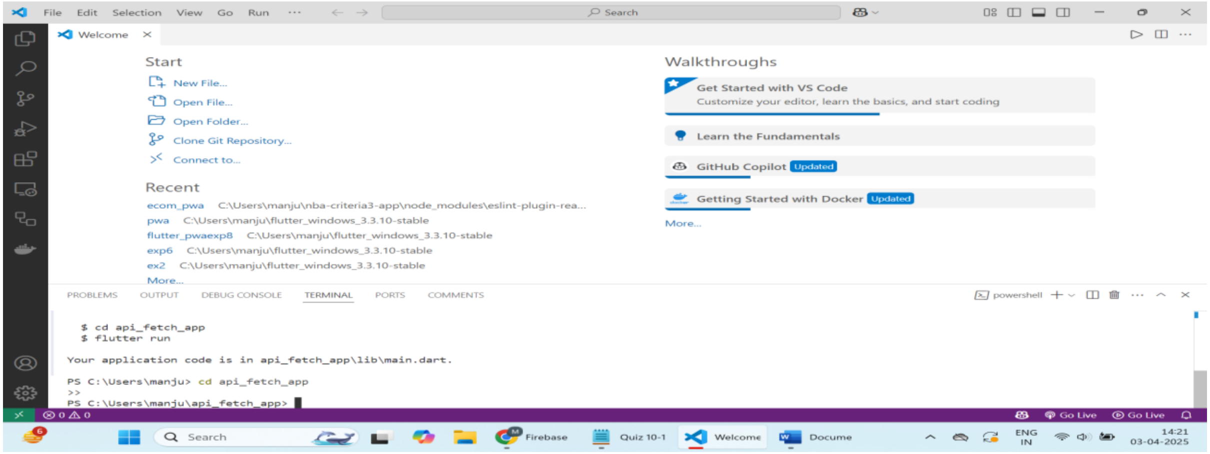
➢ **flutter create api_fetch_app**


File Edit Selection View Go Run ···
Search
Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
ecom_pwa C:\Users\manju\nba-criteria3-app\node_modules\eslint-plugin-rea...
pwa C:\Users\manju\flutter_windows_3.3.10-stable
flutter_pwaexp8 C:\Users\manju\flutter_windows_3.3.10-stable
exp6 C:\Users\manju\flutter_windows_3.3.10-stable
ex2 C:\Users\manju\flutter_windows_3.3.10-stable
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
GitHub Copilot Updated
Getting Started with Docker Updated
More...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
powershell
⊗ 0 ⚠ 0
Go Live
Go Live
93°F
Smoke
Search
Firebase
Quiz 10-1
Welcom
ENG
IN
14:19
03-04-2025

```
In order to run your application, type:

  $ cd api_fetch_app
  $ flutter run

Your application code is in api_fetch_app\lib\main.dart.

PS C:\Users\manju>
```

Navigate to the project folder:

➢ **cd api_fetch_app**

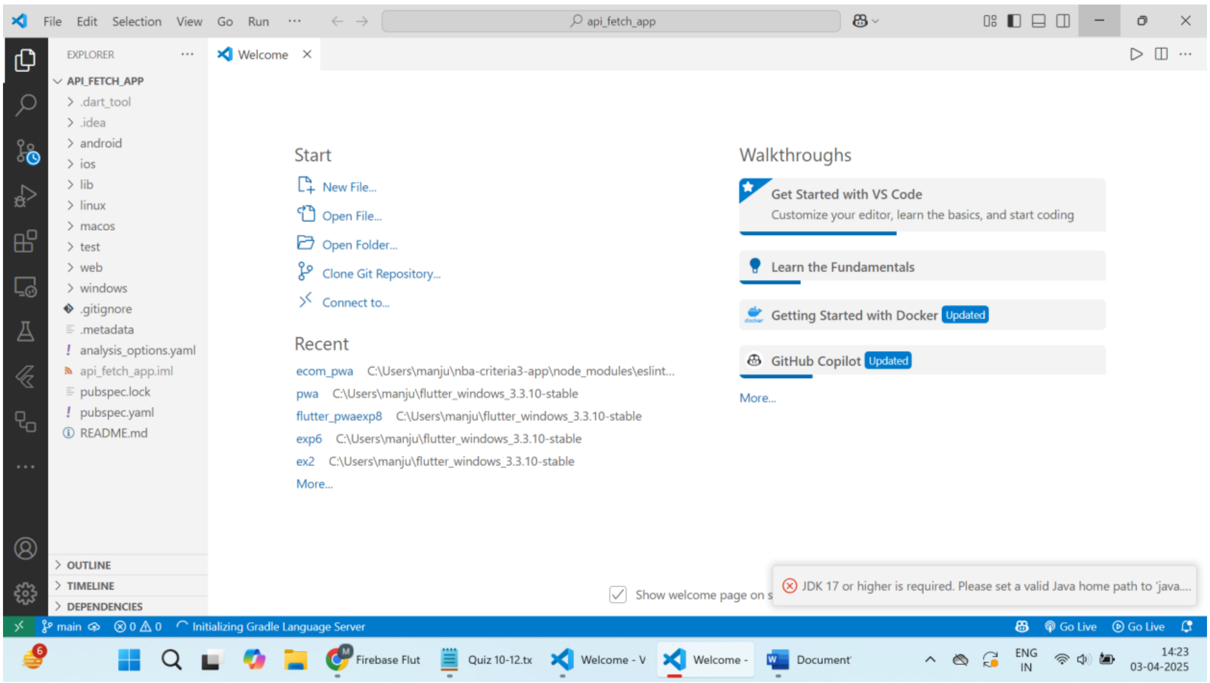


Welcome
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
ecom_pwa C:\Users\manju\nba-criteria3-app\node_modules\eslint-plugin-rea...
pwa C:\Users\manju\flutter_windows_3.3.10-stable
flutter_pwaexp8 C:\Users\manju\flutter_windows_3.3.10-stable
exp6 C:\Users\manju\flutter_windows_3.3.10-stable
ex2 C:\Users\manju\flutter_windows_3.3.10-stable
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
GitHub Copilot Updated
Getting Started with Docker Updated
More...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
powershell

```
  $ cd api_fetch_app
  $ flutter run

Your application code is in api_fetch_app\lib\main.dart.

PS C:\Users\manju> cd api_fetch_app
>>
PS C:\Users\manju\api_fetch_app> 
```

Open the project in VS Code:

- ➢ **code .**



JDK 17 or higher is required. Please set a valid Java home path to 'java....

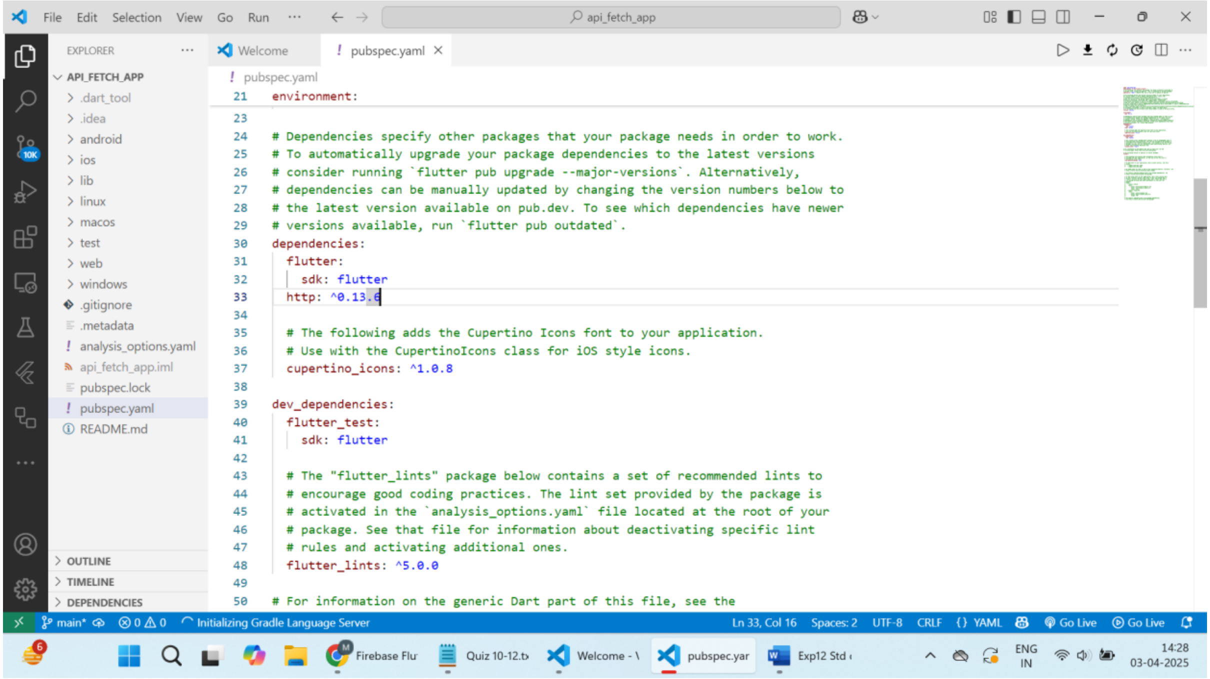
Initializing Gradle Language Server

Step 2 -Add HTTP Dependency in pubspec.yaml

To make API calls, we need the http package.

1. **Open pubspec.yaml in VS Code.**
2. **Add the http package under dependencies:**

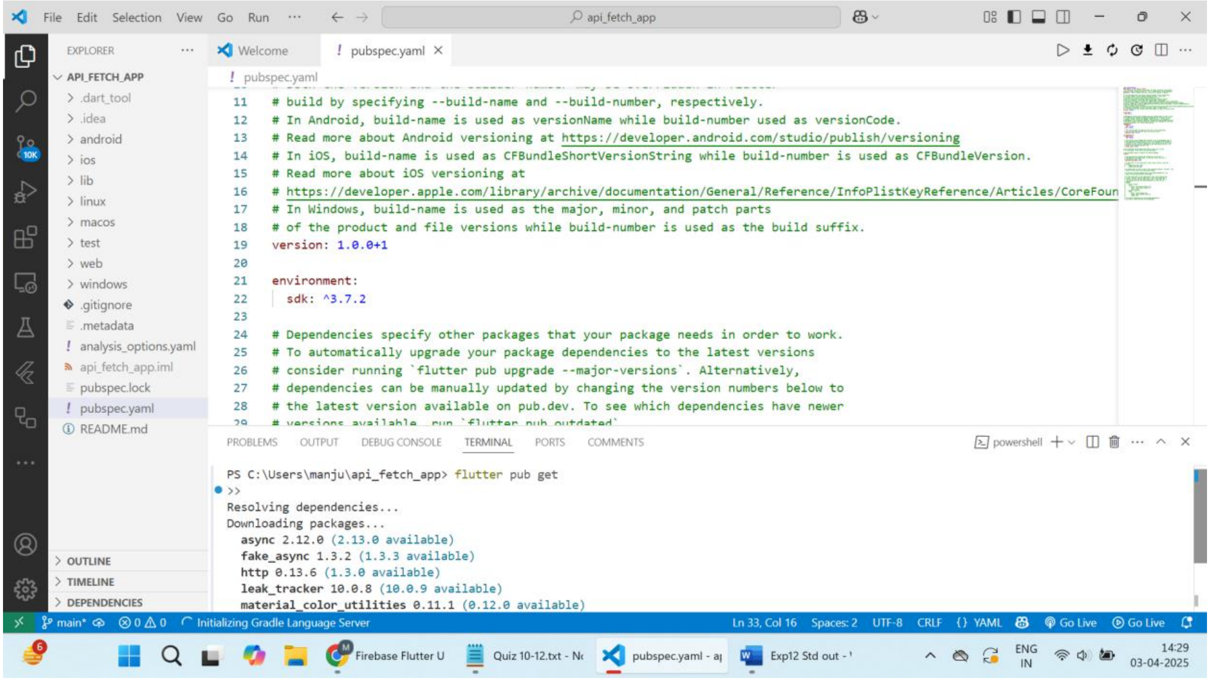
```
dependencies:
  flutter:
    sdk: flutter
  http: ^0.13.6
```



```
environment:

# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter
  http: ^0.13.6

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8

dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the `analysis_options.yaml` file located at the root of your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
  flutter_lints: ^5.0.0

# For information on the generic Dart part of this file, see the
```

Save the file and run:

➢ **flutter pub get**



```yaml
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as versionCode.
# Read more about Android versioning at https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.
# Read more about iOS versioning at
# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoun
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build suffix.
version: 1.0.0+1

environment:
  sdk: ^3.7.2

# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`
```

```
PS C:\Users\manju\api_fetch_app> flutter pub get
>>
Resolving dependencies...
Downloading packages...
  async 2.12.0 (2.13.0 available)
  fake_async 1.3.2 (1.3.3 available)
  http 0.13.6 (1.3.0 available)
  leak_tracker 10.0.8 (10.0.9 available)
  material_color_utilities 0.11.1 (0.12.0 available)
```

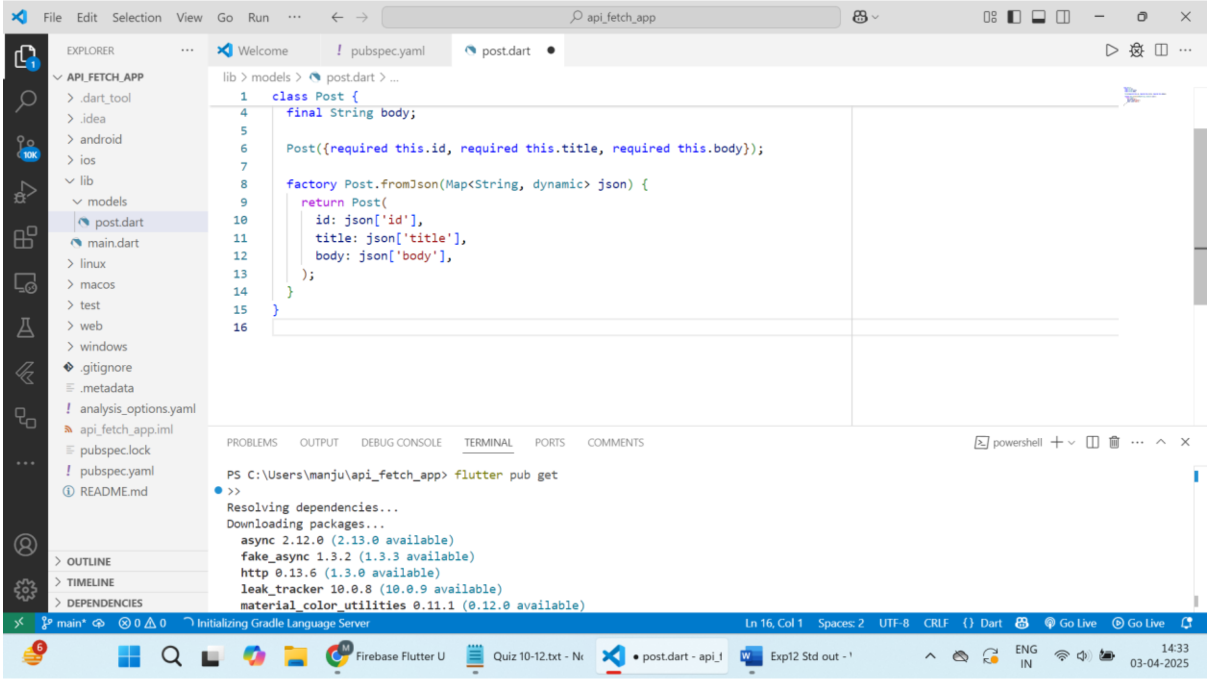
Step 3: Create a Model Class for JSON Parsing

1. Inside the lib/ folder, create a **new folder** named models.

2. Inside the models folder, create a file named post.dart.

3. Add the following Dart code to define the model class:

1. Inside the lib/ folder, create a **new folder** named models.



```
class Post {
  final String body;

  Post({required this.id, required this.title, required this.body});

  factory Post.fromJson(Map<String, dynamic> json) {
    return Post(
      id: json['id'],
      title: json['title'],
      body: json['body'],
    );
  }
}
```

```
PS C:\Users\manju\api_fetch_app> flutter pub get
>>
Resolving dependencies...
Downloading packages...
  async 2.12.0 (2.13.0 available)
  fake_async 1.3.2 (1.3.3 available)
  http 0.13.6 (1.3.0 available)
  leak_tracker 10.0.8 (10.0.9 available)
  material_color_utilities 0.11.1 (0.12.0 available)
```

Code to be Add in dart file -

```
class Post {
  final int id;
  final String title;
  final String body;

  Post({required this.id, required this.title, required this.body});

  factory Post.fromJson(Map<String, dynamic> json) {
    return Post(
      id: json['id'],
      title: json['title'],
      body: json['body'],
    );
  }
}
```

This converts JSON data from the API into a Dart object.

Step 4: Create API Service to Fetch Data

1. Inside the lib/ folder, create a new folder named services.

2. Inside services, create a file named api_service.dart.

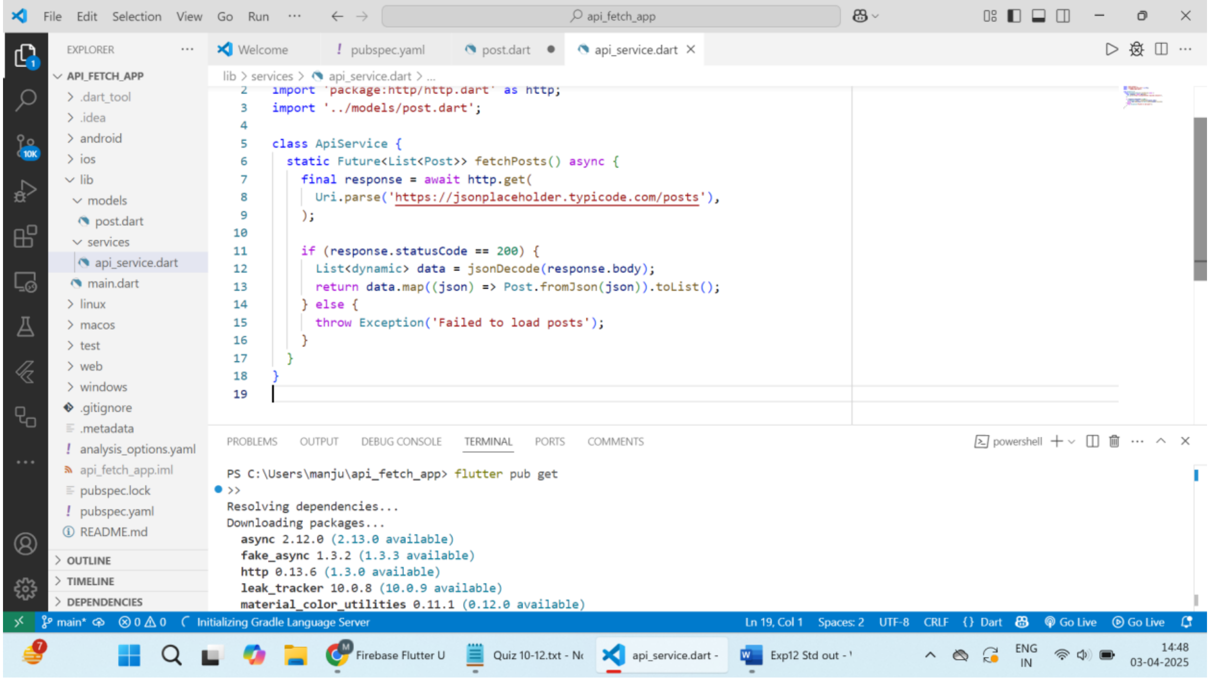
3. Add the following code to fetch API data:

api_service.dart

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import '../models/post.dart';

class ApiService {
  static Future<List<Post>> fetchPosts() async {
    final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts'));

    if (response.statusCode == 200) {
      List<dynamic> data = jsonDecode(response.body);
      return data.map((json) => Post.fromJson(json)).toList();
    } else {
      throw Exception('Failed to load posts');
    }
  }
}
```

This function:

- Sends an **HTTP GET request**.
- Decodes the **JSON response**.
- Maps JSON to a list of Post objects.



```dart
import 'package:http/http.dart' as http;
import '../models/post.dart';

class ApiService {
  static Future<List<Post>> fetchPosts() async {
    final response = await http.get(
      Uri.parse('https://jsonplaceholder.typicode.com/posts'),
    );

    if (response.statusCode == 200) {
      List<dynamic> data = jsonDecode(response.body);
      return data.map((json) => Post.fromJson(json)).toList();
    } else {
      throw Exception('Failed to load posts');
    }
  }
}
```

```
PS C:\Users\manju\api_fetch_app> flutter pub get
>>
Resolving dependencies...
Downloading packages...
  async 2.12.0 (2.13.0 available)
  fake_async 1.3.2 (1.3.3 available)
  http 0.13.6 (1.3.0 available)
  leak_tracker 10.0.8 (10.0.9 available)
  material_color_utilities 0.11.1 (0.12.0 available)
```

Step 5: Build the UI to Display Data

1. Open lib/main.dart in VS Code.
2. Replace its contents with the following code:

lib/main.dart

import 'package:flutter/material.dart';

import 'services/api_service.dart';

import 'models/post.dart';

void main() {

  WidgetsFlutterBinding.ensureInitialized();

```
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'API Fetch Example',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: HomeScreen(),
    );
  }
}

class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  late Future<List<Post>> futurePosts;

  @override
  void initState() {
    super.initState();
    futurePosts = ApiService.fetchPosts();
  }
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text('Public API Data')),
    body: FutureBuilder<List<Post>>(
      future: futurePosts,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
          return Center(child: CircularProgressIndicator());
        } else if (snapshot.hasError) {
          return Center(child: Text('Error: ${snapshot.error}'));
        } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
          return Center(child: Text('No data available'));
        } else {
          return ListView.builder(
            itemCount: snapshot.data!.length,
            itemBuilder: (context, index) {
              Post post = snapshot.data![index];
              return Card(
                margin: EdgeInsets.all(10),
                child: ListTile(
                  title: Text(
                    post.title,
                    style: TextStyle(fontWeight: FontWeight.bold),
                  ),
                  subtitle: Text(post.body),
                  ),
                );
              },
            );
          }
        },
      ),
    );
  }
}
```

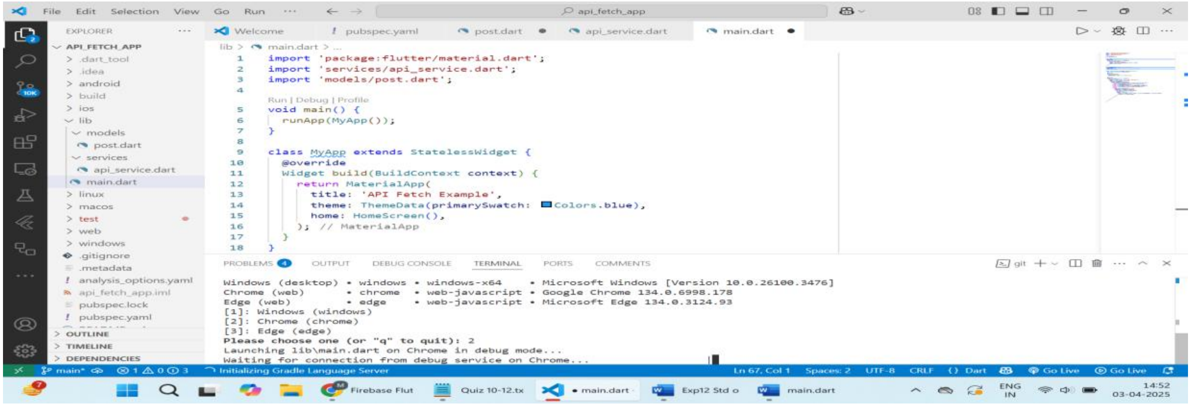
Features of this UI Code:

- Uses **FutureBuilder** to handle API data.
- Displays a **loading spinner** while fetching data.
- Shows **error messages** if API call fails.
- Uses ListView.builder to dynamically display fetched data.

Step 6: Run the Flutter App in VS Code

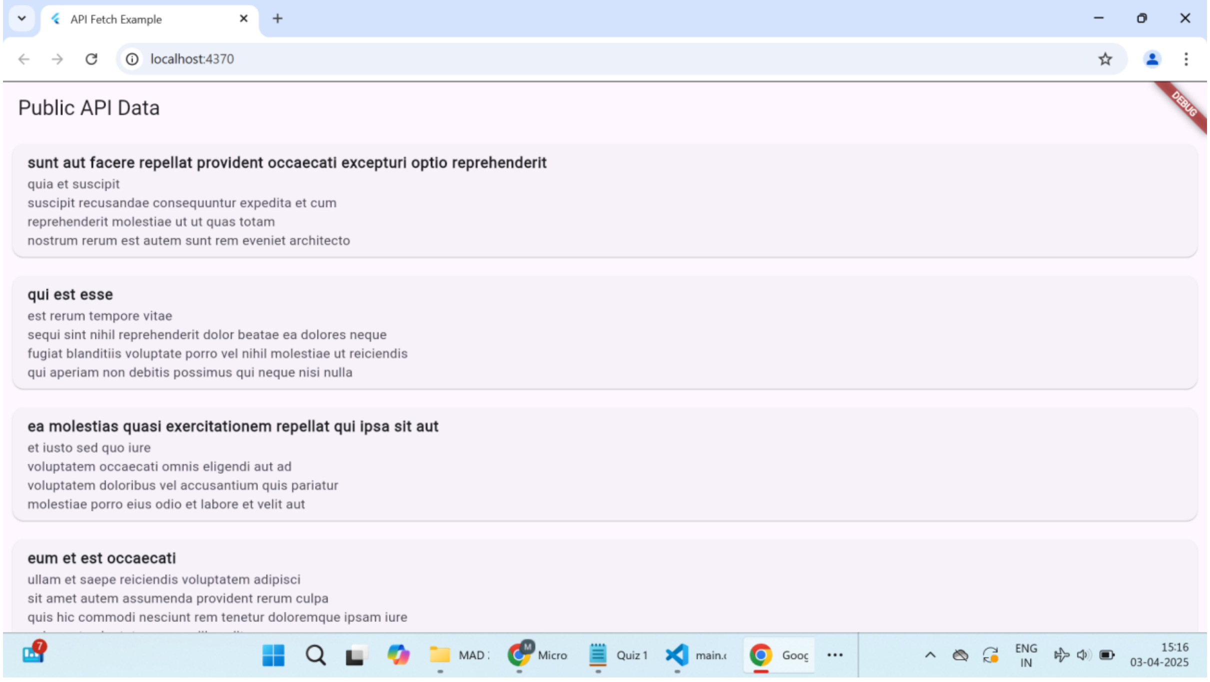
1. **Connect a device** (Android emulator or physical device).
2. Open **Terminal in VS Code** (Ctrl + ~).

   ➢ **flutter run**



```
import 'package:flutter/material.dart';
import 'services/api_service.dart';
import 'models/post.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'API Fetch Example',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: HomeScreen(),
    ); // MaterialApp
  }
}
```

```
Windows (desktop) • windows • windows-x64     • Microsoft Windows [Version 10.0.26100.3476]
Chrome (web)      • chrome  • web-javascript  • Google Chrome 134.0.6998.178
Edge (web)        • edge    • web-javascript  • Microsoft Edge 134.0.3124.93
[1]: Windows (windows)
[2]: Chrome (chrome)
[3]: Edge (edge)
Please choose one (or "q" to quit): 2
Launching lib\main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome...
```



Public API Data

sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto

qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla

ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut

eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque ipsam iure

Step 7: Debugging API Calls

1. Check API Response:
   - Open https://jsonplaceholder.typicode.com/posts in a browser.
2. **Print API Data for Debugging**
   - **Modify fetchPosts() in api_service.dart**

```
import 'dart:convert';

import 'package:http/http.dart' as http;

import '../models/post.dart';
```

```
class ApiService {
  static Future<List<Post>> fetchPosts() async {
    final url = 'https://jsonplaceholder.typicode.com/posts'; // Public API
    final response = await http.get(Uri.parse(url));

    print('Response Code: ${response.statusCode}');
    print('Response Body: ${response.body}');

    if (response.statusCode == 200) {
      List<dynamic> data = jsonDecode(response.body);
      return data.map((json) => Post.fromJson(json)).toList();
    } else {
      throw Exception('Failed to load posts');
    }
  }
}
```

Step 8 -Debug Network Requests in Browser (For Flutter Web)

open Chrome DevTools:

1. **Press F12 → Go to Network tab.**
2. **Click Fetch/XHR.**
3. **Reload the app (Ctrl + R).**
4. **Check if the API request appears in the list.**
5. **Click on the request → View response.**

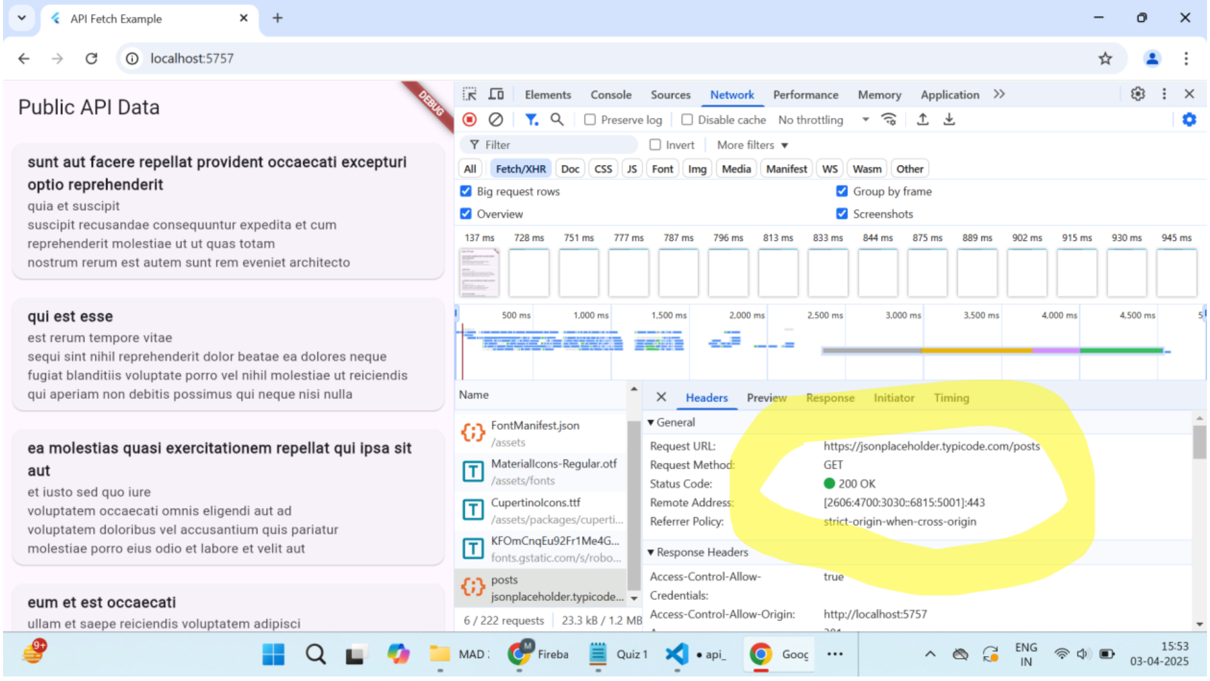
What this does:

- Prints **response status code** (e.g., 200, 404, 500).
- Prints **response body** (JSON data).

Debug Network Requests in Browser (For Flutter Web)

If using Flutter Web, open **Chrome DevTools**:

- Press F12 → Go to **Network** tab.
- Click **Fetch/XHR**.
- Reload the app (Ctrl + R).
- Check if the API request appears in the list.
- Click on the request → View response.
-



Public API Data

sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto

qui est esse
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla

ea molestias quasi exercitationem repellat qui ipsa sit aut
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut

eum et est occaecati
ullam et saepe reiciendis voluptatem adipisci

General

Request URL: https://jsonplaceholder.typicode.com/posts
Request Method: GET
Status Code: ● 200 OK
Remote Address: [2606:4700:3030::6815:5001]:443
Referrer Policy: strict-origin-when-cross-origin

Response Headers

Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://localhost:5757

If you see **"404 Not Found"** or **"500 Internal Server Error"**, the API URL is incorrect.

Conclusion – In this way we have implemented the app,which will Fetch real-time data from an API and it will Parse JSON into Dart objects and also Dynamically update UI with API data.