Lab	Type	Practical
1	A. A. A. A. A.	**WAP** to create a class Student with a method to display student details. **WAP** implements the inheritance of a class to inherit properties and methods from another class. **WAP** for implementation of a method overriding. **WAP** to create abstract classes and methods to define child classes. **WAP** creates a static variable and calls it from two different classes, observing the result.
2	A. A. A. A. B.	**WAP** to create a list with favorite and unfavorite options using a static list. **WAP** to design signup form with (Name, Email, Phone Number, Password, Confirm Password). **WAP** to validate the signup form with the above details. **WAP** to create a profile card with available widgets in Flutter. **WAP** to create a design like an e-commerce (Amazon, Flipkart) application dashboard.
3	A. B.	**WAP** to perform CRUD operations with List<dynamic>. **WAP** to do crud operation with Local database (TBL_USER(UID, Name, City, Gender)).
4	A. A. A. B. B. B.	**WAP** to show Dialog with title, AlertMessage, and buttons using Getx. **WAP** to show SnackBar after clicking on a button using Getx. **WAP** to show the BottomSheet dialog with a list of data in it. **WAP** to show Custom Dialog with different themes using Getx. **WAP** to show Custom SnackBar with different colors, and different locations after clicking on a button using Getx. **WAP** to show custom bottom sheets with different heights and auto close.
5	A. A. A. B. B.	**WAP** to Navigate between two screens using Get.to() & Get.back. **WAP** to pass data between two screens using Get.to(). **WAP** to navigate between screens using a named route. **WAP** to transit between two screens with different animations. **WAP** to transit using GetMiddleware.
6	A. A. A. B. B.	**WAP** to create and use a basic GetX controller with non-reactive variables. **WAP** to use Rx Variables with Obx() & Obs. **WAP** to bind a TextField to a RxString and reflect real-time changes. **WAP** to create a timer countdown using RxInt that updates every second. **WAP** to Use RxBool to show/hide content.
7	A. A. A. B.	**WAP** to display the list in ListView using RXList. **WAP** to display the favorite icon in it and change it based on changes on the RxListitem variable. **WAP** to add/delete items from ListView using RXList. **WAP** to do CRUD operation using RXList<dynamic> (RXList<Map>).

8	A. A. A.	**WAP** to do Basic Exception Handling Using Try-Catch (Divide by zero). **WAP** for Handling Specific Exceptions Using on Keyword e.g., FormatException, SocketException. **WAP** to Display Error Messages Using SnackBar/Dialog.
9	A. A. A. B.	**WAP** to create and initialize a local SQLite database using sqflite package (Student.db, TBL_Student). **WAP** to create a Student model class to map data between SQLite and Dart objects. WAP to list data and display in Listview using RXList. WAP to perform Create multiple tables and apply foreign key on it.
10	A. A. B. B.	**WAP** to insert student data into SQLite using GetX for state management. **WAP** to update and delete the data using Sqlite and update the list. **WAP** to Search or Filter SQLite Data Using GetX. **WAP** to Implement Lazy Loading (Pagination) from SQLite using GetX
11	A. A. B.	**WAP** to Fetch and Display Data from a REST API using GetX. **WAP** to Implement Loading, Success, and Error State in API Calls. **WAP** to Search API Integration with Real-time Filtering.
12	A. A. B.	**WAP** to POST, PUT data using API Getconnect. **WAP** to Delete data with dialog for alert using delete. **WAP** to create To-Do application with api.
13	A. A.	**WAP** to Handling Asynchronous Exceptions (Future-based). **WAP** to Creating and Using Custom Exceptions.
14	A. A.	**WAP** to create custom pub and publish it. **WAP** to use custom pub in application.
15	A. A. A. A. B.	**WAP** To demonstrate how to request and handle camera permission in a Flutter app using the **permission_handler** package. **WAP** to implement that requests storage permission to read/write files to the device. **WAP** to implement user's location after requesting the required location permissions. **WAP** to handling multiple permissions at once. **WAP** handling denied or permanently denied permissions.
16	A. A. A. A. B. B.	**WAP** to read data from a file stored in external storage. **WAP** to write data to file stored in external storage. **WAP** to delete file from external storage using code. **WAP** to download file from web and store in external storage. **WAP** to display message if same name file is already available in storage. **WAP** to display all files in available in external storage.
17	A. A. A. B.	**WAP** to display width and height of screen using media query. **WAP** to display text and background color based on width (if width > 700 ? Mobile : WEB). **WAP** to apply responsive padding using MediaQuery. **WAP** to change UI layout based on orientation.

18	A. A. A. B. B.	**WAP** to build a responsive grid layout based on screen width. **WAP** to show different widgets on small and large screens. **WAP** to apply dynamic font size based on screen width. **WAP** to create a sidebar for tablet and bottom navigation for mobile. **WAP** to create responsive form layout based on orientation and size.
19	A. A. B.	**Implement firebase configuration to your project.** **WAP** to sign in using goole sign-in/sign-up. **WAP** to save google sign-in user data to fire-sorage.
20	A. A. B.	**WAP** to perform crud operation using firestore. **WAP** to upload and retrieve files using firebase storage. **WAP** to perform real-time data sync using firebase realtime database.
21	A. A. A. B.	**WAP** to implement firebase cloud messaging configuration to application. WAP to read firebase push notification data in application. WAP to configure application to handle push notification even if application is killed. WAP to read data from data payload and open particular page in application.
22	A. A. A. B.	**WAP** to configure google map in console. WAP to implement google map in application. WAP to place markers on google map and display tool tip on it. WAP to create application that displays different hotels with markers on google map.
23	A. A. A. B.	**WAP** to access location service from application and display latitude and longitude. WAP to place camera on google map with particular location and place initial zoom. **WAP** to display custom marker on google map. WAP to display address above the map on click of marker or place.
24	A. B.	**WAP** to display animated marker on google map. **WAP** to create application for capturing geotag photo.
25	A. A. B.	**WAP** to implement internationalization using getx. WAP to choose language from settings and set as default language (User can change language any time from settings). **WAP** to change screen direction based on language.
26	A. A. A.	**WAP** to display implicit animation. **WAP** to Use AnimatedContainer to animate size and color change on user interaction. **WAP** to apply animatedopacity and animated padding on the container.
27	A. A. A.	**WAP** to Implement Explicit Animations using AnimationController. **WAP** to trigger animation on button tap the widget will move from left to right. **WAP** to trigger animation on button tap and resize smoothly with the defined curve and duration.

27	A.	**WAP** to implement draggable widget or object on the screen that reacts with bouncing or snapping effects in Flutter. **WAP** to trigger animation with move from one location to another location.
28	A. A. B. B.	**WAP** to create a Flutter application that fetches and displays the device battery level by calling a native Android function using MethodChannel. **WAP** to access and display native device information such as the model name, OS version, and manufacturer using MethodChannel in Flutter. **WAP** to launch the native camera application using MethodChannel and return a confirmation or captured image path. **WAP** to Reading Device's Native Contacts Using MethodChannel.
29	A. A. A. A.	**WAP** to create a Flutter app that computes the nth Fibonacci number using recursion inside an isolate. The app allows users to input a number, sends the task to the isolate, and then displays the result when the isolate completes the task. **WAP** to implement a file picker to select a large text file. The file will be read in a separate isolate and then displayed on the screen. **WAP** to reads a large local JSON file (containing user profiles), sends it to compute() for parsing, and displays the list of users on the UI. **WAP** to runs a Timer.periodic() and sends current time or tick count every 5 seconds to the main isolate, which updates the screen or console.
30	A. A. B.	**WAP** to establish connection between client and server. **WAP** that establishes a real-time chat between two clients using Socket.IO. **WAP** that shows a “User is typing…” indicator when another user starts typing in a chat box.