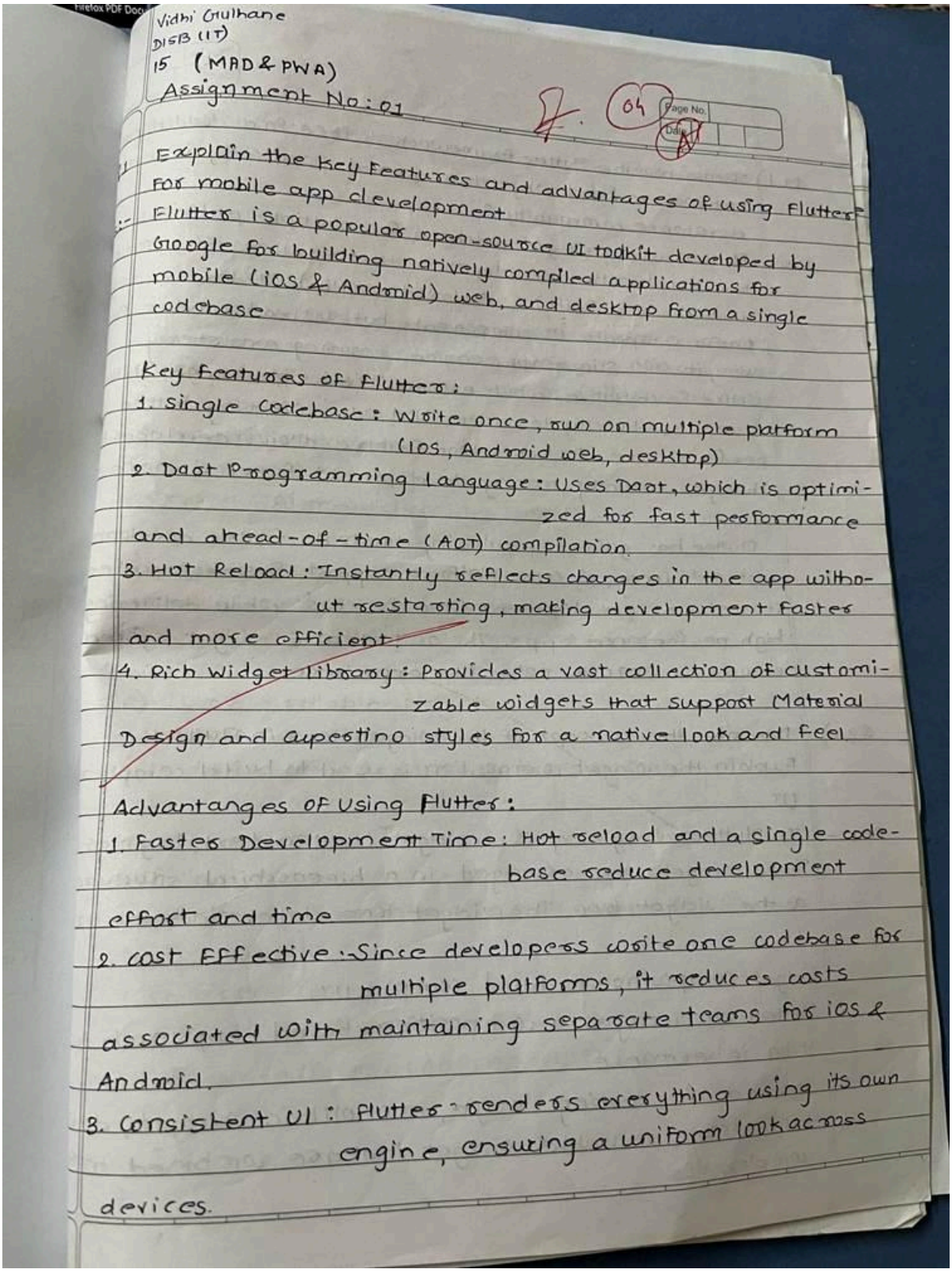
NAME: Vidhi Gulhane **Class: D15B** **ROLL NO.15**

Assignment No: 01



Vidhi Gulhane
D15B (IT)
15 (MAD & PWA)
Assignment No: 01

1 Explain the key features and advantages of using Flutter for mobile app development

:- Flutter is a popular open-source UI toolkit developed by Google for building natively compiled applications for mobile (iOS & Android) web, and desktop from a single codebase

Key features of Flutter:

1. Single Codebase: Write once, run on multiple platform (iOS, Android web, desktop)
2. Dart Programming Language: Uses Dart, which is optimized for fast performance and ahead-of-time (AOT) compilation.
3. Hot Reload: Instantly reflects changes in the app without restarting, making development faster and more efficient.
4. Rich Widget Library: Provides a vast collection of customizable widgets that support Material Design and Cupertino styles for a native look and feel.

Advantanges of Using Flutter:

1. Faster Development Time: Hot reload and a single codebase reduce development effort and time
2. Cost Effective: Since developers write one codebase for multiple platforms, it reduces costs associated with maintaining separate teams for iOS & Android.
3. Consistent UI: Flutter renders everything using its own engine, ensuring a uniform look across devices.

Q1 b) Discuss how the Flutter framework differs from traditional approaches? and why it has gained popularity in the developer community?

Ans:- Flutter uses a single codebase for multiple platforms, unli traditional native development that requires seperate code for iOS (Swift) & Android (Kotlin). It does not rely o platform-specific UI components but instead render every using its own skia graphic engine, ensuring consistency. Unlike React Native, which uses a Javascript bridge, Flutter compiles directly to native ARM code, offering better performance. Its hot reload feature allows developers to s changes instantly, making development faster & more eff

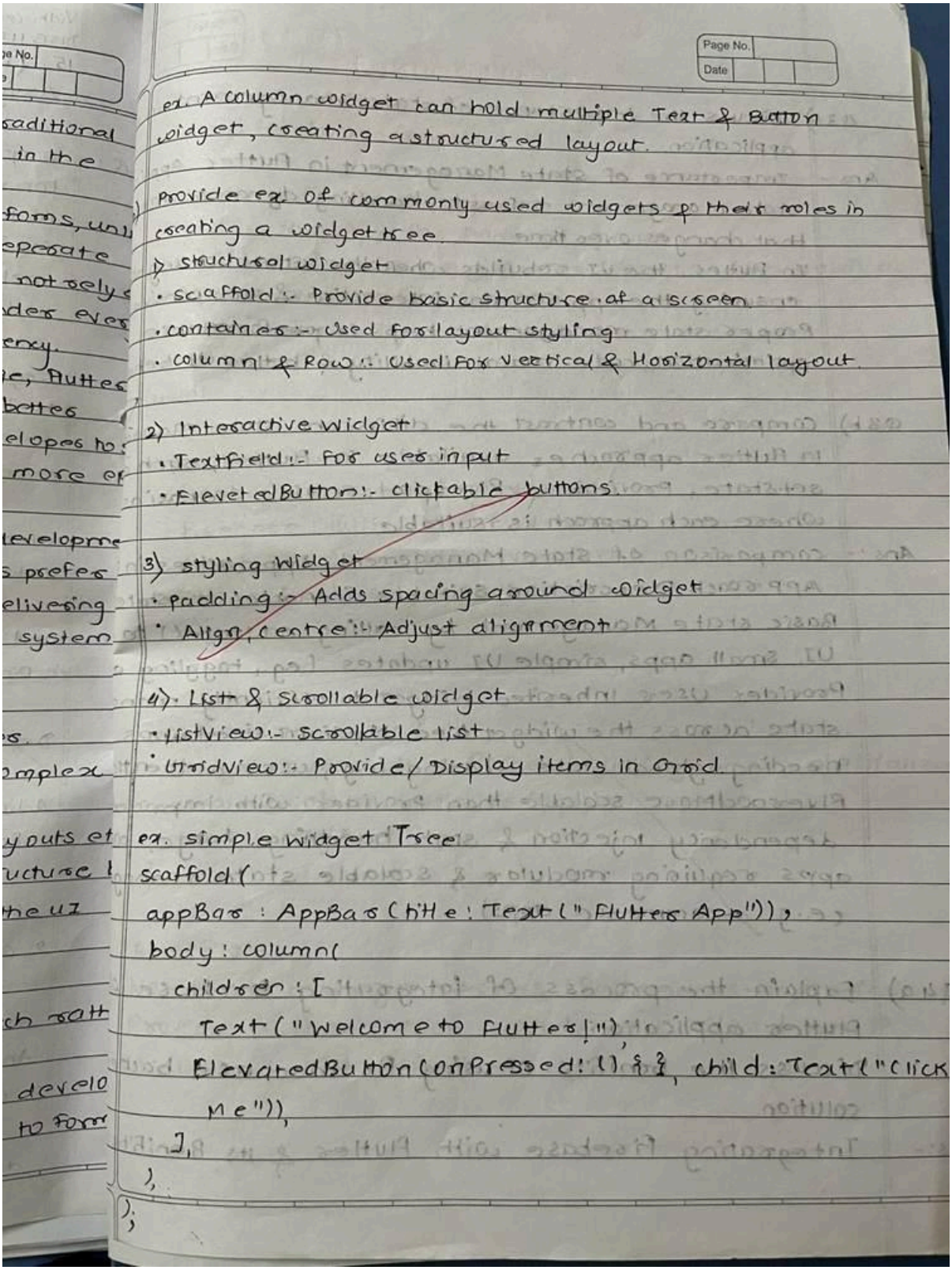
Flutter has gained popularity due to its faster developme cost efficiency, & cross platform support. Business prefer i it reduces development time & costs while delivering high performance apps. Its customizable widget system ensures a smooth, native like experience.

Q2 a) Describe the concept of the widget tree in Flutter. Explain the widget composition is used to build complex UI.

Ans:- In Flutter, everything is a widget (button, text, layouts etc These widgets are arranged in a hierarchical structure kn as the widget tree. The widget tree determines the UI

Widget composition to build complex UI:

- Flutter encourages a composition-based approach rathe than inheritance
- Instead of creating large, monolithic widget, develop build small, reusable widget that are combined to form complex UIs.



ex. A column widget can hold multiple Text & Button widget, creating a structured layout.

1) Provide ex of commonly used widgets & their roles in creating a widget tree.

1) structural widget
- scaffold :- Provide basic structure of a screen
- container :- Used for layout styling
- column & Row :- Used for Vertical & Horizontal layout

2) Interactive Widget
- TextField :- For user input
- ElevatedButton :- clickable buttons

3) styling Widget
- padding :- Adds spacing around widget
- Align, centre :- Adjust alignment

4) List & Scrollable widget
- ListView :- Scrollable list
- GridView :- Provide / Display items in Grid

ex. simple widget Tree

```
scaffold(
appBar : AppBar(title : Text("Flutter App")),
body : column(
children : [
    Text("Welcome to Flutter!"),
ElevatedButton(onPressed: (){}, child: Text("Click
    Me")),
],
),
);
```

Q3 a) Discuss the importance of state management in Flut application.

Ans:- Importance of State Management in Flutter Applicati State Management refers to handling dynamic data that changes over time.
In Flutter, the UI rebuilds when the state changes, ensuring the app remains interactive & responsive. Proper state management helps in performance optimization, code maintainability & better UI behavio

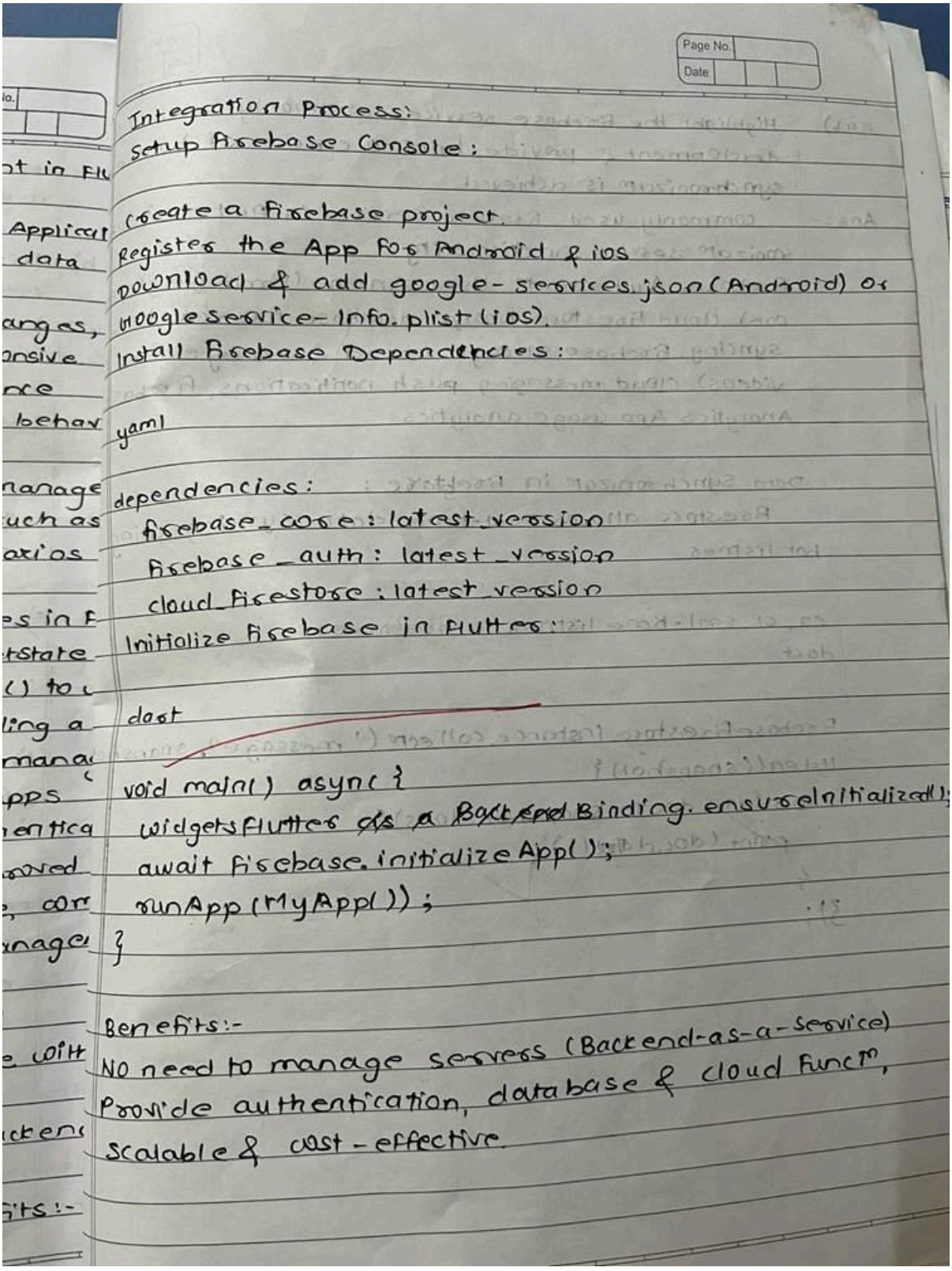
Q3 b) Compare and contrast the different state manager in flutter approaches available in Flutter, such as setstate, Provider & Riverpod. Provide scenarios where each approch is suitable.

Ans :- Comparison of State Management Approaches in Fl

Approach	Description	Suitable Scenarios
setstate	Basic state Management by calling setstate() to up UI.	Small apps, simple UI updates (eg., toggling a s
Provider	Uses Inherited Widget to efficiently manag state across the widget tree.	Medium sized apps needing global state sharing (e.g., user authenticat
Riverpod	More scalable than Provider with improved dependency injection & state handling.	Large, com apps requiring modular & scalable state manager (e.g e-commerce apps)

Q4 a) Explain the process of integrating Firebase with Flutter application.
Discuss the benifits of using Firebase as a backend solution.

s:- Integrating Firebase with Flutter & its Benifits:-



Integration Process:

Setup Firebase Console:

Create a firebase project.

Register the App for Android & iOS

Download & add google-services.json (Android) or GoogleService-Info.plist (iOS).

Install Firebase Dependencies:

yaml

```
dependencies:
  firebase_core: latest_version
  firebase_auth: latest_version
  cloud_firestore: latest_version
```

Initialize Firebase in Flutter:

dart

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}
```

Benefits:-

No need to manage servers (Backend-as-a-service)

Provide authentication, database & cloud Functn,

Scalable & cost-effective.

Page No.
Date

(Q4b) Highlight the firebase services commonly used in flutter development & provide brief overview of how data synchronizam is achieved.

Ans:- commonly used firebase services in flutter & Data Synchronizam service functionality.

Firebase Authenticam Uses sign-in (Email, Google, Facebook) cloud Firestore NoSQL database for real-time data syncing. Firebase storage upload & manage files (images videos) cloud messaging push notifications, firebase Analytics App usage analytics

Data Synchronizam in Firestore:

Firestore allows real-time data syncing using snapshot listners.

eg, of real-time listner in Firestore:

```dart
FirebaseFirestore.instance.collection('message').snapshots().
listen((snapshot){
  for (var doc in snapshot.docs){
    print(doc.data());
  }
});
```