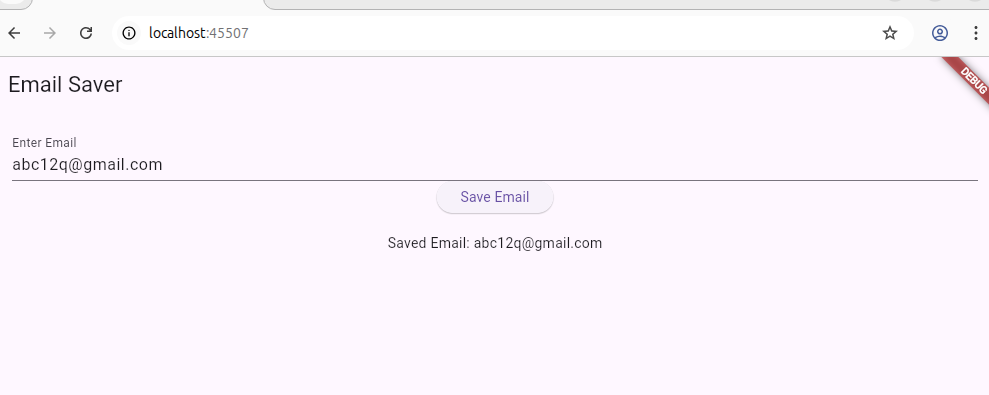
**Task 1: Application using shared preferences**

Create a simple app like this and store one more value of email with help TextField and show this email locally.

Main.dart:

|  |
| --- |
| import 'package:flutter/material.dart';  import 'package:shared\_preferences/shared\_preferences.dart';  void main() {  runApp(EmailApp());}  class EmailApp extends StatelessWidget {  const EmailApp({super.key});  @override  Widget build(BuildContext context) {  return MaterialApp(  home: EmailHomePage(), );}}  class EmailHomePage extends StatefulWidget {  const EmailHomePage({super.key});  @override  \_EmailHomePageState createState() => \_EmailHomePageState();}  class \_EmailHomePageState extends State<EmailHomePage> {  final TextEditingController \_emailController = TextEditingController();  String \_savedEmail = "";  @override  void initState() {  super.initState();  \_loadEmail();}  void \_loadEmail() async {  final prefs = await SharedPreferences.getInstance();  setState(() {  \_savedEmail = prefs.getString('email') ?? '';});}  void \_saveEmail() async {  final prefs = await SharedPreferences.getInstance();  await prefs.setString('email', \_emailController.text);  \_loadEmail(); }  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(title: Text('Email Saver')),  body: Padding(  padding: EdgeInsets.all(20),  child: Column(  children: [  TextField(  controller: \_emailController,  decoration: InputDecoration(labelText: 'Enter Email'), ),  ElevatedButton(onPressed: \_saveEmail, child: Text('Save Email')),  SizedBox(height: 20),  Text('Saved Email: $\_savedEmail'),], ), ),); }} |

**OUTPUT:**



**Task 2: Basics of SQLite**

1. What is SQLite, and why is it suitable for mobile applications?

Ans:SQLite is a lightweight, self-contained SQL database engine. It's ideal for mobile because it stores data locally on the device.

1. Explain the difference between SQLite and other relational databases like MySQL or PostgreSQL.

Ans:a)SQLite: Serverless, file-based, embedded in the app.

b)MySQL/PostgreSQL: Requires server setup, used in large-scale apps with remote access.  
 SQLite is best for small, local databases in mobile apps, while MySQL/PostgreSQL are better for backend systems.

1. Mention three features of SQLite.

Ans:**Three Features of SQLite:**

1. No server needed
2. Small and fast
3. ACID-compliant

**Task 3: SQLite in Flutter**

1,How does Flutter support SQLite?  
Ans: FLutter support SQLite by packages like sqflite to manage local databases.

2.What is the role of the sqflite package in Flutter?

Ans:It provides SQLite access through Dart APIs. It abstracts the SQL operations so you can perform database tasks using Dart/Flutter syntax.

3.Explain the lifecycle of a database in a Flutter app.

Ans:The lifecycle of a database in a Flutter app using sqflite involves the following stages:

* Opening the Database:  
   Use the openDatabase() function to open (or create) the database file. This is usually done when the app starts.
* Creating the Database Schema:  
   If the database is being created for the first time, the onCreate callback is triggered to define the schema (e.g., create tables).
* Performing CRUD Operations:  
   The app can then perform Create, Read, Update, and Delete operations using SQL queries or helper methods provided by sqflite.
* Closing the Database:  
   Once the database is no longer needed (e.g., app exit), it's good practice to close it using the close() method to free resources.