Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
Department of Computer Science and Engineering

Experiment	9&10
Aim	Capstone project with database connectivity
Objective	• Create Registration page • Create database • Store, fetch, delete values from database. • User Authentication from database
Name	Abhijeet Shrimant Jadhav
UCID	2024510021
Class	FY-MCA
Batch	A
Date of Submission	06-05-2025

Technology used	Framework- Flutter, Tool used - Android Studio Programming- Dart Firebase
Task	Create a basic Music player app and store data and music in the database. Fetch and also delete a few music from the database. You can use either firebase database.
Code with proper label	**main.dart :**

```dart
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'screens/user_home_screen.dart';
import 'screens/admin_home_screen.dart';
import 'screens/login_screen.dart'; // You'll need to
create this
import 'screens/wrapper.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Music Player',
```

```
      theme: ThemeData(
        primarySwatch: Colors.indigo,
        visualDensity:
VisualDensity.adaptivePlatformDensity,
      ),
      // initialRoute: '/',
      // routes: {
      //   '/': (context) => const LoginScreen(), //
Simple login screen
      //   '/user': (context) => const UserHomeScreen(),
      //   '/admin': (context) => const
AdminHomeScreen(),
      // },
      home: const Wrapper(),
    );
  }
}
```

song.dart :

```
class Song {
  final String id;
  final String title;
  final String artist;
  final String url;
  final String? albumArt; // Added album art URL

  Song({
    required this.id,
    required this.title,
    required this.artist,
    required this.url,
    this.albumArt,
  });

  factory Song.fromMap(Map<String, dynamic> data, String
docId) {
    return Song(
      id: docId,
      title: data['title'] ?? '',
      artist: data['artist'] ?? '',
      url: data['url'] ?? '',
      albumArt: data['albumArt'],
```

```
    );
  }

  Map<String, dynamic> toMap() {
    return {
      'title': title,
      'artist': artist,
      'url': url,
      'albumArt': albumArt,
    };
  }
}
```

add_edit_song.dart :

```
import 'package:flutter/material.dart';
import '../models/song.dart';
import '../services/firebase_service.dart';

class AddEditSongScreen extends StatefulWidget {
  final Song? song;

  const AddEditSongScreen({super.key, this.song});

  @override
  _AddEditSongScreenState createState() =>
_AddEditSongScreenState();
}

class _AddEditSongScreenState extends
State<AddEditSongScreen> {
  final _formKey = GlobalKey<FormState>();
  late String _title;
  late String _artist;
  late String _url;
  late String? _albumArt;

  final FirebaseService _firebaseService =
FirebaseService();

  @override
  void initState() {
    super.initState();
```

```
    _title = widget.song?.title ?? '';
    _artist = widget.song?.artist ?? '';
    _url = widget.song?.url ?? '';
    _albumArt = widget.song?.albumArt;
  }

  void _saveSong() async {
    if (_formKey.currentState!.validate()) {
      _formKey.currentState!.save();
      final newSong = Song(
        id: widget.song?.id ?? '',
        title: _title,
        artist: _artist,
        url: _url,
        albumArt: _albumArt,
      );

      if (widget.song == null) {
        await _firebaseService.addSong(newSong);
      } else {
        await _firebaseService.updateSong(newSong);
      }

      Navigator.pop(context);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.song == null ? 'Add Song' :
'Edit Song'),
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(20),
        child: Form(
          key: _formKey,
          child: Column(
            crossAxisAlignment:
CrossAxisAlignment.stretch,
            children: [
              const Text(
```

```
                "Song Details",
                style: TextStyle(fontSize: 20,
fontWeight: FontWeight.bold),
              ),
              const SizedBox(height: 16),
              TextFormField(
                initialValue: _title,
                decoration: const InputDecoration(
                  labelText: 'Title',
                  border: OutlineInputBorder(),
                ),
                onSaved: (value) => _title = value!,
                validator: (value) => value!.isEmpty ?
'Required' : null,
              ),
              const SizedBox(height: 16),
              TextFormField(
                initialValue: _artist,
                decoration: const InputDecoration(
                  labelText: 'Artist',
                  border: OutlineInputBorder(),
                ),
                onSaved: (value) => _artist = value!,
                validator: (value) => value!.isEmpty ?
'Required' : null,
              ),
              const SizedBox(height: 16),
              TextFormField(
                initialValue: _url,
                decoration: const InputDecoration(
                  labelText: 'Audio URL',
                  border: OutlineInputBorder(),
                ),
                onSaved: (value) => _url = value!,
                validator: (value) => value!.isEmpty ?
'Required' : null,
              ),
              const SizedBox(height: 16),
              TextFormField(
                initialValue: _albumArt,
                decoration: const InputDecoration(
                  labelText: 'Album Art URL (optional)',
                  border: OutlineInputBorder(),
```

```
                ),
                onSaved: (value) => _albumArt = value,
              ),
              const SizedBox(height: 24),
              ElevatedButton.icon(
                onPressed: _saveSong,
                icon: const Icon(Icons.save),
                label: const Text('Save Song'),
                style: ElevatedButton.styleFrom(
                  padding: const
EdgeInsets.symmetric(vertical: 14),
                  textStyle: const TextStyle(fontSize:
16),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

admin_home_screen.dart :

```
import 'package:flutter/material.dart';
import '../models/song.dart';
import '../services/firebase_service.dart';
import '../services/auth_service.dart';
import 'add_edit_song_screen.dart';

class AdminHomeScreen extends StatefulWidget {
  const AdminHomeScreen({super.key});

  @override
  _AdminHomeScreenState createState() =>
_AdminHomeScreenState();
}

class _AdminHomeScreenState extends
State<AdminHomeScreen> {
  final FirebaseService _firebaseService =
FirebaseService();
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Admin Panel - Manage Songs'),
        actions: [
          IconButton(
            icon: const Icon(Icons.logout),
            tooltip: 'Logout',
            onPressed: () async {
              await AuthService().signOut();
            },
          ),
        ],
      ),
      body: StreamBuilder<List<Song>>(
        stream: _firebaseService.songsStream,
        builder: (context, snapshot) {
          if (snapshot.hasError) {
            return Center(child: Text('Error:
${snapshot.error}'));
          }

          if (snapshot.connectionState ==
ConnectionState.waiting) {
            return const Center(child:
CircularProgressIndicator());
          }

          final songs = snapshot.data!;

          if (songs.isEmpty) {
            return const Center(child: Text('No songs
available'));
          }

          return ListView.builder(
            padding: const EdgeInsets.all(12),
            itemCount: songs.length,
            itemBuilder: (context, index) {
              final song = songs[index];
              return Card(
```

```
                elevation: 4,
                shape: RoundedRectangleBorder(
                  borderRadius:
BorderRadius.circular(12),
                ),
                margin: const
EdgeInsets.symmetric(vertical: 8),
                child: ListTile(
                  contentPadding:
                      const
EdgeInsets.symmetric(horizontal: 16, vertical: 8),
                  leading: ClipRRect(
                    borderRadius:
BorderRadius.circular(8),
                    child: song.albumArt != null
                        ? Image.network(song.albumArt!,
                            width: 60, height: 60, fit:
BoxFit.cover)
                        : const Icon(Icons.music_note,
size: 40),
                  ),
                  title: Text(
                    song.title,
                    style: const TextStyle(
                      fontWeight: FontWeight.bold,
                      fontSize: 16,
                    ),
                  ),
                  subtitle: Text(song.artist),
                  trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      IconButton(
                        icon: const Icon(Icons.edit,
color: Colors.blue),
                        onPressed: () {
                          Navigator.push(
                            context,
                            MaterialPageRoute(
                              builder: (_) =>
AddEditSongScreen(song: song),
                            ),
                          );
```

```
                        },
                      ),
                      IconButton(
                        icon: const Icon(Icons.delete,
color: Colors.red),
                        onPressed: () =>
_firebaseService.deleteSong(song.id),
                      ),
                    ],
                  ),
                ),
              );
            },
          );
        },
      ),
      floatingActionButton:
FloatingActionButton.extended(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (_) => const AddEditSongScreen(),
            ),
          );
        },
        label: const Text('Add Song'),
        icon: const Icon(Icons.add),
      ),
    );
  }
}
```

home_screen.dart :

```
import 'package:flutter/material.dart';
import '../models/song.dart';
import '../services/firebase_service.dart';
import 'now_playing_screen.dart';

class HomeScreen extends StatelessWidget {
  final FirebaseService _firebaseService =
FirebaseService();
```

```
  HomeScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Music Player'),
      ),
      body: StreamBuilder<List<Song>>(
        stream: _firebaseService.songsStream,
        builder: (context, snapshot) {
          if (snapshot.hasError) {
            return Center(child: Text('Error:
${snapshot.error}'));
          }

          if (snapshot.connectionState ==
ConnectionState.waiting) {
            return const Center(child:
CircularProgressIndicator());
          }

          final songs = snapshot.data!;

          return ListView.builder(
            itemCount: songs.length,
            itemBuilder: (context, index) {
              final song = songs[index];
              return ListTile(
                title: Text(song.title),
                subtitle: Text(song.artist),
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (_) =>
NowPlayingScreen(song: song),
                    ),
                  );
                },
              );
            },
```

```
          );
        },
      ),
    );
  }
}
```

login_screen.dart :

```dart
import 'package:flutter/material.dart';
import '../services/auth_service.dart';
import 'package:firebase_auth/firebase_auth.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({super.key});

  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final AuthService _auth = AuthService();
  final _formKey = GlobalKey<FormState>();
  bool _isLogin = true;
  bool _isAdminLogin = false;

  // Form fields
  String _email = '';
  String _password = '';
  String _firstName = '';
  String _confirmPassword = '';

  // Error message
  String _errorMessage = '';

  void _toggleLogin() {
    setState(() {
      _isLogin = !_isLogin;
      _errorMessage = '';
    });
  }

  void _toggleAdminLogin() {
```

```
    setState(() {
      _isAdminLogin = !_isAdminLogin;
      _errorMessage = '';
    });
  }

  Future<void> _submit() async {
    if (_formKey.currentState!.validate()) {
      _formKey.currentState!.save();

      try {
        User? user;
        if (_isAdminLogin) {
          user = await _auth.adminLogin(_email,
_password);
          if (user != null) {
            Navigator.pushReplacementNamed(context,
'/admin');
          } else {
            setState(() {
              _errorMessage = 'Invalid admin
credentials';
            });
          }
        } else if (_isLogin) {
          user = await _auth.userLogin(_email,
_password);
          if (user != null) {
            Navigator.pushReplacementNamed(context,
'/user');
          } else {
            setState(() {
              _errorMessage = 'Invalid email or
password';
            });
          }
        } else {
          if (_password != _confirmPassword) {
            setState(() {
              _errorMessage = 'Passwords do not match';
            });
            return;
          }
```

```
          user = await _auth.registerUser(_email,
_password, _firstName);
          if (user != null) {
            Navigator.pushReplacementNamed(context,
'/user');
          }
        }
      } catch (e) {
        setState(() {
          _errorMessage = e.toString();
        });
      }
    }
  }

  InputDecoration _buildInputDecoration(String label,
IconData icon) {
    return InputDecoration(
      labelText: label,
      prefixIcon: Icon(icon),
      border: OutlineInputBorder(borderRadius:
BorderRadius.circular(10)),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      body: Center(
        child: SingleChildScrollView(
          padding: const EdgeInsets.symmetric(horizontal:
24.0, vertical: 16),
          child: Form(
            key: _formKey,
            child: Card(
              elevation: 6,
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(16),
              ),
              margin: const
EdgeInsets.symmetric(horizontal: 10),
```

```
              child: Padding(
                padding: const EdgeInsets.all(24.0),
                child: Column(
                  mainAxisSize: MainAxisSize.min,
                  children: [
                    const Icon(Icons.music_note, size:
60, color: Colors.blue),
                    const SizedBox(height: 10),
                    const Text(
                      'Music Player',
                      style:
                          TextStyle(fontSize: 26,
fontWeight: FontWeight.bold),
                    ),
                    const SizedBox(height: 30),
                    if (!_isAdminLogin && !_isLogin) ...[
                      TextFormField(
                        decoration:
                            _buildInputDecoration('First
Name', Icons.person),
                        validator: (value) =>
                            value!.isEmpty ? 'Please
enter your name' : null,
                        onSaved: (value) => _firstName =
value!,
                      ),
                      const SizedBox(height: 16),
                    ],
                    TextFormField(
                      decoration:
_buildInputDecoration('Email', Icons.email),
                      keyboardType:
TextInputType.emailAddress,
                      validator: (value) =>
                          value!.isEmpty ? 'Please enter
your email' : null,
                      onSaved: (value) => _email =
value!,
                    ),
                    const SizedBox(height: 16),
                    TextFormField(
                      decoration:
_buildInputDecoration('Password', Icons.lock),
```

```
                      obscureText: true,
                      validator: (value) => value!.length
< 6
                          ? 'Password must be at least 6
characters'
                          : null,
                      onSaved: (value) => _password =
value!,
                    ),
                    if (!_isLogin && !_isAdminLogin) ...[
                      const SizedBox(height: 16),
                      TextFormField(
                        decoration:
_buildInputDecoration(
                            'Confirm Password',
Icons.lock_outline),
                        obscureText: true,
                        validator: (value) =>
value!.isEmpty
                            ? 'Please confirm your
password'
                            : null,
                        onSaved: (value) =>
_confirmPassword = value!,
                      ),
                    ],
                    if (_errorMessage.isNotEmpty) ...[
                      const SizedBox(height: 16),
                      Text(
                        _errorMessage,
                        style: const TextStyle(color:
Colors.red),
                      ),
                    ],
                    const SizedBox(height: 24),
                    SizedBox(
                      width: double.infinity,
                      child: ElevatedButton(
                        onPressed: _submit,
                        style: ElevatedButton.styleFrom(
                          padding: const
EdgeInsets.symmetric(vertical: 14),
                          shape: RoundedRectangleBorder(
```

```
                              borderRadius:
BorderRadius.circular(10),
                            ),
                            backgroundColor:
Colors.blueAccent,
                          ),
                          child: Text(
                            _isLogin ? 'Login' :
'Register',
                            style: const
TextStyle(fontSize: 16),
                          ),
                        ),
                      ),
                      const SizedBox(height: 12),
                      TextButton(
                        onPressed: _toggleLogin,
                        child: Text(
                          _isLogin
                              ? 'Need an account? Register'
                              : 'Already have an account?
Login',
                          style: const TextStyle(color:
Colors.black87),
                        ),
                      ),
                      if (!_isAdminLogin)
                        TextButton(
                          onPressed: _toggleAdminLogin,
                          child: const Text(
                            'Admin Login',
                            style: TextStyle(fontWeight:
FontWeight.bold),
                          ),
                        )
                      else
                        TextButton(
                          onPressed: _toggleAdminLogin,
                          child: const Text(
                            'User Login',
                            style: TextStyle(fontWeight:
FontWeight.bold),
                          ),
```

```
                        ),
                    ],
                ),
            ),
        ),
    ),
  ),
 );
 }
}
```

auth_service.dart :

```
import 'package:firebase_auth/firebase_auth.dart';

class AuthService {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  // Get the current user
  User? get currentUser => _auth.currentUser;

  // Admin login
  Future<User?> adminLogin(String email, String password)
async {
    try {
      if (email == 'admin@gmail.com' && password ==
'admin123') {
        UserCredential result = await
_auth.signInWithEmailAndPassword(
          email: email,
          password: password,
        );
        return result.user;
      }
      return null;
    } catch (e) {
      print("Admin login error: $e");
      return null;
    }
  }

  // User registration
```

```
  Future<User?> registerUser(
      String email, String password, String firstName)
async {
    try {
      UserCredential result = await
_auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );

      // Update user display name
      await result.user?.updateDisplayName(firstName);
      return result.user;
    } catch (e) {
      print("Registration error: $e");
      return null;
    }
  }

  // User login
  Future<User?> userLogin(String email, String password)
async {
    try {
      UserCredential result = await
_auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return result.user;
    } catch (e) {
      print("Login error: $e");
      return null;
    }
  }

  // Sign out
  Future<void> signOut() async {
    await _auth.signOut();
  }

  // Auth state changes
  Stream<User?> get user {
    return _auth.authStateChanges();
```

```
  }
}
```

firebase_service.dart :

```
import 'package:firebase_auth/firebase_auth.dart';
class AuthService {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  // Get the current user
  User? get currentUser => _auth.currentUser;
  // Admin login
  Future<User?> adminLogin(String email, String password)
async {
    try {
      if (email == 'admin@gmail.com' && password ==
'admin123') {
        UserCredential result = await
_auth.signInWithEmailAndPassword(
          email: email,
          password: password,
        );
        return result.user;
      }
      return null;
    } catch (e) {
      print("Admin login error: $e");
      return null;
    }
  }
  // User registration
  Future<User?> registerUser(
      String email, String password, String firstName)
async {
    try {
      UserCredential result = await
_auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );

      // Update user display name
      await result.user?.updateDisplayName(firstName);
      return result.user;
    } catch (e) {
```

```
      print("Registration error: $e");
      return null;
    }
  }
  // User login
  Future<User?> userLogin(String email, String password)
async {
    try {
      UserCredential result = await
_auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return result.user;
    } catch (e) {
      print("Login error: $e");
      return null;
    }
  }
  // Sign out
  Future<void> signOut() async {
    await _auth.signOut();
  }
  // Auth state changes
  Stream<User?> get user {
    return _auth.authStateChanges();
  }
}
```

music_player_screen,dart :

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:just_audio/just_audio.dart';
import 'package:audio_session/audio_session.dart';
import 'package:rxdart/rxdart.dart'; // For combining
streams
class MusicPlayerScreen extends StatefulWidget {
  @override
  _MusicPlayerScreenState createState() =>
_MusicPlayerScreenState();
}
class _MusicPlayerScreenState extends
State<MusicPlayerScreen> {
```

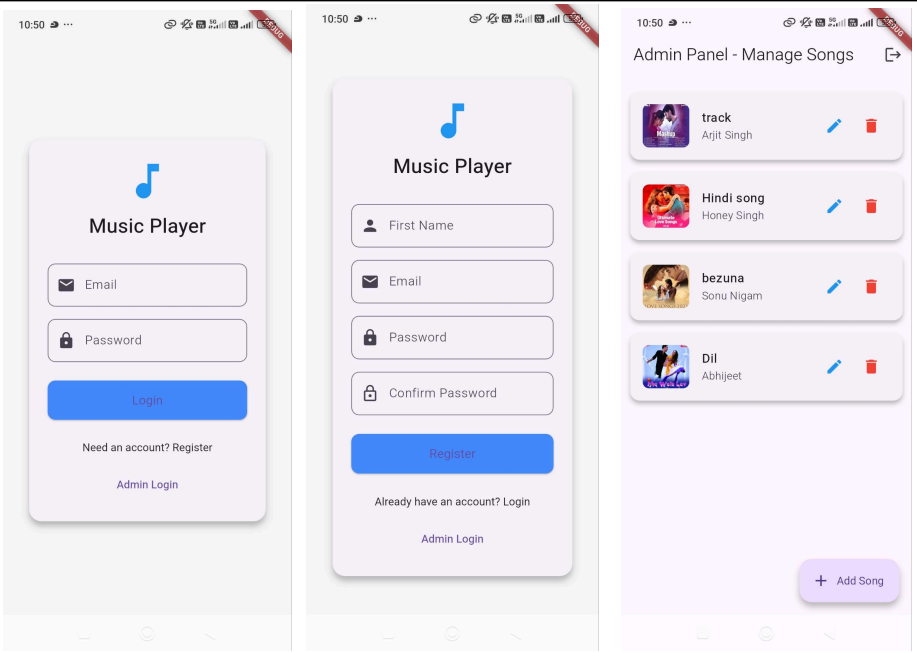
```
  final AudioPlayer _player = AudioPlayer();
  List<DocumentSnapshot> _songs = [];
  int _currentIndex = -1;
  @override
  void initState() {
    super.initState();
    _initAudioSession();
  }
  Future<void> _initAudioSession() async {
    final session = await AudioSession.instance;
    await
session.configure(AudioSessionConfiguration.music());
  }
  Future<void> _playSong(int index) async {
    if (index < 0 || index >= _songs.length) return;
    final song = _songs[index];
    final url = song['url'];
    try {
      await _player.setUrl(url);
      await _player.play();
      setState(() {
        _currentIndex = index;
      });
    } catch (e) {
      print("Error playing song: $e");
    }
  }
  void _pauseOrResume() {
    if (_player.playing) {
      _player.pause();
    } else {
      _player.play();
    }
    setState(() {});
  }
  void _stop() {
    _player.stop();
    setState(() {});
  }
  void _nextSong() {
    if (_currentIndex + 1 < _songs.length) {
      _playSong(_currentIndex + 1);
    }
```

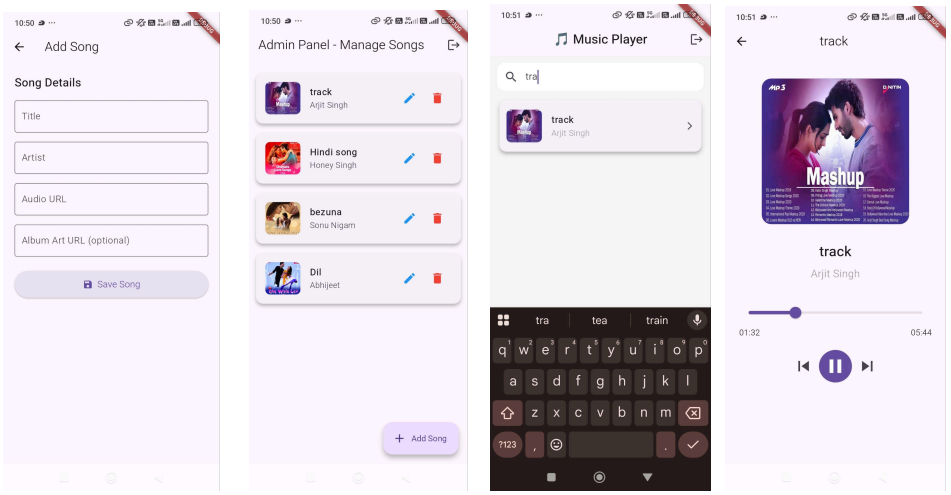
```
  }
  void _previousSong() {
    if (_currentIndex - 1 >= 0) {
      _playSong(_currentIndex - 1);
    }
  }
  Stream<DurationState> get _durationStateStream =>
      Rx.combineLatest2<Duration, Duration?,
DurationState>(
        _player.positionStream,
        _player.durationStream,
        (position, duration) =>
            DurationState(position: position, total:
duration ?? Duration.zero),
      );
  @override
  void dispose() {
    _player.dispose();
    super.dispose();
  }
  Widget _buildControls() {
    return Column(
      children: [
        StreamBuilder<DurationState>(
          stream: _durationStateStream,
          builder: (context, snapshot) {
            final durationState = snapshot.data;
            final position = durationState?.position ??
Duration.zero;
            final total = durationState?.total ??
Duration.zero;

            return Slider(
              min: 0,
              max: total.inMilliseconds.toDouble(),
              value: position.inMilliseconds
                  .toDouble()
                  .clamp(0.0,
total.inMilliseconds.toDouble()),
              onChanged: (value) {
                _player.seek(Duration(milliseconds:
value.toInt()));
              },
```

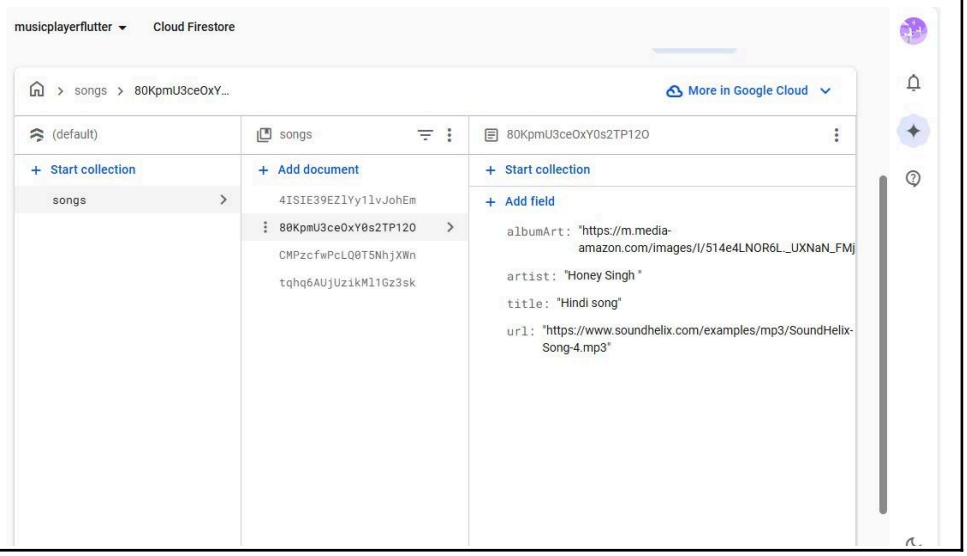
```
            );
          },
        ),
        Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            IconButton(
                icon: Icon(Icons.skip_previous),
onPressed: _previousSong),
            IconButton(
              icon: Icon(_player.playing ? Icons.pause :
Icons.play_arrow),
              onPressed: _pauseOrResume,
              iconSize: 36,
            ),
            IconButton(icon: Icon(Icons.stop), onPressed:
_stop),
            IconButton(icon: Icon(Icons.skip_next),
onPressed: _nextSong),
          ],
        ),
      ],
    );
  }
  Widget _buildSongTile(DocumentSnapshot songDoc, int
index) {
    final title = songDoc['title'];
    return ListTile(
      leading: Icon(Icons.music_note),
      title: Text(title),
      trailing: IconButton(
        icon: Icon(Icons.play_arrow),
        onPressed: () => _playSong(index),
      ),
    );
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Music Player'),
      ),
      body: Column(
```

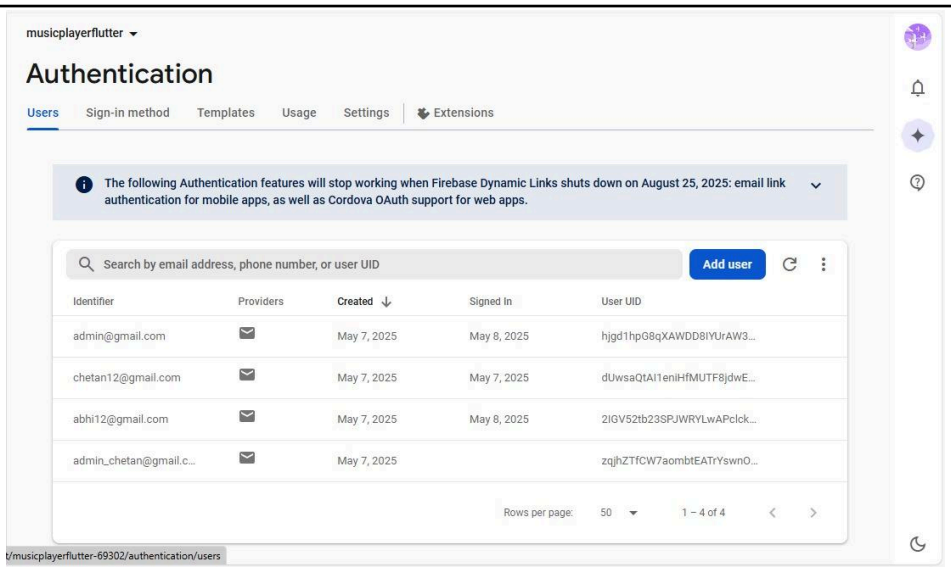
```
        children: [
          if (_currentIndex != -1) _buildControls(),
          Expanded(
            child: StreamBuilder<QuerySnapshot>(
              stream:

FirebaseFirestore.instance.collection('songs').snapshots(
),
              builder: (context, snapshot) {
                if (!snapshot.hasData) return
CircularProgressIndicator();
                _songs = snapshot.data!.docs;
                return ListView.builder(
                  itemCount: _songs.length,
                  itemBuilder: (context, index) =>
                      _buildSongTile(_songs[index],
index),
                );
              },
            ),
          ),
        ],
      ),
    );
  }
}

class DurationState {
  final Duration position;
  final Duration total;

  DurationState({required this.position, required
this.total});
}
```

Screenshots









Conclusion	In this session, I learned, How to connect Flutter apps with Firebase for authentication and database storage. How to initialize and use Firestore as a cloud-hosted NoSQL database. How to perform CRUD operations (Create, Read, Update, Delete) easily using Firestore collections and documents.