Week 6: Create an E-Book App Using Flutter

AIM:

- Create an E-Book app with sections for book categories and a simple book reader UI.
- Implement navigation between categories and book details.

CODE:

main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'src/login_signup_screen.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: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const WelcomeScreen(),
    );
  }
}

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

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

class _WelcomeScreenState extends State<WelcomeScreen> with SingleTickerProviderStateMixin {
  double _opacity = 0.0;
  late AnimationController _animationController;
  late Animation<double> _scaleAnimation;
  late Animation<Offset> _slideAnimation;
```

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

  _animationController = AnimationController(
    duration: const Duration(seconds: 2),
    vsync: this,
  );

  _scaleAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(
    CurvedAnimation(parent: _animationController, curve: Curves.easeInOut),
  );

  _slideAnimation = Tween<Offset>(begin: const Offset(0.0, 1.0), end: Offset.zero).animate(
    CurvedAnimation(parent: _animationController, curve: Curves.easeInOut),
  );

  Future.delayed(const Duration(seconds: 1), () {
    setState(() {
      _opacity = 1.0;
    });
    _animationController.forward();
  });

  // Show logo for 8 seconds
  Future.delayed(const Duration(seconds: 8), () {
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const LoginSignupScreen()),
    );
  });
}

@override
void dispose() {
  _animationController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: AnimatedOpacity(
        opacity: _opacity,
        duration: const Duration(seconds: 1),
        child: ScaleTransition(
          scale: _scaleAnimation,
          child: SlideTransition(
            position: _slideAnimation,
            child: Image.asset(
              "assets/load.gif", // Ensure this GIF exists in the assets folder
              width: 300,
              height: 300,
              fit: BoxFit.cover,
          ),
        ),
      ),
    ),
  ),
 );
 }
}
```

login_signup_page.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'home_screen.dart';
import 'package:fluttertoast/fluttertoast.dart';

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

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

class _LoginSignupScreenState extends State<LoginSignupScreen> {
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  bool isSignUp = false;
  bool isLoading = false; // Add loading state

  final FirebaseAuth _auth = FirebaseAuth.instance;

  // Handle Google Sign In
  Future<User?> signInWithGoogle() async {
    setState(() {
      isLoading = true; // Show loading indicator
    });

    final GoogleSignIn googleSignIn = GoogleSignIn();
    final GoogleSignInAccount? googleUser = await googleSignIn.signIn();
    if (googleUser == null) {
      setState(() {
        isLoading = false; // Hide loading indicator
      });
      return null;
    }
    final GoogleSignInAuthentication googleAuth = await googleUser.authentication;

    final OAuthCredential credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    final UserCredential userCredential = await _auth.signInWithCredential(credential);

    setState(() {
      isLoading = false; // Hide loading indicator
    });

    return userCredential.user;
  }

  // Show alert dialog for errors
  void showAlert(String message) {
    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: const Text('Error'),
        content: Text(message),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.pop(context);
            },
            child: const Text('OK'),
          ),
        ],
      ),
    );
  }

  // Show Toast message for authentication failure
  void showToast(String message) {
    Fluttertoast.showToast(msg: message, toastLength: Toast.LENGTH_SHORT);
  }

  // Email validation
  bool isValidEmail(String email) {
    final regex = RegExp(r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$');
    return regex.hasMatch(email);
  }

  // Handle email/password sign-up
```

```
 Future<void> signUpWithEmailPassword() async {
   setState(() {
     isLoading = true; // Show loading indicator
   });

   if (_emailController.text.isEmpty || _passwordController.text.isEmpty) {
     showAlert('Please fill in both email and password');
     setState(() {
       isLoading = false; // Hide loading indicator
     });
     return;
   }

   if (!isValidEmail(_emailController.text)) {
     showAlert('Please enter a valid email address');
     setState(() {
       isLoading = false; // Hide loading indicator
     });
     return;
   }

   if (_passwordController.text.length < 6) {
     showAlert('Password must be at least 6 characters');
     setState(() {
       isLoading = false; // Hide loading indicator
     });
     return;
   }

   try {
     await _auth.createUserWithEmailAndPassword(
       email: _emailController.text.trim(),
       password: _passwordController.text.trim(),
     );
     setState(() {
       isLoading = false; // Hide loading indicator
     });
     Navigator.pushReplacement(
       context,
       MaterialPageRoute(builder: (context) => const HomeScreen()),
     );
   } on FirebaseAuthException catch (e) {
     setState(() {
       isLoading = false; // Hide loading indicator
     });
     if (e.code == 'weak-password') {
       showToast('The password is too weak.');
```

```
    } else if (e.code == 'email-already-in-use') {
      showToast('An account already exists for that email.');
    } else {
      showToast('Authentication failed! Please try again.');
    }
  } catch (e) {
    setState(() {
      isLoading = false; // Hide loading indicator
    });
    showToast('Something went wrong. Please try again.');
  }
}

// Handle email/password login
Future<void> logInWithEmailPassword() async {
  setState(() {
    isLoading = true; // Show loading indicator
  });

  if (_emailController.text.isEmpty || _passwordController.text.isEmpty) {
    showAlert('Please fill in both email and password');
    setState(() {
      isLoading = false; // Hide loading indicator
    });
    return;
  }

  if (!isValidEmail(_emailController.text)) {
    showAlert('Please enter a valid email address');
    setState(() {
      isLoading = false; // Hide loading indicator
    });
    return;
  }

  if (_passwordController.text.length < 6) {
    showAlert('Password must be at least 6 characters');
    setState(() {
      isLoading = false; // Hide loading indicator
    });
    return;
  }

  try {
    await _auth.signInWithEmailAndPassword(
      email: _emailController.text.trim(),
      password: _passwordController.text.trim(),
    );
    setState(() {
      isLoading = false; // Hide loading indicator
    });
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const HomeScreen()),
    );
  } on FirebaseAuthException catch (e) {
    setState(() {
      isLoading = false; // Hide loading indicator
    });
    if (e.code == 'user-not-found') {
      showToast('No user found for that email.');
    } else if (e.code == 'wrong-password') {
      showToast('Incorrect password.');
    } else {
      showToast('Invalid email or password. Please try again.');
    }
  } catch (e) {
    setState(() {
      isLoading = false; // Hide loading indicator
    });
    showToast('Something went wrong. Please try again.');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.blue,
      title: const Text('E-Book App'),
      centerTitle: true,
      elevation: 0,
    ),
    body: isLoading
        ? Center(
            child: CircularProgressIndicator(
              color: Colors.blue,  // Standard blue
               // Customize the color if needed
            ),
          )
        : SingleChildScrollView(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              children: <Widget>[
// Branding Image or Logo
Center(
  child: Image.asset(
    'assets/images/logo.png', // Replace with your logo
    height: 250,
    width: 350,
  ),
),
const SizedBox(height: 20),

// AnimatedSwitcher for flip animation between Login and Signup
AnimatedSwitcher(
  duration: const Duration(milliseconds: 600),
  transitionBuilder: (Widget child, Animation<double> animation) {
    final rotate = Tween(begin: 0.0, end: 1.0).animate(animation);
    return RotationTransition(
      turns: rotate,
      child: child,
    );
  },
  child: isSignUp ? buildSignUpForm() : buildLoginForm(),
),
const SizedBox(height: 20),

// Toggle between Sign Up and Login with more spacing
TextButton(
  onPressed: () {
    setState(() {
      isSignUp = !isSignUp; // Toggle between SignUp and Login
    });
  },
  child: Text(
    isSignUp
        ? 'Already have an account? Login'
        : 'Don\'t have an account? Sign Up',
    style: const TextStyle(fontSize: 16),
  ),
),
const SizedBox(height: 20),

// Google Sign-In Button
ElevatedButton.icon(
  onPressed: () async {
    User? user = await signInWithGoogle();
    if (user != null) {
      Navigator.pushReplacement(
        context,
                MaterialPageRoute(builder: (context) => const HomeScreen()),
              );
            } else {
              showToast('Google sign-in failed.');
            }
          },
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.blue, // Google blue color
            padding: const EdgeInsets.symmetric(vertical: 15),
            minimumSize: const Size(double.infinity, 50),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(15),
            ),
          ),
          icon: const Icon(Icons.g_mobiledata, color: Colors.white),
          label: const Text(
            'Continue with Google',
            style: TextStyle(fontSize: 18, color: Colors.white),
          ),
        ),
        const SizedBox(height: 30),

        // Optional: Add Footer or Disclaimer with more padding
        const Divider(),
        const Text(
          'By signing up, you agree to our Terms and Conditions',
          style: TextStyle(fontSize: 12, color: Colors.grey),
        ),
      ],
    ),
  ),
 );
}

Widget buildLoginForm() {
  return Column(
    key: const ValueKey('login'),
    children: [
      TextField(
        controller: _emailController,
        decoration: InputDecoration(
          prefixIcon: const Icon(Icons.email, color: Colors.blue),
          labelText: 'Email Address',
          hintText: 'Enter your email',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(15),
          ),
          filled: true,
          fillColor: Colors.white,
        ),
      ),
      const SizedBox(height: 15),

      TextField(
        controller: _passwordController,
        obscureText: true,
        decoration: InputDecoration(
          prefixIcon: const Icon(Icons.lock, color: Colors.blue),
          labelText: 'Password',
          hintText: 'Enter your password',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(15),
          ),
          filled: true,
          fillColor: Colors.white,
        ),
      ),
      const SizedBox(height: 25),

      ElevatedButton(
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.blue,
          foregroundColor: Colors.white,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(15),
          ),
          padding: const EdgeInsets.symmetric(vertical: 15),
          minimumSize: const Size(double.infinity, 50),
        ),
        onPressed: logInWithEmailPassword,
        child: const Text(
          'Login',
          style: TextStyle(fontSize: 18),
        ),
      ),
    ],
  );
}

Widget buildSignUpForm() {
  return Column(
    key: const ValueKey('signup'),
    children: [
      TextField(
        controller: _emailController,
        decoration: InputDecoration(
          prefixIcon: const Icon(Icons.email, color: Colors.blue),
          labelText: 'Email Address',
          hintText: 'Enter your email',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(15),
          ),
          filled: true,
          fillColor: Colors.white,
        ),
      ),
      const SizedBox(height: 15),

      TextField(
        controller: _passwordController,
        obscureText: true,
        decoration: InputDecoration(
          prefixIcon: const Icon(Icons.lock, color: Colors.blue),
          labelText: 'Password',
          hintText: 'Enter your password',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(15),
          ),
          filled: true,
          fillColor: Colors.white,
        ),
      ),
      const SizedBox(height: 25),

      ElevatedButton(
        style: ElevatedButton.styleFrom(
          backgroundColor: const Color.fromARGB(255, 59, 93, 231),
          foregroundColor: Colors.white,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(15),
          ),
          padding: const EdgeInsets.symmetric(vertical: 15),
          minimumSize: const Size(double.infinity, 50),
        ),
        onPressed: signUpWithEmailPassword,
        child: const Text(
          'Sign Up',
          style: TextStyle(fontSize: 18),
        ),
      ),
    ],
  );
}
}
```

home_page.dart

```
import 'package:flutter/material.dart';
import 'pdf_viewer_screen.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'login_signup_screen.dart';

class Book {
  final String name;
  final String author;
  final String publishDate;
  final String pdfPath;
  final String imagePath;

  Book(this.name, this.author, this.publishDate, this.pdfPath, this.imagePath);
}

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

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  final List<Book> _books = [
    Book("Flutter for Beginners", "John Doe", "2022", "assets/books/book1.pdf",
"assets/images/book1.jpg"),
    Book("Mastering Dart", "Jane Smith", "2021", "assets/books/book2.pdf",
"assets/images/book2.jpg"),
    Book("Advanced Flutter", "Mike Johnson", "2023", "assets/books/book3.pdf",
"assets/images/book3.jpg"),
    Book("Flutter Cookbook", "Alice Brown", "2020", "assets/books/book4.pdf",
"assets/images/book4.jpg"),
    Book("Dart in Depth", "Charlie White", "2019", "assets/books/book5.pdf",
"assets/images/book5.jpg"),
    Book("Flutter UI Design", "David Black", "2024", "assets/books/book6.pdf",
"assets/images/book6.jpg"),
  ];
  bool _isGridView = false;

  void _logout() async {
  await FirebaseAuth.instance.signOut();
  await GoogleSignIn().signOut(); // Sign out from Google if used

  // Navigate to the login/signup screen and remove all previous screens from the stack
  Navigator.pushAndRemoveUntil(
    context,
    MaterialPageRoute(builder: (context) => const LoginSignupScreen()),
    (route) => false, // This removes all previous routes from the stack
  );
}



  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.blue.shade900,
        title: const Text(
          'E-Book Store',
          style: TextStyle(fontWeight: FontWeight.bold, color: Colors.white),
        ),
        actions: [
          IconButton(
            icon: Icon(_isGridView ? Icons.list : Icons.grid_view, color: Colors.white),
            onPressed: () {
              setState(() {
                _isGridView = !_isGridView;
              });
            },
          ),
          IconButton(
  icon: const Icon(Icons.logout, color: Colors.white),
  onPressed: () => _logout(), // Explicitly passing context
),

        ],
      ),
      body: _isGridView ? _buildGridView() : _buildListView(),
      floatingActionButton: FloatingActionButton(
        backgroundColor: Colors.blue.shade900,
        onPressed: () {
          setState(() {
            _isGridView = !_isGridView;
          });
        },
        child: Icon(_isGridView ? Icons.list : Icons.grid_view, color: Colors.white),
      ),
    );
  }

  Widget _buildListView() {
    return ListView.builder(
      itemCount: _books.length,
      itemBuilder: (context, index) {
        final book = _books[index];
        return Card(
          color: Colors.blue.shade100,
          margin: const EdgeInsets.all(8.0),
          elevation: 5,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(15.0),
          ),
          child: ListTile(
            contentPadding: const EdgeInsets.symmetric(vertical: 10.0, horizontal: 15.0),
            leading: ClipRRect(
              borderRadius: BorderRadius.circular(10.0),
              child: Image.asset(
                book.imagePath,
                width: 60,
                height: 80,
                fit: BoxFit.cover,
                errorBuilder: (context, error, stackTrace) => const Icon(Icons.broken_image,
size: 60),
              ),
            ),
            title: Text(
              book.name,
              style: const TextStyle(fontSize: 18, fontWeight: FontWeight.bold, color:
Colors.blue),
            ),
            subtitle: Text("Author: ${book.author}\nPublished: ${book.publishDate}",
style: TextStyle(color: Colors.blue.shade700)),
            trailing: const Icon(Icons.arrow_forward_ios, color: Colors.blue),
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => PdfViewerScreen(pdfAssetPath: book.pdfPath),
                ),
              );
            },
          ),
        );
      },
    );
  }

  Widget _buildGridView() {
    return GridView.builder(
      padding: const EdgeInsets.all(8.0),
      itemCount: _books.length,
      gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisCount: 2,
        crossAxisSpacing: 8.0,
        mainAxisSpacing: 8.0,
        childAspectRatio: 0.7,
      ),
      itemBuilder: (context, index) {
        final book = _books[index];
        return GestureDetector(
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => PdfViewerScreen(pdfAssetPath: book.pdfPath),
              ),
            );
          },
          child: Card(
            color: Colors.blue.shade100,
            elevation: 5,
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(15.0),
            ),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                Expanded(
                  child: ClipRRect(
                    borderRadius: const BorderRadius.vertical(top: Radius.circular(15.0)),
                    child: Image.asset(
                      book.imagePath,
                      width: double.infinity,
                      fit: BoxFit.cover,
                      errorBuilder: (context, error, stackTrace) => const
Icon(Icons.broken_image, size: 60),
                    ),
                  ),
                ),
                Padding(
                  padding: const EdgeInsets.all(8.0),
                  child: Text(
                    book.name,
                    textAlign: TextAlign.center,
                    style: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold, color:
Colors.blue),
                  ),
                ),
              ],
            ),
          ),
        );
      },
    );
  }
}
```

pdf_viewer_screen.dart

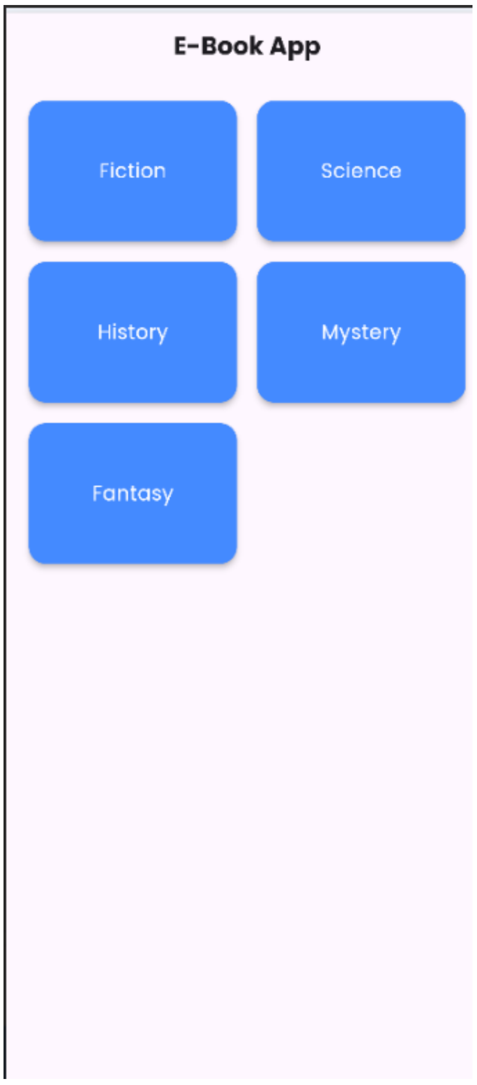
```

import 'dart:io';
import 'package:flutter/material.dart';
import 'package:flutter_pdfview/flutter_pdfview.dart';
import 'package:path_provider/path_provider.dart';
import 'package:flutter/services.dart';
import 'package:permission_handler/permission_handler.dart';

class PdfViewerScreen extends StatefulWidget {
  final String pdfAssetPath;

  const PdfViewerScreen({super.key, required this.pdfAssetPath});

  @override
  State<PdfViewerScreen> createState() => _PdfViewerScreenState();
}

class _PdfViewerScreenState extends State<PdfViewerScreen> {
  String? localFilePath;
  bool isLoading = true;
  bool isScrolling = false; // Default: Swipe mode enabled
  int totalPages = 0;
  PDFViewController? pdfController;
  UniqueKey pdfViewKey = UniqueKey(); // Key to force widget rebuild
```

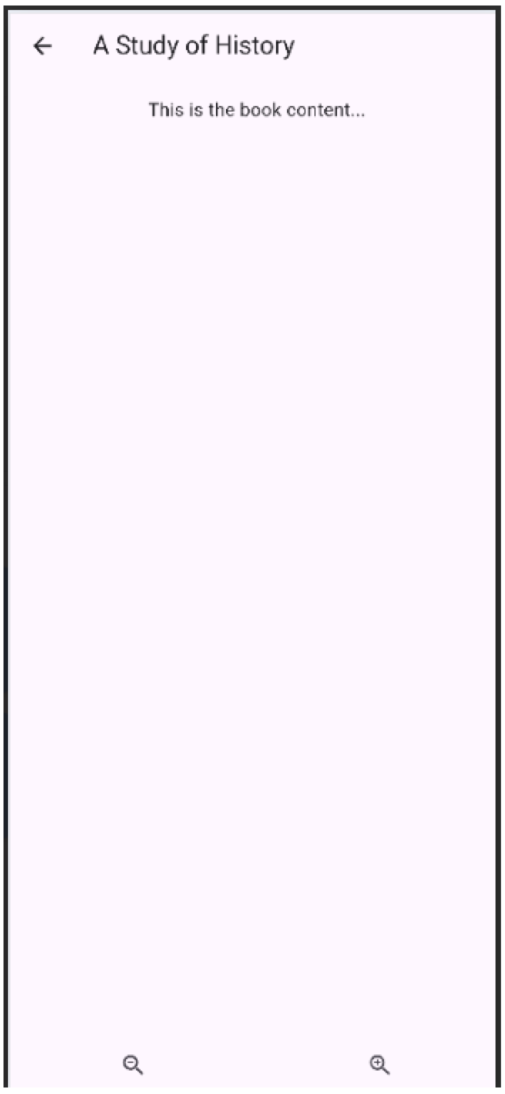
```
@override
void initState() {
  super.initState();
  _loadPdf();
}

Future<void> _loadPdf() async {
  try {
    final ByteData data = await rootBundle.load(widget.pdfAssetPath);
    final List<int> bytes = data.buffer.asUint8List();
    final Directory tempDir = await getTemporaryDirectory();
    final File tempFile = File("${tempDir.path}/temp.pdf");

    await tempFile.writeAsBytes(bytes, flush: true);

    setState(() {
      localFilePath = tempFile.path;
      isLoading = false;
    });
  } catch (e) {
    debugPrint("Error loading PDF: $e");
    setState(() {
      isLoading = false;
    });
  }
}

void _toggleScrollMode() {
  setState(() {
    isScrolling = !isScrolling;
    pdfViewKey = UniqueKey(); // Force PDFView to rebuild
  });
}

Future<void> _downloadPdf() async {
  if (localFilePath == null) return;

  try {
    if (Platform.isAndroid) {
      if (await Permission.storage.request().isGranted ||
          await Permission.manageExternalStorage.request().isGranted) {

        final Directory? downloadsDir = await getExternalStorageDirectory();
        if (downloadsDir != null) {
          final File destinationFile = File("${downloadsDir.path}/downloaded.pdf");
          await File(localFilePath!).copy(destinationFile.path);
        // Show pop-up after successful download
        showDialog(
          context: context,
          builder: (BuildContext context) {
            return AlertDialog(
              title: const Text("Download Complete"),
              content: Text("PDF downloaded to:\n${destinationFile.path}"),
              actions: [
                TextButton(
                  onPressed: () {
                    Navigator.of(context).pop(); // Close the dialog
                  },
                  child: const Text("OK"),
                ),
              ],
            );
          },
        );
      }
    } else {
      // Show a pop-up when permission is denied
      showDialog(
        context: context,
        builder: (BuildContext context) {
          return AlertDialog(
            title: const Text("Permission Denied"),
            content: const Text("Storage permission is required to download the PDF."),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.of(context).pop();
                  openAppSettings(); // Open app settings for permissions
                },
                child: const Text("Open Settings"),
              ),
              TextButton(
                onPressed: () {
                  Navigator.of(context).pop();
                },
                child: const Text("Cancel"),
              ),
            ],
          );
        },
      );
    }
```

```
      }
    } catch (e) {
      debugPrint("Error downloading PDF: $e");
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.blueAccent,
        title: const Text('E-Book Reader', style: TextStyle(color: Colors.white)),
        actions: [
          IconButton(
            icon: const Icon(Icons.download, color: Colors.white),
            onPressed: _downloadPdf,
          ),
        ],
      ),
      body: isLoading
          ? const Center(child: CircularProgressIndicator())
          : localFilePath != null
              ? Column(
                  children: [
                    Padding(
                      padding: const EdgeInsets.all(8.0),
                      child: ElevatedButton(
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.blue,
                          foregroundColor: Colors.white,
                        ),
                        onPressed: _toggleScrollMode,
                        child: Text(isScrolling ? "Switch to Swipe Mode" : "Switch to Scroll
Mode"),
                      ),
                    ),
                    Expanded(
                      child: PDFView(
                        key: pdfViewKey, // Ensures widget rebuilds
                        filePath: localFilePath!,
                        enableSwipe: true,
                        swipeHorizontal: !isScrolling, // Enable left/right swipe
                        autoSpacing: true,
                        pageSnap: true,
                        pageFling: true,
                        onRender: (pages) {
                          setState(() {
```

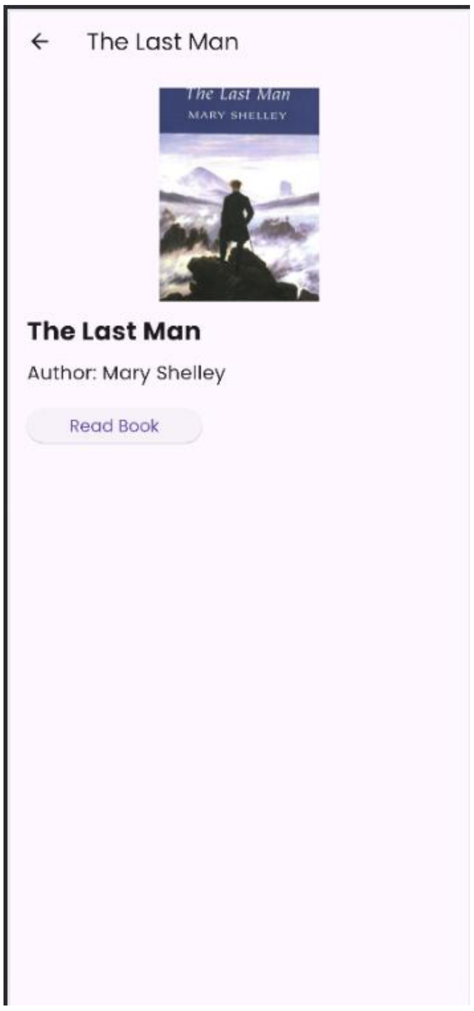
```
                    totalPages = pages ?? 0;
                  });
                },
                onViewCreated: (PDFViewController controller) {
                  setState(() {
                    pdfController = controller;
                  });
                },
                onError: (error) {
                  debugPrint("PDF Error: $error");
                },
              ),
            ),
          ],
        )
      : const Center(
          child: Text("Failed to load PDF", style: TextStyle(fontSize: 18, color:
Colors.red)),
        ),
    );
  }
}
```

OUTPUT:



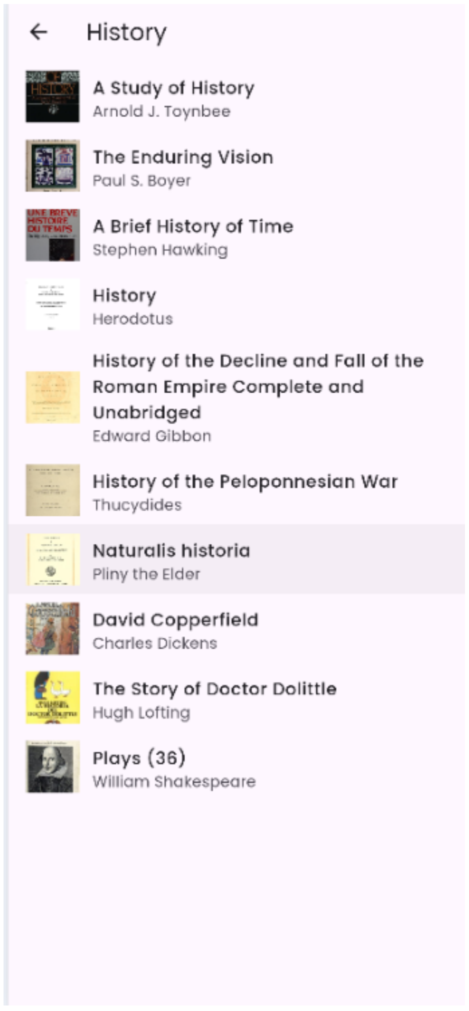


This is the book content...



The Last Man

Author: Mary Shelley



A Study of History
Arnold J. Toynbee

The Enduring Vision
Paul S. Boyer

A Brief History of Time
Stephen Hawking

History
Herodotus

History of the Decline and Fall of the Roman Empire Complete and Unabridged
Edward Gibbon

History of the Peloponnesian War
Thucydides

Naturalis historia
Pliny the Elder

David Copperfield
Charles Dickens

The Story of Doctor Dolittle
Hugh Lofting

Plays (36)
William Shakespeare