PROJECT REPORT
FINTRACK

Full-Stack Financial Management System for SMEs & Freelancers

Course
Mobile Application Development

Date
1/4/2026

Anas Nadeem 23SP-102-CS(B)
Anasnadeem193@gmail.com

1. PROJECT INTRODUCTION

FinTrack Pro is a high-performance mobile application developed using the **Flutter** framework and **Firebase** ecosystem. It is designed to empower small business owners, freelancers, and shopkeepers by digitalizing their financial activities. The app enables real-time tracking of sales and expenses while providing automated tax calculations and verifiable PDF reports.

1.1 Problem Statement

Manual bookkeeping is prone to human error, loss of physical records, and difficulties in calculating profit/loss across different currencies. FinTrack Pro automates these tasks to ensure 100% accuracy and data security.

---

2. TECHNICAL SPECIFICATIONS

- **Language:** Dart
- **Framework:** Flutter SDK (Material 3)
- **Database:** Cloud Firestore (NoSQL)
- **Auth:** Firebase Authentication
- **API:** ExchangeRate-API (RESTful)
- **Architecture:** Service-Based Architecture

---

3. CORE FEATURES

- **Secure Auth:** Business profile creation with persistent login.
- **Currency Engine:** Live forex rates for PKR, USD, INR, and AED.
- **Tax Logic:** Optional 18% GST calculation on every entry.
- **Smart Analytics:** Visual Pie Charts for financial health monitoring.
- **Verified Reporting:** PDF Invoices with secure QR Code auditing.

---

4. DATABASE ARCHITECTURE

The system utilizes a hierarchical NoSQL structure. Each user has a unique `uid`, ensuring that financial data remains private and isolated.

---

5. SOURCE CODE

PART A: Entry Point

This section contains the core setup of the application.

5.1 App Entry Point (`main.dart`)

```dart
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_fonts/google_fonts.dart';
import 'firebase_options.dart';
import 'screens/login_screen.dart';
import 'screens/dashboard_screen.dart';

void main() async {
  // Ensures that widget binding is initialized before calling native code like
Firebase
  WidgetsFlutterBinding.ensureInitialized();

  // Initializing Firebase with platform-specific options (Android, iOS, Web)
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );

  // Starting the Flutter application
  runApp(const FinTrackApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'FinTrack Pro',
      debugShowCheckedModeBanner: false, // Disables the debug banner in the top
right corner
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: const Color(0xFF673AB7)),
        useMaterial3: true, // Enabling Material 3 design system for modern UI
components
        textTheme: GoogleFonts.poppinsTextTheme(), // Implementing Poppins font
for better typography
```

```
      ),

      // PERSISTENT AUTHENTICATION LOGIC:
      // StreamBuilder monitors the user's login state in real-time.
      // If a user is already logged in, they are directed to the Dashboard.
      // Otherwise, the app shows the Login Screen.
      home: StreamBuilder<User?>(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          // Displaying a loader while Firebase is checking the authentication
status
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Scaffold(
              body: Center(child: CircularProgressIndicator()),
            );
          }

          // User is authenticated, navigate to the Dashboard
          if (snapshot.hasData) {
            return const DashboardScreen();
          }

          // No active session found, navigate to the Login Screen
          return const LoginScreen();
        },
      ),
    );
  }
}
```

5.2 Firebase Bridge (`firebase_options.dart`)

```
// File generated by FlutterFire CLI.
// ignore_for_file: type=lint
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
/// This class automatically selects the correct configuration based on the
current platform (Web, Android, iOS, etc.)
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    // Logic to detect if the app is running on a Web browser
    if (kIsWeb) {
```

```
    return web;
  }

  // Switch case to provide platform-specific Firebase configurations
  switch (defaultTargetPlatform) {
    case TargetPlatform.android:
      return android;
    case TargetPlatform.iOS:
      return ios;
    case TargetPlatform.macOS:
      return macos;
    case TargetPlatform.windows:
      return windows;
    case TargetPlatform.linux:
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for linux - '
        'you can reconfigure this by running the FlutterFire CLI again.',
      );
    default:
      throw UnsupportedError(
        'DefaultFirebaseOptions are not supported for this platform.',
      );
  }
}

// --- Web Configuration Details ---
static const FirebaseOptions web = FirebaseOptions(
  apiKey: 'AIzaSyDXodqkfQHT85FJCgHvF1O2-JJ8gKRFICs',
  appId: '1:157534509468:web:9389d011f91f37c0721059',
  messagingSenderId: '157534509468',
  projectId: 'fin-track-5d379',
  authDomain: 'fin-track-5d379.firebaseapp.com',
  storageBucket: 'fin-track-5d379.firebasestorage.app',
  measurementId: 'G-ZK1MRXV1V2',
);

// --- Android Configuration Details ---
static const FirebaseOptions android = FirebaseOptions(
  apiKey: 'AIzaSyAYoI1xARRHmyWNfidtLBt6KcSbdjusCfg',
  appId: '1:157534509468:android:9928bbe5fe31c9e1721059',
  messagingSenderId: '157534509468',
  projectId: 'fin-track-5d379',
  storageBucket: 'fin-track-5d379.firebasestorage.app',
);
```

```
  // --- iOS Configuration Details ---
  static const FirebaseOptions ios = FirebaseOptions(
    apiKey: 'AIzaSyDDMWmTNHkO0n23MOQ7wINbSFXo0s6C01o',
    appId: '1:157534509468:ios:08c2079fbc1bb6ae721059',
    messagingSenderId: '157534509468',
    projectId: 'fin-track-5d379',
    storageBucket: 'fin-track-5d379.firebasestorage.app',
    iosBundleId: 'com.example.finTrack',
  );

  // --- macOS Configuration Details ---
  static const FirebaseOptions macos = FirebaseOptions(
    apiKey: 'AIzaSyDDMWmTNHkO0n23MOQ7wINbSFXo0s6C01o',
    appId: '1:157534509468:ios:08c2079fbc1bb6ae721059',
    messagingSenderId: '157534509468',
    projectId: 'fin-track-5d379',
    storageBucket: 'fin-track-5d379.firebasestorage.app',
    iosBundleId: 'com.example.finTrack',
  );

  // --- Windows Configuration Details ---
  static const FirebaseOptions windows = FirebaseOptions(
    apiKey: 'AIzaSyDXodqkfQHT85FJCgHvF1O2-JJ8gKRFICs',
    appId: '1:157534509468:web:79ab6a14d3ffabe8721059',
    messagingSenderId: '157534509468',
    projectId: 'fin-track-5d379',
    authDomain: 'fin-track-5d379.firebaseapp.com',
    storageBucket: 'fin-track-5d379.firebasestorage.app',
    measurementId: 'G-DDYVYTSPV4',
  );
}
```

PART B: Backend Services (The Engine)

These files handle the background logic, APIs, and database communication.

5.3 Authentication Service (`services/auth_service.dart`)

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

class AuthService {
  final FirebaseAuth _auth = FirebaseAuth.instance;
```

```dart
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  // Stream to monitor real-time authentication state changes
  Stream<User?> get user => _auth.authStateChanges();

  // SIGN UP method to create a new user account with extended profile data
  Future<User?> register({
    required String email,
    required String password,
    required String name,
    required String businessName,
    required String phone,
    required String baseCurrency,
  }) async {
    try {
      // 1. Authenticate and create user credentials in Firebase Auth
      UserCredential result = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      User? user = result.user;

      if (user != null) {
        // 2. Synchronize the user's name with the Firebase Auth DisplayName
property
        await user.updateDisplayName(name);

        // 3. Create a comprehensive User Profile document in Cloud Firestore
        // This data is utilized for generating business reports and setting
dashboard preferences
        await _firestore.collection('users').doc(user.uid).set({
          'uid': user.uid,
          'name': name,
          'businessName': businessName,
          'email': email,
          'phone': phone,
          'baseCurrency': baseCurrency, // Stores default preference (e.g., PKR,
AED, INR)
          'createdAt': FieldValue.serverTimestamp(),
        });
      }
      return user;
    } on FirebaseAuthException catch (e) {
      // Handling specific Firebase Authentication exceptions
      debugPrint("Auth Error: ${e.message}");
```

```
      return null;
    } catch (e) {
      // Catch-all for general logic or connection errors
      debugPrint("General Error: ${e.toString()}");
      return null;
    }
  }

  // SIGN IN method to authenticate existing users
  Future<User?> login(String email, String password) async {
    try {
      UserCredential result = await _auth.signInWithEmailAndPassword(
        email: email,
        password: password
      );
      return result.user;
    } catch (e) {
      debugPrint("Login Error: ${e.toString()}");
      return null;
    }
  }

  // SIGN OUT method to terminate the current session
  Future<void> logout() async {
    try {
      await _auth.signOut();
    } catch (e) {
      debugPrint("Logout Error: ${e.toString()}");
    }
  }

  // HELPER: Fetches specific user profile details from Firestore
  // Used for displaying personalized business info on the dashboard and reports
  Future<DocumentSnapshot> getUserProfile(String uid) async {
    return await _firestore.collection('users').doc(uid).get();
  }
}
```

5.4 Finance Logic Service (`services/finance_service.dart`)

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

class FinanceService {
  final _db = FirebaseFirestore.instance;
```

```
  final _auth = FirebaseAuth.instance;

  // --- Real-time Financial Calculation Logic ---
  // This stream calculates Total Income, Total Expense, and Net Balance in real-
time
  Stream<Map<String, double>> getFinancialSummary() {
    // Accessing current user's UID for data security and isolation
    String uid = _auth.currentUser!.uid;

    return _db
        .collection('users')
        .doc(uid)
        .collection('transactions')
        .snapshots() // Listening for real-time updates in the database
        .map((snapshot) {
      double totalIncome = 0;
      double totalExpense = 0;

      // Iterating through all transaction documents to categorize and sum
amounts
      for (var doc in snapshot.docs) {
        double amount = (doc['amount'] as num).toDouble();
        if (doc['type'] == 'income') {
          totalIncome += amount;
        } else {
          totalExpense += amount;
        }
      }

      // Returning a mapped summary of the business's financial health
      return {
        'income': totalIncome,
        'expense': totalExpense,
        'balance': totalIncome - totalExpense,
      };
    });
  }
}
```

5.5 Data Management Service (`services/firestore_service.dart`)

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/foundation.dart'; // Required for using debugPrint
```

```dart
class FirestoreService {
  final FirebaseFirestore _db = FirebaseFirestore.instance;
  final String? uid = FirebaseAuth.instance.currentUser?.uid;

  // --- ADD TRANSACTION METHOD ---
  // Stores a new financial record in the user's specific sub-collection
  Future<void> addTransaction({
    required String title,
    required double amount,
    required String type,
    required String category,
  }) async {
    // Ensuring the user is authenticated before attempting to write data
    if (uid == null) return;

    try {
      // Data is structured as: users -> {uid} -> transactions -> {auto-id doc}
      await _db.collection('users').doc(uid).collection('transactions').add({
        'title': title,
        'amount': amount,
        'type': type, // Categorized as 'income' or 'expense'
        'category': category,
        'date': FieldValue.serverTimestamp(), // Captures the exact server-side
timestamp
      });
    } catch (e) {
      // debugPrint is preferred over print for production-level logging
      debugPrint("Error adding transaction: $e");
    }
  }

  // --- REAL-TIME TRANSACTION STREAM ---
  // Retrieves a live stream of transactions, sorted by the most recent date
  Stream<QuerySnapshot> getTransactions() {
    return _db
        .collection('users')
        .doc(uid)
        .collection('transactions')
        .orderBy('date', descending: true) // Ensuring the latest entries appear
first
        .snapshots();
  }
}
```

5.6 External API Service (`services/currency_service.dart`)

```
import 'dart:convert';
import 'package:flutter/foundation.dart'; // Required for debugPrint $cite: 10.1$
import 'package:http/http.dart' as http; // Package for making REST API calls
$cite: 7.1$

class CurrencyService {

  // --- 1. Fetch Live Exchange Rates from External API ---
  // Retrieves real-time rates with PKR as the base currency $cite: 7.1$
  static Future<double> getLiveRate(String targetCurrency) async {
    // If the selected currency is PKR, no conversion is necessary
    if (targetCurrency == "PKR") return 1.0;

    try {
      // Fetching the latest forex market data from ExchangeRate-API $cite: 7.1$
      final response = await http.get(
        Uri.parse('https://api.exchangerate-api.com/v4/latest/PKR')
      );

      if (response.statusCode == 200) {
        final data = json.decode(response.body);

        // Extracting the conversion rate for the requested target currency (USD,
INR, AED)
        if (data['rates'] != null && data['rates'][targetCurrency] != null) {
          return (data['rates'][targetCurrency] as num).toDouble();
        }
      }

      // Fallback to hardcoded rates if the API response is unsuccessful
      return _getFallbackRate(targetCurrency);
    } catch (e) {
      // Using debugPrint instead of print for better production logging $cite:
10.1$
      debugPrint("Currency API Error: $e");
      return _getFallbackRate(targetCurrency);
    }
  }

  // --- 2. Currency Symbol Logic ---
  // Maps ISO currency codes to their respective visual symbols $cite: 5.1, 7.1$
  static String getSymbol(String code) {
    switch (code) {
      case "USD":
```

```
        return "\$"; // United States Dollar
      case "INR":
        return "₹"; // Indian Rupee
      case "AED":
        return "د.إ"; // United Arab Emirates Dirham
      case "PKR":
      default:
        return "Rs"; // Pakistani Rupee
    }
  }

  // --- 3. Offline Fallback Strategy ---
  // Provides estimated rates in case of network unavailability $cite: 7.1$
  static double _getFallbackRate(String code) {
    Map<String, double> rates = {
      "USD": 0.0036, // Approximate conversion: 1 PKR to USD
      "INR": 0.30,   // Approximate conversion: 1 PKR to INR
      "AED": 0.013,  // Approximate conversion: 1 PKR to AED
    };
    return rates[code] ?? 1.0;
  }
}
```

5.7 Reporting & PDF Service (`services/pdf_service.dart`)

```
import 'package:pdf/pdf.dart';
import 'package:pdf/widgets.dart' as pw;
import 'package:printing/printing.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart'; // Required for fetching
user-specific business details
import 'currency_service.dart';

class PdfService {
  static Future<void> generateInvoice(
    String description,
    double balance,
    double income,
    double expense,
    String date,
    String currencyCode
  ) async {
    final pdf = pw.Document();
    String symbol = CurrencyService.getSymbol(currencyCode);
```

```
    // --- 1. Fetching Business Profile from Firestore ---
    // Retrieves personalized information to brand the audit report
    final String uid = FirebaseAuth.instance.currentUser?.uid ?? "";
    final userDoc = await
FirebaseFirestore.instance.collection('users').doc(uid).get();

    String businessName = "FinTrack Business Pro";
    String userPhone = "N/A";
    String ownerName = "Authorized User";

    if (userDoc.exists) {
      var data = userDoc.data() as Map<String, dynamic>;
      businessName = data['businessName'] ?? "FinTrack Business Pro";
      userPhone = data['phone'] ?? "N/A";
      ownerName = data['name'] ?? "User";
    }

    // --- 2. Font Loading for Internationalization ---
    // Loading Google Fonts to support special symbols like INR (₹) and Arabic
(د.إ)
    final arabicFont = await PdfGoogleFonts.notoSansArabicRegular();
    final baseFont = await PdfGoogleFonts.notoSansRegular();
    final boldFont = await PdfGoogleFonts.notoSansBold();

    pdf.addPage(
      pw.Page(
        pageFormat: PdfPageFormat.a4,
        theme: pw.ThemeData.withFont(
          base: baseFont,
          bold: boldFont,
          fontFallback: [arabicFont, baseFont],
        ),
        build: (pw.Context context) {
          return pw.Padding(
            padding: const pw.EdgeInsets.all(30),
            child: pw.Column(
              crossAxisAlignment: pw.CrossAxisAlignment.start,
              children: [
                // --- Professional Header Section ---
                // Displays real-time business identity and audit timestamp
                pw.Row(
                  mainAxisAlignment: pw.MainAxisAlignment.spaceBetween,
                  children: [
                    pw.Column(
                      crossAxisAlignment: pw.CrossAxisAlignment.start,
```

```
                    children: [
                      pw.Text(businessName,
                        style: pw.TextStyle(fontSize: 24, fontWeight:
pw.FontWeight.bold, color: PdfColors.blueAccent)),
                      pw.Text("Contact: $userPhone", style: const
pw.TextStyle(fontSize: 10)),
                      pw.Text("Owner: $ownerName", style: const
pw.TextStyle(fontSize: 10)),
                    ]
                  ),
                  pw.Column(
                    crossAxisAlignment: pw.CrossAxisAlignment.end,
                    children: [
                      pw.Text("FINANCIAL AUDIT", style: pw.TextStyle(fontSize:
18, fontWeight: pw.FontWeight.bold)),
                      pw.Text("Date: $date"),
                      pw.Text("Currency: $currencyCode"),
                    ]
                  ),
                ],
              ),
              pw.SizedBox(height: 10),
              pw.Divider(thickness: 2, color: PdfColors.blueAccent),
              pw.SizedBox(height: 20),

              // --- Financial Summary Table ---
              // Organized representation of income, expense, and net balance
              pw.TableHelper.fromTextArray(
                border: pw.TableBorder.all(width: 0.5, color: PdfColors.grey),
                headerStyle: pw.TextStyle(fontWeight: pw.FontWeight.bold,
color: PdfColors.white),
                headerDecoration: const pw.BoxDecoration(color:
PdfColors.blueAccent),
                headers: ['Category', 'Amount in $currencyCode'],
                data: [
                  ['Total Business Income', '$symbol
${income.toStringAsFixed(2)}'],
                  ['Total Business Expense', '$symbol
${expense.toStringAsFixed(2)}'],
                  ['Net Profit / Loss', '$symbol
${balance.toStringAsFixed(2)}'],
                ],
              ),

              pw.SizedBox(height: 40),
```

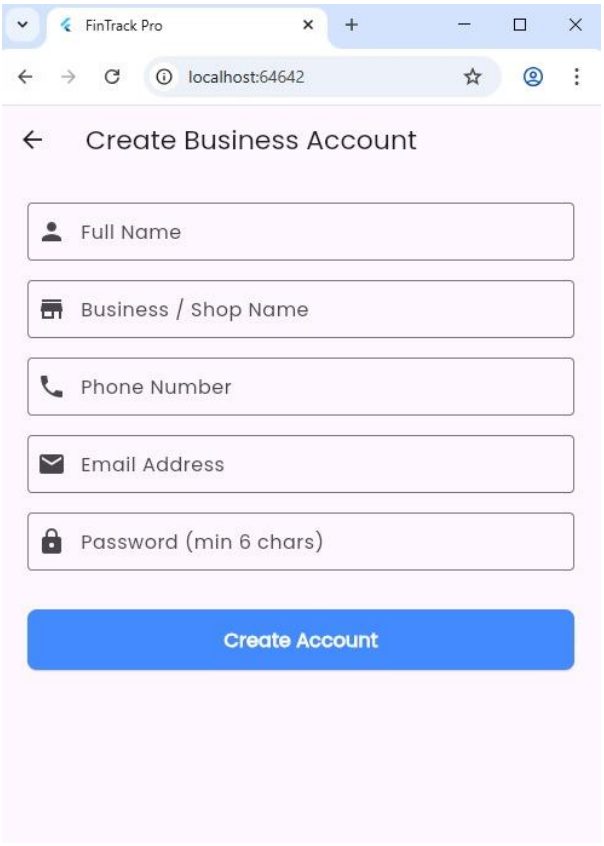
```
                // --- Secure QR Audit Section ---
                // Generates a QR code containing full transaction breakdown for
offline verification
                pw.Row(
                  mainAxisAlignment: pw.MainAxisAlignment.spaceBetween,
                  crossAxisAlignment: pw.CrossAxisAlignment.end,
                  children: [
                    pw.Column(
                      children: [
                        pw.BarcodeWidget(
                          barcode: pw.Barcode.qrCode(),
                          // String data contains the full audit trail for the
scanner
                          data: "--- FINTRACK OFFICIAL AUDIT ---\n"
                                "Business: $businessName\n"
                                "Owner: $ownerName\n"
                                "Date: $date\n"
                                "------------------------------\n"
                                "Total Income: $symbol
${income.toStringAsFixed(2)}\n"
                                "Total Expense: $symbol
${expense.toStringAsFixed(2)}\n"
                                "Net Balance: $symbol
${balance.toStringAsFixed(2)}\n"
                                "------------------------------\n"
                                "Security Status: 100% Verified",
                          width: 130,
                          height: 130,
                        ),
                        pw.SizedBox(height: 5),
                        pw.Text("Scan for Audit Verification", style: const
pw.TextStyle(fontSize: 8)),
                      ],
                    ),
                    pw.Column(
                      children: [
                        pw.Text(ownerName, style: pw.TextStyle(fontWeight:
pw.FontWeight.bold)),
                        pw.Container(
                          width: 160,
                          padding: const pw.EdgeInsets.only(top: 5),
                          decoration: const pw.BoxDecoration(
                            border: pw.Border(top: pw.BorderSide(width: 1, color:
PdfColors.black)),
```

```
                        ),
                        child: pw.Center(child: pw.Text("Authorized Signature",
style: const pw.TextStyle(fontSize: 10))),
                      ),
                    ],
                  ),
                ],
              ),
              pw.Spacer(),
              pw.Divider(thickness: 0.5, color: PdfColors.grey),
              pw.Center(
                child: pw.Text("This is a system generated report secured by
FinTrack Business Architecture",
                  style: const pw.TextStyle(fontSize: 8, color:
PdfColors.grey)),
              ),
            ],
          ),
        );
      },
    ),
  );

  // Launching the PDF print/save dialog on the device
  await Printing.layoutPdf(
    onLayout: (PdfPageFormat format) async => pdf.save(),
    name: 'Report_${businessName}_$date.pdf',
  );
 }
}
```

PART C: User Interface (The User Flow)

This section follows the actual steps a user takes while using the app.

5.8 Account Creation (`screens/signup_screen.dart`)



Code:

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

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

  @override
  State<SignupScreen> createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
  // Text editing controllers to manage and retrieve user input for registration
  final _nameController = TextEditingController();
  final _businessController = TextEditingController();
  final _phoneController = TextEditingController();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();

  final AuthService _authService = AuthService();
  bool _isLoading = false;
```

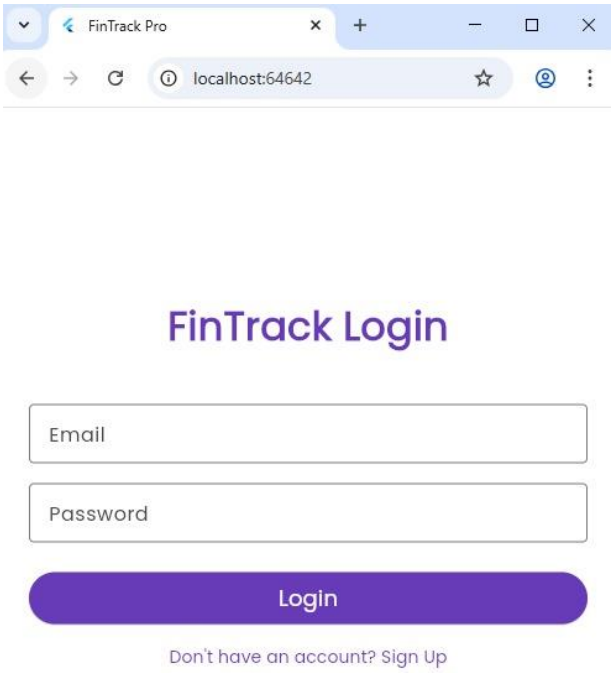
```
  // Setting PKR as the default base currency for new users in the backend
  final String _defaultCurrency = 'PKR';

  // Function to handle the business account registration process
  void _handleSignup() async {
    // Validating that all mandatory fields are filled before proceeding
    if (_nameController.text.trim().isEmpty ||
        _businessController.text.trim().isEmpty ||
        _emailController.text.trim().isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text("Please fill in all required fields")),
      );
      return;
    }

    setState(() => _isLoading = true);

    // Registering the user and storing extended business profile details in
Firestore
    var user = await _authService.register(
      email: _emailController.text.trim(),
      password: _passwordController.text.trim(),
      name: _nameController.text.trim(),
      businessName: _businessController.text.trim(),
      phone: _phoneController.text.trim(),
      baseCurrency: _defaultCurrency,
    );

    if (!mounted) return;
    setState(() => _isLoading = false);

    if (user != null) {
      // Success: Inform the user and navigate back to the login screen
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text("Account Created Successfully! Please
Login.")),
      );
      Navigator.pop(context);
    } else {
      // Failure: Alert the user regarding network or credential issues
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text("Signup Failed: Check your network or email
address.")),
      );
```

```
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Create Business Account")),
      body: SingleChildScrollView(
        // SingleChildScrollView ensures the UI remains accessible when the
keyboard appears
        padding: const EdgeInsets.all(24.0),
        child: Column(
          children: [
            // User's Legal Full Name input
            TextField(
              controller: _nameController,
              decoration: const InputDecoration(
                labelText: 'Full Name',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.person),
              ),
            ),
            const SizedBox(height: 16),

            // Registered Business or Shop name for professional reporting
            TextField(
              controller: _businessController,
              decoration: const InputDecoration(
                labelText: 'Business / Shop Name',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.store),
              ),
            ),
            const SizedBox(height: 16),

            // Official Business Contact Number
            TextField(
              controller: _phoneController,
              keyboardType: TextInputType.phone,
              decoration: const InputDecoration(
                labelText: 'Phone Number',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.phone),
              ),
            ),
```

```
            const SizedBox(height: 16),

            // Business Email Address for account recovery and login
            TextField(
              controller: _emailController,
              keyboardType: TextInputType.emailAddress,
              decoration: const InputDecoration(
                labelText: 'Email Address',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.email),
              ),
            ),
            const SizedBox(height: 16),

            // Secure account password input
            TextField(
              controller: _passwordController,
              obscureText: true,
              decoration: const InputDecoration(
                labelText: 'Password (min 6 chars)',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.lock),
              ),
            ),
            const SizedBox(height: 32),

            // Displays a loading indicator or the action button based on the
state
            _isLoading
              ? const CircularProgressIndicator()
              : SizedBox(
                  width: double.infinity,
                  height: 50,
                  child: ElevatedButton(
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.blueAccent,
                      shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(8)),
                    ),
                    onPressed: _handleSignup,
                    child: const Text(
                      "Create Account",
                      style: TextStyle(color: Colors.white, fontSize: 16,
fontWeight: FontWeight.bold),
                    ),
```

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

  @override
  void dispose() {
    // Releasing memory by disposing controllers when the screen is removed from
the widget tree
    _nameController.dispose();
    _businessController.dispose();
    _phoneController.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }
}
```

5.9 Secure Login (`screens/login_screen.dart`)



Code:

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

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

  @override
  State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  // Controllers to capture user input for authentication
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  final AuthService _authService = AuthService();
  bool _isLoading = false;

  // Method to handle the authentication logic using Firebase
```

```
  void _handleLogin() async {
    setState(() => _isLoading = true);

    // Attempting to sign in the user through the AuthService
    var user = await _authService.login(_emailController.text.trim(),
_passwordController.text.trim());

    setState(() => _isLoading = false);

    if (user != null) {
      if (mounted) {
        // Navigating to the Dashboard upon successful authentication
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => const DashboardScreen()),
        );
      }
    } else {
      if (mounted) {
        // Displaying an error message if the login credentials are incorrect
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text("Login Failed: Check your
email/password")),
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: Padding(
        padding: const EdgeInsets.all(24.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            const Text(
              "FinTrack Login",
              textAlign: TextAlign.center,
              style: TextStyle(fontSize: 32, fontWeight: FontWeight.bold, color:
Color(0xFF673AB7)),
            ),
            const SizedBox(height: 40),
```

```
            // Input field for the user's registered email address
            TextField(
              controller: _emailController,
              decoration: const InputDecoration(labelText: 'Email', border:
OutlineInputBorder()),
            ),
            const SizedBox(height: 16),

            // Input field for the user's secure password
            TextField(
              controller: _passwordController,
              obscureText: true,
              decoration: const InputDecoration(labelText: 'Password', border:
OutlineInputBorder()),
            ),
            const SizedBox(height: 24),

            // Displaying a progress indicator while authentication is in
progress
            _isLoading
              ? const Center(child: CircularProgressIndicator())
              : ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: const Color(0xFF673AB7),
                    padding: const EdgeInsets.symmetric(vertical: 16),
                  ),
                  onPressed: _handleLogin,
                  child: const Text("Login", style: TextStyle(color:
Colors.white, fontSize: 18)),
                ),
            const SizedBox(height: 10),

            // Navigation link for new users to register an account
            TextButton(
              onPressed: () {
                // Navigating to the SignupScreen to create a new business
account
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => const SignupScreen()),
                );
              },
              child: const Text("Don't have an account? Sign Up", style:
TextStyle(color: Color(0xFF673AB7))),
```

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

5.10 Main Dashboard (`screens/dashboard_screen.dart`)

Code:

```
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:fl_chart/fl_chart.dart';
import 'package:intl/intl.dart';
import '../services/finance_service.dart';
import '../services/pdf_service.dart';
import '../services/currency_service.dart';
import '../services/auth_service.dart';
import 'add_transaction_screen.dart';
```

```dart
import 'transaction_list_screen.dart';
import 'login_screen.dart';

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

  @override
  State<DashboardScreen> createState() => _DashboardScreenState();
}

class _DashboardScreenState extends State<DashboardScreen> {
  String _selectedCurrency = "PKR";
  double _exchangeRate = 1.0;
  bool _isLoadingRate = false;

  // Initializing AuthService and capturing the Current User's UID for data
isolation
  final AuthService _authService = AuthService();
  final String _uid = FirebaseAuth.instance.currentUser?.uid ?? "";

  // Standard logout procedure to clear session and navigate to LoginScreen
  void _logout(BuildContext context) async {
    await FirebaseAuth.instance.signOut();
    if (context.mounted) {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (context) => const LoginScreen()),
      );
    }
  }

  // Triggered when the user selects a different currency from the dropdown
  void _handleCurrencyChange(String? newCurrency) async {
    if (newCurrency == null || newCurrency == _selectedCurrency) return;
    setState(() => _isLoadingRate = true);

    // Fetching live exchange rates from the CurrencyService API
    double rate = await CurrencyService.getLiveRate(newCurrency);
    setState(() {
      _selectedCurrency = newCurrency;
      _exchangeRate = rate;
      _isLoadingRate = false;
    });
  }
```

```
  @override
  Widget build(BuildContext context) {
    String symbol = CurrencyService.getSymbol(_selectedCurrency);
    String currentDate = DateFormat('yyyy-MM-dd').format(DateTime.now());

    return Scaffold(
      backgroundColor: Colors.grey[100],
      appBar: AppBar(
        // Dynamic Title: Fetching and displaying the Business Name and User Name
from Firestore
        title: FutureBuilder<DocumentSnapshot>(
          future: _authService.getUserProfile(_uid),
          builder: (context, snapshot) {
            if (snapshot.hasData && snapshot.data!.exists) {
              var userData = snapshot.data!.data() as Map<String, dynamic>;
              String bName = userData['businessName'] ?? "FinTrack Pro";
              String uName = userData['name'] ?? "User";

              return Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(bName, style: const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold, color: Colors.white)),
                  Text("Hi, $uName", style: const TextStyle(fontSize: 12, color:
Colors.white70)),
                ],
              );
            }
            // Fallback text while data is loading or if it does not exist
            return const Text('FinTrack Business Pro', style:
TextStyle(fontWeight: FontWeight.bold, color: Colors.white));
          },
        ),
        backgroundColor: Colors.blueAccent,
        elevation: 0,
        iconTheme: const IconThemeData(color: Colors.white),
        actions: [
          _isLoadingRate
              ? const Center(child: Padding(padding:
EdgeInsets.symmetric(horizontal: 10), child: SizedBox(width: 20, height: 20,
child: CircularProgressIndicator(color: Colors.white, strokeWidth: 2))))
              : DropdownButton<String>(
                  value: _selectedCurrency,
                  dropdownColor: Colors.blueAccent,
                  underline: Container(),
```

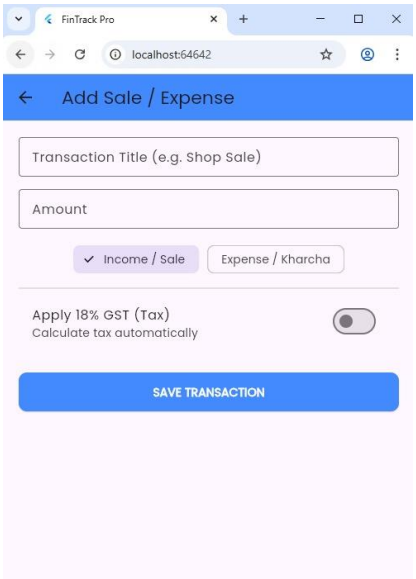
```
                icon: const Icon(Icons.currency_exchange, color: Colors.white),
                items: ['PKR', 'USD', 'INR', 'AED'].map((val) =>
DropdownMenuItem(
                  value: val,
                  child: Text(val, style: const TextStyle(color:
Colors.white)),
                )).toList(),
                onChanged: _handleCurrencyChange,
              ),
          IconButton(
            icon: const Icon(Icons.logout, color: Colors.white),
            onPressed: () => _logout(context),
          ),
        ],
      ),
      body: StreamBuilder<Map<String, double>>(
        // Real-time financial data streaming from Firestore
        stream: FinanceService().getFinancialSummary(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) return const Center(child:
CircularProgressIndicator());

          final rawData = snapshot.data!;
          // Applying live exchange rates to the base amounts for multi-currency
support
          final double displayIncome = rawData['income']! * _exchangeRate;
          final double displayExpense = rawData['expense']! * _exchangeRate;
          final double displayBalance = rawData['balance']! * _exchangeRate;

          return SingleChildScrollView(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              children: [
                _buildMainBalanceCard(displayBalance, symbol),
                const SizedBox(height: 20),
                Row(
                  children: [
                    Expanded(child: _buildSummaryCard("Income", displayIncome,
Colors.green, Icons.add_circle, symbol, context)),
                    const SizedBox(width: 15),
                    Expanded(child: _buildSummaryCard("Expense", displayExpense,
Colors.red, Icons.remove_circle, symbol, context)),
                  ],
                ),
                const SizedBox(height: 30),
```

```
                const Text("Visual Report", style: TextStyle(fontSize: 18,
fontWeight: FontWeight.bold)),
                const SizedBox(height: 10),
                // Graphical representation of Income vs Expenses using a
PieChart
                Container(
                  height: 220,
                  padding: const EdgeInsets.all(10),
                  decoration: BoxDecoration(color: Colors.white, borderRadius:
BorderRadius.circular(20)),
                  child: PieChart(PieChartData(sections: [
                    PieChartSectionData(color: Colors.green, value:
displayIncome, title: 'In', radius: 50, titleStyle: const TextStyle(color:
Colors.white, fontWeight: FontWeight.bold)),
                    PieChartSectionData(color: Colors.red, value: displayExpense,
title: 'Out', radius: 50, titleStyle: const TextStyle(color: Colors.white,
fontWeight: FontWeight.bold)),
                  ])),
                ),
                const SizedBox(height: 30),
                // Button to trigger professional PDF report generation
                ElevatedButton.icon(
                  onPressed: () {
                    PdfService.generateInvoice(
                      "Business Summary Report",
                      displayBalance,
                      displayIncome,
                      displayExpense,
                      currentDate,
                      _selectedCurrency
                    );
                  },
                  icon: const Icon(Icons.picture_as_pdf, color: Colors.white),
                  label: const Text("GENERATE PDF INVOICE", style:
TextStyle(color: Colors.white)),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.blueAccent,
                    minimumSize: const Size(double.infinity, 55),
                    shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(12))
                  ),
                ),
                const SizedBox(height: 80),
              ],
            ),
```

```
          );
        },
      ),
      floatingActionButton: FloatingActionButton.extended(
        onPressed: () => Navigator.push(context, MaterialPageRoute(builder:
(context) => const AddTransactionScreen())),
        label: const Text("Add Entry", style: TextStyle(color: Colors.white,
fontWeight: FontWeight.bold)),
        icon: const Icon(Icons.add, color: Colors.white),
        backgroundColor: Colors.blueAccent,
      ),
    );
  }

  // Widget to display the primary Net Profit/Loss balance card
  Widget _buildMainBalanceCard(double amount, String symbol) {
    return Container(
      width: double.infinity,
      padding: const EdgeInsets.all(25),
      decoration: BoxDecoration(
        gradient: const LinearGradient(colors: [Colors.blueAccent,
Colors.indigoAccent]),
        borderRadius: BorderRadius.circular(20)
      ),
      child: Column(
        children: [
          const Text("Net Profit/Loss", style: TextStyle(color: Colors.white70,
fontSize: 16)),
          Text("$symbol ${amount.toStringAsFixed(2)}", style: const
TextStyle(color: Colors.white, fontSize: 32, fontWeight: FontWeight.bold)),
        ],
      ),
    );
  }

  // Generic card widget for Income and Expense summaries
  Widget _buildSummaryCard(String title, double amount, Color color, IconData
icon, String symbol, BuildContext context) {
    return GestureDetector(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => TransactionListScreen(
              filterType: title.toLowerCase(),
```

```
              currencySymbol: symbol,
              exchangeRate: _exchangeRate,
            ),
          ),
        );
      },
      child: Container(
        padding: const EdgeInsets.all(15),
        decoration: BoxDecoration(color: Colors.white, borderRadius:
BorderRadius.circular(15), boxShadow: const [BoxShadow(color: Colors.black12,
blurRadius: 5)]),
        child: Column(
          children: [
            Icon(icon, color: color, size: 28),
            const SizedBox(height: 8),
            Text(title, style: const TextStyle(color: Colors.grey, fontWeight:
FontWeight.w500)),
            Text("$symbol ${amount.toStringAsFixed(2)}", style: TextStyle(color:
color, fontWeight: FontWeight.bold, fontSize: 14)),
          ],
        ),
      ),
    );
  }
}
```

5.11 Data Entry UI (`screens/add_transaction_screen.dart`)



Code:

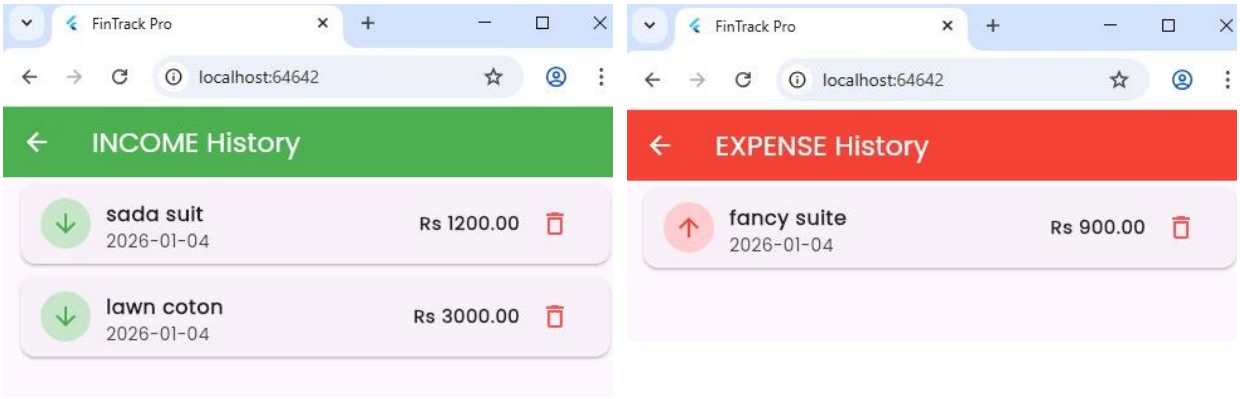
```dart
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';

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

  @override
  State<AddTransactionScreen> createState() => _AddTransactionScreenState();
}

class _AddTransactionScreenState extends State<AddTransactionScreen> {
  final _titleController = TextEditingController();
  final _amountController = TextEditingController();

  String _transactionType = 'income'; // Default type set to income
  bool _isTaxApplied = false; // Toggle for GST calculation logic
  final double _taxRate = 18.0; // Standard tax rate for the application

  void _saveToFirebase() async {
    // Validation to ensure all required fields are provided
    if (_titleController.text.isEmpty || _amountController.text.isEmpty) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(content: Text("Please fill all fields")),
      );
      return;
    }

    double baseAmount = double.parse(_amountController.text);
    // Calculation logic for applying 18% GST if the toggle is enabled
    double finalAmount = _isTaxApplied
        ? baseAmount + (baseAmount * (_taxRate / 100))
        : baseAmount;

    try {
      // Fetching the unique ID of the currently logged-in user
      String uid = FirebaseAuth.instance.currentUser!.uid;

      // Saving the transaction details under the specific user's collection in
Firestore
      await FirebaseFirestore.instance
          .collection('users')
          .doc(uid)
          .collection('transactions')
          .add({
```

```
        'title': _titleController.text,
        'amount': finalAmount,
        'type': _transactionType,
        'category': 'General',
        'date': DateTime.now(),
        'taxIncluded': _isTaxApplied,
      });

      if (mounted) {
        Navigator.pop(context); // Return to previous screen after successful
save
      }
    } catch (e) {
      // Changed print to debugPrint to comply with Flutter production standards
      debugPrint("Error saving: $e");
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Add Sale / Expense"),
        backgroundColor: Colors.blueAccent,
      ),
      body: SingleChildScrollView(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          children: [
            TextField(
              controller: _titleController,
              decoration: const InputDecoration(
                labelText: "Transaction Title (e.g. Shop Sale)",
                border: OutlineInputBorder(),
              ),
            ),
            const SizedBox(height: 15),

            TextField(
              controller: _amountController,
              decoration: const InputDecoration(
                labelText: "Amount",
                prefixText: "Rs. ",
                border: OutlineInputBorder(),
              ),
```

```
              keyboardType: TextInputType.number,
            ),
            const SizedBox(height: 20),

            // Selection chips for Transaction Type
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                ChoiceChip(
                  label: const Text("Income / Sale"),
                  selected: _transactionType == 'income',
                  onSelected: (val) => setState(() => _transactionType =
'income'),
                ),
                const SizedBox(width: 10),
                ChoiceChip(
                  label: const Text("Expense / Kharcha"),
                  selected: _transactionType == 'expense',
                  onSelected: (val) => setState(() => _transactionType =
'expense'),
                ),
              ],
            ),

            const SizedBox(height: 10),
            const Divider(),

            // GST Toggle Switch
            SwitchListTile(
              title: const Text("Apply 18% GST (Tax)"),
              subtitle: const Text("Calculate tax automatically"),
              value: _isTaxApplied,
              onChanged: (val) => setState(() => _isTaxApplied = val),
              activeThumbColor: Colors.blueAccent,
            ),

            const SizedBox(height: 30),

            ElevatedButton(
              onPressed: _saveToFirebase,
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.blueAccent,
                minimumSize: const Size(double.infinity, 55),
                shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
```

```
              ),
              child: const Text("SAVE TRANSACTION",
                style: TextStyle(color: Colors.white, fontWeight:
FontWeight.bold)),
            ),
          ],
        ),
      ),
    );
  }
}
```

5.12 Records History (`screens/transaction_list_screen.dart`)



Code:

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

class TransactionListScreen extends StatelessWidget {
  final String filterType;
  final String currencySymbol;
  final double exchangeRate;

  const TransactionListScreen({
    super.key,
    required this.filterType,
    required this.currencySymbol,
    required this.exchangeRate
  });

  // --- DELETE TRANSACTION LOGIC ---
```

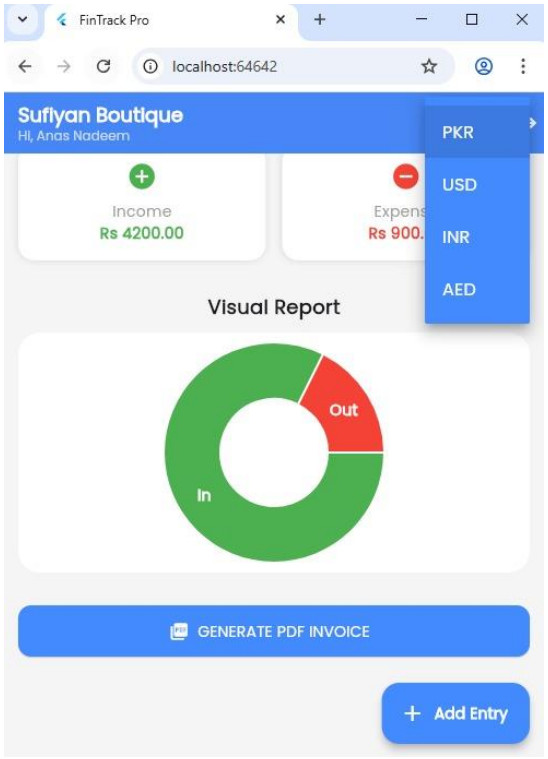
```
  // Removes a specific record from the user's transaction sub-collection in
Firestore
  void _deleteTransaction(BuildContext context, String docId) async {
    String uid = FirebaseAuth.instance.currentUser!.uid;

    await FirebaseFirestore.instance
        .collection('users')
        .doc(uid)
        .collection('transactions')
        .doc(docId)
        .delete();

    // Safety check to ensure the context is still valid after the async database
call
    if (!context.mounted) return;

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Entry deleted successfully")),
    );
  }

  // --- UPDATE / EDIT TRANSACTION DIALOG ---
  // Provides a UI interface to modify existing transaction details
  void _showEditDialog(BuildContext context, String docId, String oldTitle,
double oldAmount) {
    final TextEditingController titleEdit = TextEditingController(text:
oldTitle);
    final TextEditingController amountEdit = TextEditingController(text:
oldAmount.toString());

    showDialog(
      context: context,
      builder: (dialogContext) => AlertDialog(
        title: Text("Edit ${filterType.toUpperCase()}"),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            TextField(
              controller: titleEdit,
              decoration: const InputDecoration(labelText: "Description")
            ),
            TextField(
              controller: amountEdit,
              decoration: const InputDecoration(labelText: "Amount (in Base
Currency)"),
```

```
            keyboardType: TextInputType.number,
          ),
        ],
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(dialogContext),
          child: const Text("Cancel")
        ),
        ElevatedButton(
          onPressed: () async {
            String uid = FirebaseAuth.instance.currentUser!.uid;

            // Updating the transaction document with new validated data
            await FirebaseFirestore.instance
                .collection('users')
                .doc(uid)
                .collection('transactions')
                .doc(docId)
                .update({
              'title': titleEdit.text,
              'amount': double.tryParse(amountEdit.text) ?? 0.0,
            });

            // Closing the dialog after a successful update
            if (!dialogContext.mounted) return;
            Navigator.pop(dialogContext);
          },
          child: const Text("Update"),
        ),
      ],
    ),
  );
}

@override
Widget build(BuildContext context) {
  String uid = FirebaseAuth.instance.currentUser!.uid;

  return Scaffold(
    appBar: AppBar(
      title: Text("${filterType.toUpperCase()} History"),
      backgroundColor: filterType == 'income' ? Colors.green : Colors.red,
      foregroundColor: Colors.white,
    ),
```

```
      body: StreamBuilder<QuerySnapshot>(
        // Streaming filtered data based on transaction type (income/expense)
        stream: FirebaseFirestore.instance
            .collection('users')
            .doc(uid)
            .collection('transactions')
            .where('type', isEqualTo: filterType)
            .snapshots(),
        builder: (context, snapshot) {
          if (snapshot.hasError) return Center(child: Text("Error:
${snapshot.error}"));
          if (!snapshot.hasData) return const Center(child:
CircularProgressIndicator());

          var docs = snapshot.data!.docs;
          if (docs.isEmpty) return Center(child: Text("No $filterType records
found."));

          return ListView.builder(
            itemCount: docs.length,
            itemBuilder: (context, index) {
              var doc = docs[index];
              var data = doc.data() as Map<String, dynamic>;
              String docId = doc.id;

              // Currency conversion logic for real-time display
              double rawAmount = (data['amount'] ?? 0.0).toDouble();
              double convertedAmount = rawAmount * exchangeRate;

              return Card(
                margin: const EdgeInsets.symmetric(horizontal: 15, vertical: 5),
                elevation: 2,
                child: ListTile(
                  // Triggering the edit dialog on tapping the list item
                  onTap: () => _showEditDialog(context, docId, data['title'] ??
"", rawAmount),
                  leading: CircleAvatar(
                    backgroundColor: filterType == 'income' ? Colors.green[100] :
Colors.red[100],
                    child: Icon(
                      filterType == 'income' ? Icons.arrow_downward :
Icons.arrow_upward,
                      color: filterType == 'income' ? Colors.green : Colors.red,
                    ),
                  ),
```

```
                  title: Text(
                    data['title'] ?? "No Title",
                    style: const TextStyle(fontWeight: FontWeight.bold)
                  ),
                  subtitle: Text(
                    data['date'] != null
                      ? (data['date'] as Timestamp).toDate().toString().split('
')[0]
                      : ""
                  ),
                  trailing: Row(
                    mainAxisSize: MainAxisSize.min,
                    children: [
                      // Displaying the converted currency amount with
appropriate symbol
                      Text(
                        "$currencySymbol ${convertedAmount.toStringAsFixed(2)}",
                        style: const TextStyle(
                          fontWeight: FontWeight.bold,
                          fontSize: 15,
                          color: Colors.black87
                        ),
                      ),
                      const SizedBox(width: 8),
                      // Button to initiate the deletion of a transaction record
                      IconButton(
                        icon: const Icon(Icons.delete_outline, color:
Colors.redAccent),
                        onPressed: () => _deleteTransaction(context, docId),
                      ),
                    ],
                  ),
                ),
              );
            },
          );
        },
      ),
    );
  }
}
```

5.13 Visual Analytics UI (`screens/analytics_screen.dart`)



Code:

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:fl_chart/fl_chart.dart';
import '../services/firestore_service.dart';

class AnalyticsScreen extends StatelessWidget {
  // Parameters to handle multi-currency display and real-time conversion
  final String currency;
  final double rate;

  const AnalyticsScreen({super.key, required this.currency, required this.rate});

  @override
  Widget build(BuildContext context) {
    final FirestoreService firestoreService = FirestoreService();

    return Scaffold(
      appBar: AppBar(
        title: Text("Financial Analytics ($currency)"),
        backgroundColor: const Color(0xFF673AB7),
        foregroundColor: Colors.white,
      ),
```

```
      body: StreamBuilder<QuerySnapshot>(
        // Listening to real-time transaction updates from Firestore
        stream: firestoreService.getTransactions(),
        builder: (context, snapshot) {
          if (!snapshot.hasData) return const Center(child:
CircularProgressIndicator());

          double income = 0;
          double expense = 0;

          for (var doc in snapshot.data!.docs) {
            // Applying exchange rate multiplication for currency conversion
            double amt = (doc['amount'] as num).toDouble() * rate;
            if (doc['type'] == 'income') {
              income += amt;
            } else {
              expense += amt;
            }
          }

          double total = income + expense;

          // Determining the correct currency symbol based on selection
          String symbol = currency == "USD" ? "\$" : "$currency ";

          return Padding(
            padding: const EdgeInsets.all(20.0),
            child: Column(
              children: [
                const Text("Income vs Expense", style: TextStyle(fontSize: 20,
fontWeight: FontWeight.bold)),
                const SizedBox(height: 30),
                SizedBox(
                  height: 250,
                  child: PieChart(
                    PieChartData(
                      sections: [
                        // Visual representation of Income percentage
                        PieChartSectionData(
                          value: income,
                          title: total > 0 ? '${((income / total) *
100).toStringAsFixed(1)}%' : '0%',
                          color: Colors.green,
                          radius: 60,
```

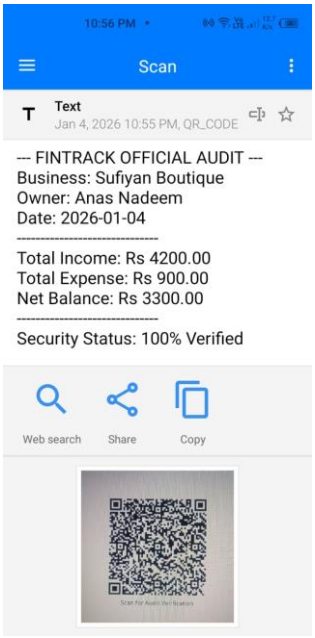
```
                        titleStyle: const TextStyle(color: Colors.white,
fontWeight: FontWeight.bold),
                      ),
                      // Visual representation of Expense percentage
                      PieChartSectionData(
                        value: expense,
                        title: total > 0 ? '${((expense / total) *
100).toStringAsFixed(1)}%' : '0%',
                        color: Colors.red,
                        radius: 60,
                        titleStyle: const TextStyle(color: Colors.white,
fontWeight: FontWeight.bold),
                      ),
                    ],
                  ),
                ),
              ),
              const SizedBox(height: 40),
              // Legend with Dynamic Currency values based on live rates
              _buildStatTile("Total Income",
"$symbol${income.toStringAsFixed(2)}", Colors.green),
              _buildStatTile("Total Expense",
"$symbol${expense.toStringAsFixed(2)}", Colors.red),
              _buildStatTile("Net Savings", "$symbol${(income -
expense).toStringAsFixed(2)}", const Color(0xFF673AB7)),
            ],
          ),
        );
      },
    ),
  );
}

// Reusable widget to display financial statistics in a list format
Widget _buildStatTile(String label, String value, Color color) {
  return Card(
    elevation: 2,
    margin: const EdgeInsets.symmetric(vertical: 8),
    child: ListTile(
      leading: CircleAvatar(backgroundColor: color, radius: 10),
      title: Text(label),
      trailing: Text(value, style: TextStyle(fontWeight: FontWeight.bold,
fontSize: 16, color: color)),
    ),
  );
```

```
  }
}
```

---

6. UNIQUE SELLING POINT (USP): SECURE QR AUDIT

The innovation in FinTrack Pro lies in its **QR-Audit Verification**. Each generated PDF report contains a QR code that stores a full breakdown of the data (Income, Expense, Balance). This allows any auditor or business owner to verify the report's authenticity by scanning it, ensuring the figures have not been manually altered.



d7. CONCLUSION

FinTrack Pro successfully digitizes financial management for small businesses. By combining real-time cloud data with dynamic currency APIs and professional reporting, it provides a scalable solution for modern-day accounting.