






Expense Tracker App - Final Submission

Overview

A comprehensive mobile expense tracking application with Firebase integration, visual analytics, and goal tracking capabilities. The app helps users monitor their spending habits, set budgets, and receive intelligent insights to manage their finances effectively.

Features Implemented

Core Requirements

-  Firebase database integration for online data storage
-  Interactive graphs showing amount spent per category over user-selectable periods
-  Visual display of spending goal achievement over the past month
-  Minimum and maximum spending goals with visual indicators
-  Mobile-optimized interface

Custom Features (2 Required)

Feature 1: Budget RecommendationEngine

- **Description:** Intelligent notification system that alerts users when they approach spending limits.
- **Implementation:** Real-time monitoring of spending against goals with progressive alerts at 75%, 90%, and 100% of the budget.
- **Value:** Helps users stay within budget proactively rather than reactively.

Feature 2: Expense prediction Engine

- **Description:** AI-powered analysis of spending patterns with personalized insights.
- **Implementation:** Analyzes spending trends, identifies peak spending days/times, and provides actionable recommendations.

- **Value:** Helps users understand their spending behavior and make informed financial decisions.

Technical Implementation

Database Structure (Firestore)

```
expenses/  
├── userId/  
│   ├── transactions/  
│   │   ├── transactionId/  
│   │   │   ├── amount: number  
│   │   │   ├── category: string  
│   │   │   ├── description: string  
│   │   │   ├── date: timestamp  
│   │   │   └── type: "expense" | "income"  
│   │   └── goals/  
│   │       ├── categoryId/  
│   │       │   ├── minAmount: number  
│   │       │   ├── maxAmount: number  
│   │       │   └── period: "monthly" | "weekly"  
│   └── profile/  
│       ├── name: string  
│       ├── email: string  
│       └── preferences: object
```

Key Components

1. **TransactionService:** Handles CRUD operations for expenses/income.
2. **GoalService:** Manages spending goals and tracking.
3. **AnalyticsService:** Generates charts and insights.
4. **NotificationService:** Manages smart alerts.
5. **AuthService:** Firebase authentication integration.

Charts and Visualizations

- Category Spending Chart: Interactive pie/bar chart showing spending by category.
- Goal Progress Indicators: Visual progress bars with color-coded status.
- Trend Analysis: Line charts showing spending patterns over time.
- Achievement Dashboard: Visual representation of goal adherence.

Testing Strategy

Unit Tests

- Service layer testing for all Firebase operations.
- Chart data processing validation.
- Goal calculation accuracy tests.

Integration Tests

- Firebase connectivity and data persistence.
- Real-time data synchronization.
- Cross-component communication.

UI Tests

- Navigation flow testing.
- Form validation and submission.
- Chart rendering and interaction.

CI/CD: GitHub Actions Automated Workflows

1. **Build and Test:** Runs on every push/pull request
 - a. Gradle build verification
 - b. Unit test execution
 - c. Code quality checks
2. **Release Build:** Triggered on version tags

- a. Production APK generation
- b. Automated testing suite
- c. Artifact archiving

Configuration Files

- `.github/workflows/build.yml` : Main CI pipeline
- `gradle.properties` : Build configuration
- `app/build.gradle` : Dependencies and build settings

Setup Instructions

Prerequisites

- Android Studio Arctic Fox or later
- Firebase project with Firestore enabled
- Android SDK 21+ (minimum), Target SDK 34

Installation

Add `google-services.json` to the `app/` directory.

- Enable Firestore and Authentication in the Firebase console.
- Configure Firebase security rules.

Build and Run

```
./gradlew build
./gradlew installDebug
```

Video Demonstration Checklist

- App launch and authentication flow.

- Adding new expenses across different categories.
- Setting minimum and maximum spending goals.
- Viewing category spending graphs with period selection.
- Demonstrating goal progress visualization.
- Showing custom features: notifications & pattern analysis.
- Data persistence verification (online storage proof).
- Professional narration explaining each feature.
- Recorded on a physical device with voiceover and responsive UI demonstration.

Architecture Decisions

State Management

- MVVM pattern with LiveData / StateFlow
- Repository pattern for data access
- Dependency injection with Hilt

UI Framework

- Jetpack Compose for modern UI
- Material Design 3 components
- Responsive layouts for various screen sizes

Data Visualization

- MPAndroidChart library for interactive charts
- Custom chart components for goal visualization
- Real-time data binding

Performance Optimizations

- Lazy loading for large transaction lists
- Image caching for category icons
- Offline-first architecture with sync capabilities
- Efficient Firebase queries with pagination

Security Considerations

- Firebase security rules implementation
- User data encryption at rest
- Secure authentication flow
- Input validation and sanitization

Future Enhancements

- Receipt scanning with OCR
- Multi-currency support
- Export functionality (PDF/CSV)
- Family account sharing
- Advanced budgeting features

Contact Information

Developer: [minenhle Dladla _ st10113403]

GitHub: [<https://github.com/VCDN-2025/prog7313-poe-summative-Minenhled>]

Video Demo: [Link to demonstration video]

Last Updated: 9 June 2025

References

Google. (n.d.). *Android developers*. <https://developer.android.com>

Google. (n.d.). *Firebase documentation*. <https://firebase.google.com/docs>

JetBrains. (n.d.). *Kotlin language documentation*. <https://kotlinlang.org/docs/home.html>

Stack Overflow. (n.d.). *Android tag – Stack Overflow*.
<https://stackoverflow.com/questions/tagged/android>