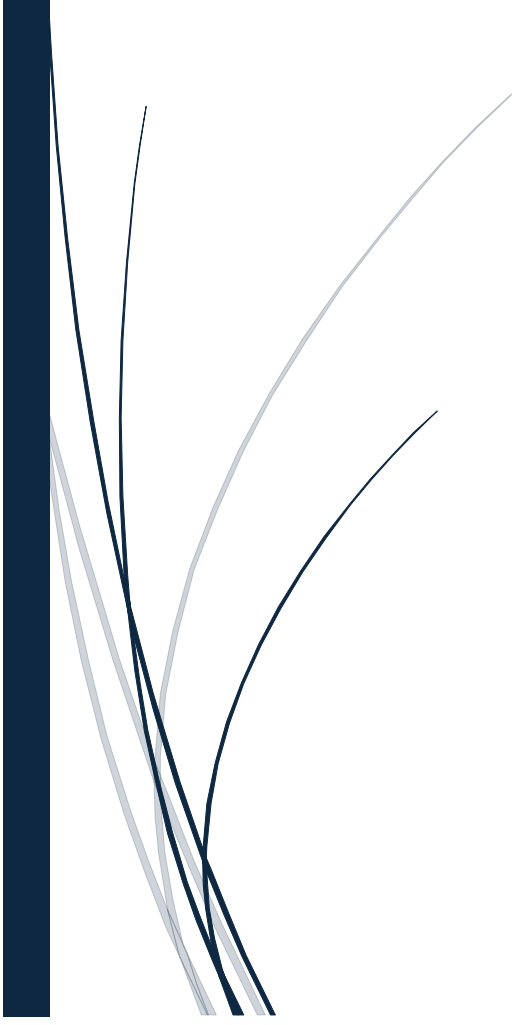
9/26/2024

UserDocumentation

COUNTIFY



OPSC7312 POE

VARSITY COLLEGE

Countify

Countify is a simple Android counter app developed using Kotlin in Android Studio. The app allows users to create and manage custom counters, either by registering an account with their email or by using Google SSO (Single Sign-On). Counters are persisted using Firestore Database, and the app integrates various user preferences, including notifications, theme, and language settings. The app also displays random advice fetched from a third-party API.

Table of Contents

Purpose

Countify was built to provide users with a simple and intuitive way to create, edit, and track multiple counters. It enables users to increment values easily for a variety of personal or professional use cases. Additionally, the app offers settings customization for a personalized user experience, as well as integration with external APIs to enhance functionality.

Features

User Authentication:

- Register using email and password or with Google SSO.

Counters:

- View a list of created counters.
- Add new counters with custom titles, descriptions, and increment values.
- Data is stored and retrieved from Firestore Database.

Settings Page:

- Manage account details (name, email, surname).
- Toggle settings for:
    - Notifications (On/Off).
    - Theme (Dark/Light).
    - Language (English/Afrikaans).
- Random Advice API integration: Displays a new random piece of advice every time the settings page is loaded.

Persistent Data:

- All user preferences (settings, counters) are saved to the Firestore Database and reloaded when the app starts.

Design Considerations

User Interface:

The user interface is designed to be minimalistic and intuitive, with a Floating Action Button (FAB) for creating new counters and easy navigation to the settings page.

User Authentication:

We utilized Firebase Authentication for secure login, offering both email registration and Single Sign-On (SSO) via Google for convenience. User details are stored in Firestore for easy retrieval and display in the settings.

Firestore Database:

All counters and user preferences are stored in Firestore for scalability, real-time updates, and secure storage.

Random Advice API Integration:

The settings page connects to the external Random Advice API, fetching a fresh piece of advice on every load. This adds a fun, interactive element to the app.

Setup and Installation

Prerequisites

- Android Studio (latest version)
- Kotlin configured in the Android project

Steps to Compile and Run

1. **Clone the Repository:**
    - git clone https://github.com/ST10036509/Countify.git
    - cd countify
2. **Build the Project:**
    - Open the project in Android Studio.
    - Sync Gradle files by clicking on the "Sync Now" prompt.
3. **Run the Project:**
    - Click on the "Run" button in Android Studio or press Shift + F10.

Usage

Register/Login:

- Open the app and sign up using an email address or log in using Google SSO.

Manage Counters:

- Once logged in, you'll see a list of counters. To add a new counter, press the FAB and enter the title, description, and increment value. The counter will be saved to Firestore.

Settings:

- Navigate to the settings page to manage preferences. You can toggle notifications, change the theme, and select the language. These preferences are saved in the Firestore and applied on app restart.
- You can also view your account details (name, surname, and email) on this page.
- A random piece of advice will be fetched from the Random Advice API each time the settings page is opened.

GitHub

GitHub Usage:

We used GitHub for source control, allowing our team to push and pull code updates seamlessly. Branching was used to isolate features and bug fixes, ensuring that the main codebase remained stable.

- **Branches:** Each feature was developed on its own branch and merged into the main branch via pull requests.
- **Commits:** We maintained meaningful commit messages, documenting changes clearly.
- **Pull Requests:** Code reviews were done on pull requests to ensure high code quality and collaboration.

GitHub Actions:

We implemented GitHub Actions to automate testing and ensure code consistency. Whenever a new branch was pushed or a pull request was opened, tests would automatically run, helping to catch bugs early.

- **Automated Testing:** Unit tests were written for critical functions, and GitHub Actions would trigger these tests on every push to a branch.
- **CI/CD Pipeline:** The pipeline ensured that every merged change passed tests before being integrated into the main branch.

Further Details

Firestore Integration:

- Firestore was used to store both the user counters and their settings/preferences. Each user has their own document, storing counters (with titles, descriptions, and increment values) and a separate collection for their settings (notifications, theme, language). This separation makes it easy to scale and manage data.

Firebase Authentication:

- Firebase Authentication was used to handle email-based login and Google SSO. Upon successful login, user details are pulled from Firebase to display in the settings page.

API Integration:

- The Random Advice API is a public REST API that returns a random piece of advice. We integrated it into the settings page using a simple HTTP GET request, which is triggered every time the settings page is loaded. The response is parsed and displayed in a TextView.