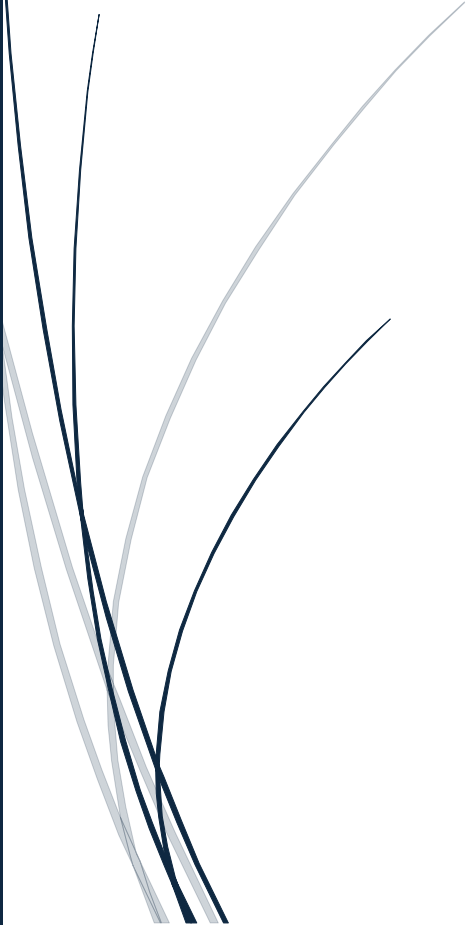


11/1/2024

User Documentation

COUNTIFY



OPSC7312 POE
VARSITY COLLEGE

TABLE OF CONTENTS

Group Members:.....	2
Introdcution.....	3
What is Countify?	3
Purpose.....	3
Features.....	4
User Authentication:	4
Counters:	4
Settings Features:.....	6
Persistent Data:.....	6
Push Notifications:	7
Database Syncing	7
Design Considerations	8
User Interface:.....	8
User Authentication:	8
Firestore Database:	9
Random Advice API Integration:.....	10
Internal Designs	11
Setup and Installation	12
Prerequisites	12
Steps to Compile and Run.....	12
Usage	13
Google Play Console Upload Process	14
Starting application Upload Process	14
Inputting Application Details.....	14
Application Descriptions	15
Internal Testing Configuration.....	15
Bundle Buidling	16
Application Release.....	18
Full Release Errors.....	19
GitHub.....	20
GitHub Usage:	20
GitHub Actions:.....	21

Further Details	23
Firestore Integration:.....	23
Firebase Authentication:.....	23
API Integration:.....	23
References	24

GROUP MEMBERS:

Ethan Schoonbee (PM) – **ST10036509**

Leonard Bester – **ST10026396**

Ross Harper – **ST10048758**

Cameron Brooks – **ST10032014**

INTRODCUTION

WHAT IS 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.

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

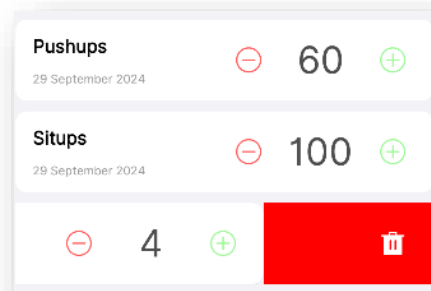
USER AUTHENTICATION:

- Register using email and password or with Google SSO.

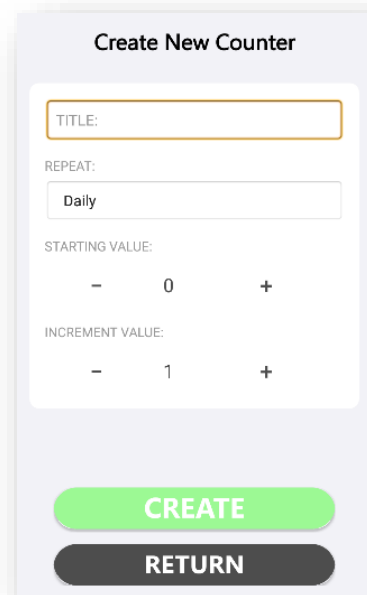


COUNTERS:

- View a list of created counters.

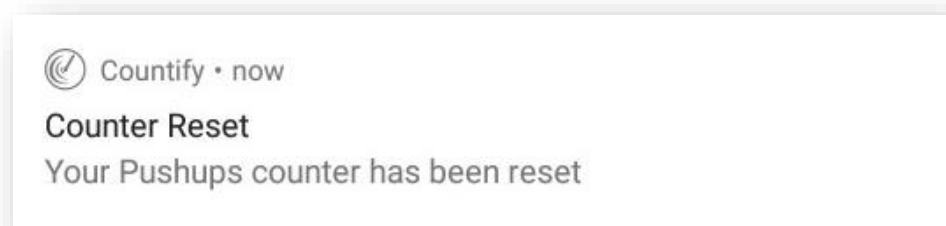


- Add new counters with custom titles, descriptions, and increment values.



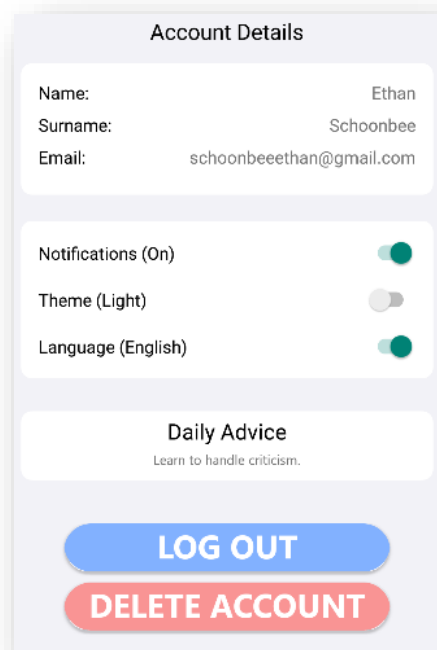
The image shows a mobile app interface for creating a new counter. The title is "Create New Counter". Below the title is a text input field labeled "TITLE:". Underneath is a "REPEAT:" section with a dropdown menu currently set to "Daily". Below that is a "STARTING VALUE:" section with a numeric input field set to "0", flanked by minus and plus buttons. The next section is "INCREMENT VALUE:" with a numeric input field set to "1", also flanked by minus and plus buttons. At the bottom are two buttons: a green "CREATE" button and a dark grey "RETURN" button.

- Data is stored and retrieved from Firestore Database.
- Counters reset to their selected **starting value** after the repeat time has expired.



SETTINGS FEATURES:

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



- The delete account button will delete the user and log out *Countify*.

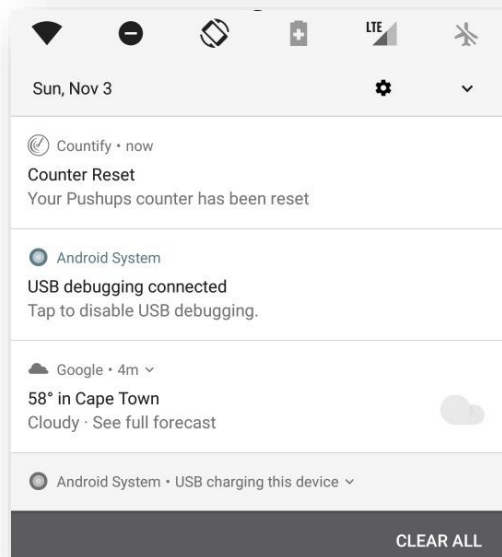
PERSISTENT DATA:

- All user preferences (settings, counters) are saved to the Firestore Database and reloaded when the application boots up.



PUSH NOTIFICATIONS:

- In the local code base, push notifications are built and sent to the user.



Countify notification featuring in the list of notifications on Android device

DATABASE SYNCING

- Users can add a counter while being offline utilising *SQLite* to save local data. (Kreibich, 2010) Once an internet connection is established on the android device, the *SQLite* data will be merged with the **Google Firebase** cloud database.



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.



FAB



Settings Button

USER AUTHENTICATION:

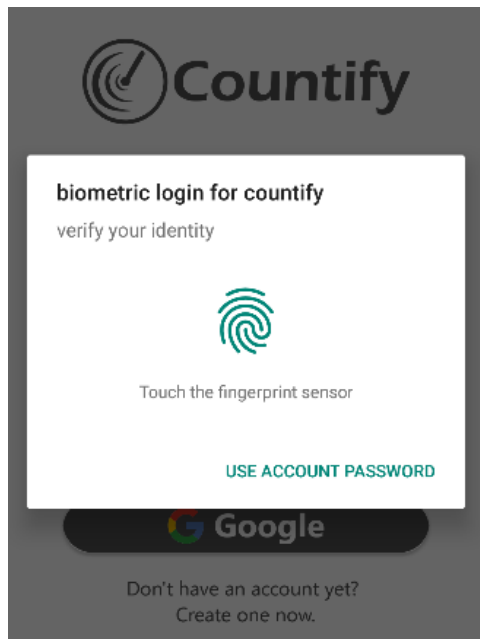
We utilized Firebase Authentication for secure login, offering both email registration and Single Sign-On (SSO) via Google for convenience and Biometrics for secure logging back in (Android). User details are stored in Firestore for easy retrieval and display in the settings.

A login form with a white background and a subtle shadow. It features two input fields: 'EMAIL' with an envelope icon and 'PASSWORD' with a lock icon. Below the fields is a prominent green 'LOGIN' button. Underneath the button is a dark gray button with the Google logo and the text 'Google'. At the bottom, there is a link that says 'Don't have an account yet? Create one now.'

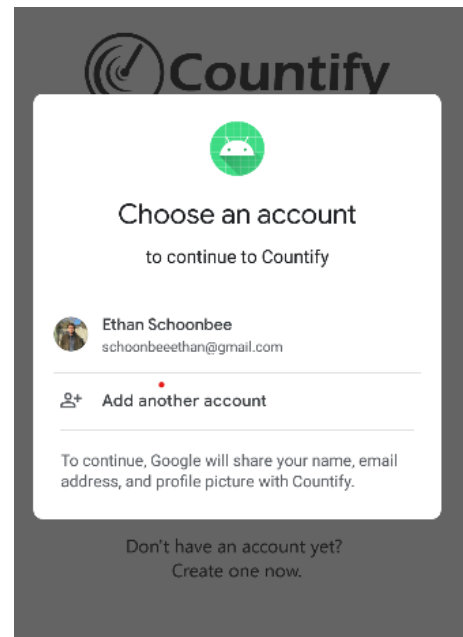
Email & Password Login

A registration form with a white background and a subtle shadow. It features five input fields: 'FIRST NAME' with a person icon, 'LAST NAME' with a person icon, 'EMAIL' with an envelope icon, 'PHONE NUMBER' with a phone icon, and 'PASSWORD' with a lock icon. Below these is a 'CONFIRM PASSWORD' field with a lock icon. A green 'REGISTER' button is positioned below the fields. Underneath is a dark gray button with the Google logo and the text 'Google'. At the bottom, there is a link that says 'Already have an account? Login here.'

User Registration



Biometric Login



Google SSO Register and Login

FIRESTORE DATABASE:

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



Search by email address, phone number, or user UID					Add user	↺	⋮
Identifier	Providers	Created ↓	Signed In	User UID			
mygoogleemail@gmail...	✉	Sep 29, 2024	Sep 29, 2024	shtFBJkb5bYUMWjPloyAb5sf...			
tester1@gmail.com	✉	Sep 29, 2024	Sep 29, 2024	Ccpf2egq1KhSjk9ZOWtJYrTnj...			
test4@gmail.com	✉	Sep 27, 2024	Sep 27, 2024	WrDwi2MyXNbkbV8iJYd5p1T...			
schoonbeeethan@gmai...	🌐	Sep 25, 2024	Sep 27, 2024	6CCyy24x5kY7D9CtCvnOHh4...			
test2@gmail.com	✉	Sep 22, 2024	Sep 28, 2024	ej4fXRTydoXcEiufG0NgGYzJX...			
joesoap@gmail.com	✉	Sep 21, 2024	Sep 21, 2024	J1cgDiWNxdQ1BcoBIEFuw9v...			
test@gmail.com	✉	Sep 20, 2024	Sep 22, 2024	lTrSxIBYJHah5YOFKgv2P0nyT...			

Firestore Authentication

(default)	counters	1xJYdD5LDtLojcp5Phxs
+ Start collection	+ Add document	+ Start collection
counters >	1xJYdD5LDtLojcp5Phxs >	+ Add field
counters_tests	8pK9JhQWDHCjaUSWKGIO	count: 0
users	Ms8cdIKqC5RlEmdYlp1I	createdTimestamp: 1727554397302
	VrYb4PBteflcbC0garCY	incrementValue: 1
	x3iVj2BYfhSV2Nf6PCZd	name: "oui oui Mon ami"
		repetition: "Daily"
		userId: "ej4fXRTydoXcEiufG0NgGYzJXMn1"

Firestore NoSQL Database

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.

Daily Advice

If it still itches after a week, go to the doctors.

INTERNAL DESIGNS

FRAGMENTED PAGES:

The primary expression of forms and user interfaces for *Countify* is **fragments**. Fragments are modular forms that allow for more flexible UI design which allows for a diverse range of phone sizes to make use of the user interfaces.

Instead of creating and swapping to an entire new form, a **fragment** will populate will be opened in the main form. One could think of it as being contained inside the main form and swapping fragments allows for the same for to display while the user interface, along with the associated logic, can change to different functional pages. (Wilson, 2016)

NOTIFICATION CHANNELS & CLASSES:

The push notifications in *Countify*, makes use of a **notification channel** within android in the **Main Activity** of the application on start-up.

A **notification compact.builder** is used and structured within it's own class as a service. This class builds the required notification giving it an identification number, a title, an icon and a content message. This class then uses the **notification channel** to send to the android mobile device which will display the *Countify* notification in the notifications list. (Nudelman, 2013)

SETUP AND INSTALLATION

PREREQUISITES

- Android Studio (latest version) (Andorid, 2024)
- Kotlin configured in the Android project (JetBrains, 2024)

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. (Google, 2024)

Manage Counters:

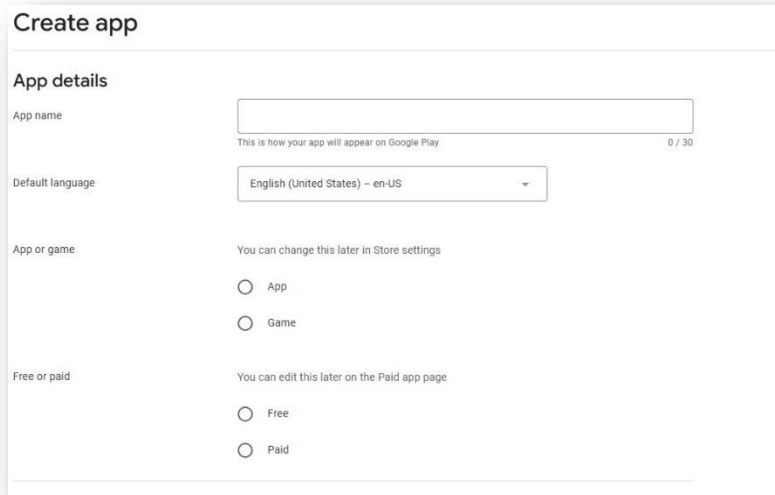
- Once logged in, you'll see a list of counters. To add a new counter, press the FAB and enter the title, description, and increment value. The counter will be saved to Firestore. Counters can be deleted but swiping left on the counter you wish to remove.

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.

GOOGLE PLAY CONSOLE UPLOAD PROCESS

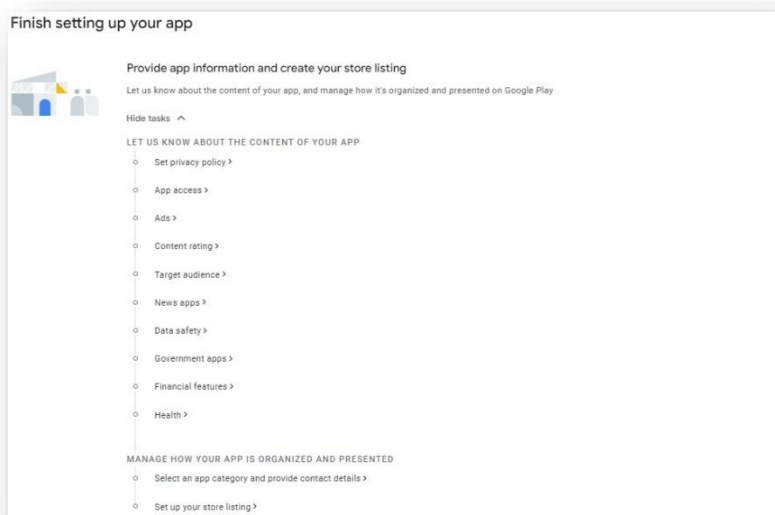
STARTING APPLICATION UPLOAD PROCESS



The screenshot shows the 'Create app' screen in the Google Play Console. It is titled 'Create app' and has a sub-header 'App details'. There are four main sections: 'App name' with a text input field and a character count '0 / 30'; 'Default language' with a dropdown menu set to 'English (United States) - en-US'; 'App or game' with two radio buttons, 'App' and 'Game'; and 'Free or paid' with two radio buttons, 'Free' and 'Paid'. Each section has a brief instruction on how to change the settings later.

The **first step** is to initialise the upload process by filling in the application name, game status and purchase status.

INPUTTING APPLICATION DETAILS



The screenshot shows the 'Finish setting up your app' screen in the Google Play Console. It is titled 'Finish setting up your app' and has a sub-header 'Provide app information and create your store listing'. Below this, there is a list of tasks to complete, organized into two sections: 'LET US KNOW ABOUT THE CONTENT OF YOUR APP' and 'MANAGE HOW YOUR APP IS ORGANIZED AND PRESENTED'. The tasks include 'Set privacy policy', 'App access', 'Ads', 'Content rating', 'Target audience', 'News apps', 'Data safety', 'Government apps', 'Financial features', 'Health', 'Select an app category and provide contact details', and 'Set up your store listing'.

This step requires the application **details and content** like privacy policy, Content rating, targeted audience and the category of the application be entered appropriately according to the content of *Countify*.

APPLICATION DESCRIPTIONS

← Store listings

Create default store listing

Edit your app's name, icon, screenshots and more to present how your app looks to users on Google Play. [Show more](#)

Default – English (United States) – en-US Manage translations ▾

* – Required fields. Enter all fields in English (United States) – en-US

Listing assets

Check the [Metadata policy](#) and [Help Center guidance](#) to avoid common issues with your store listing. Review all [program policies](#) before submitting your app.

If you're eligible to [provide advance notice](#) to the app review team, contact us before publishing your store listing.

App name * 8 / 30
This is how your app will appear on Google Play

Short description * 0 / 80
A short description for your app. Users can expand to view your full description.

Full description * 0 / 4000

Graphics

Entering the **description** of the application. Listing its **purpose and features** and explaining the functionality of *Countify*.

INTERNAL TESTING CONFIGURATION

Internal testing

Quickly share your app now for initial quality checks

Before you've finished setting up your app, you can quickly distribute builds to your own device, or to a small group of your own trusted users. Builds are normally available to users within seconds of being added in Play Console. This is optional, and you'll still need to run a closed test before publishing to everyone in production. [Learn more](#)

2 of 3 complete ^

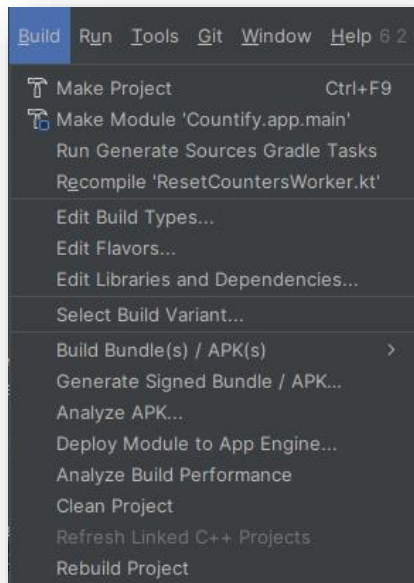
- ✓ Select testers

CREATE AND ROLL OUT A RELEASE

- ✓ Create a new release
- Preview and confirm the release >

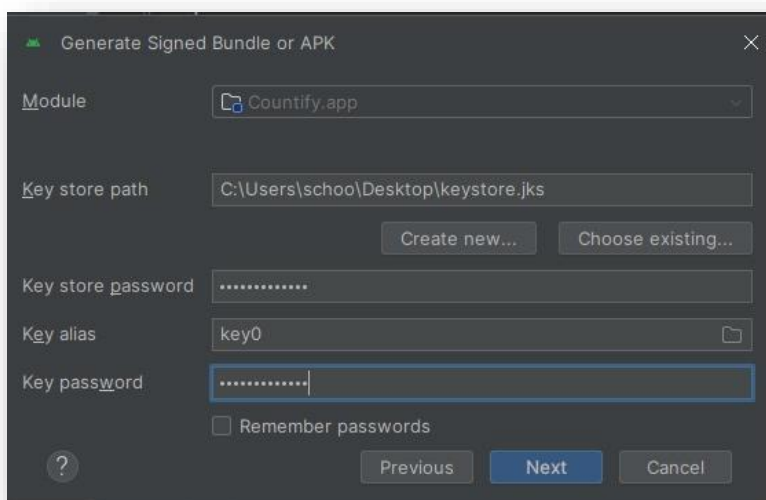
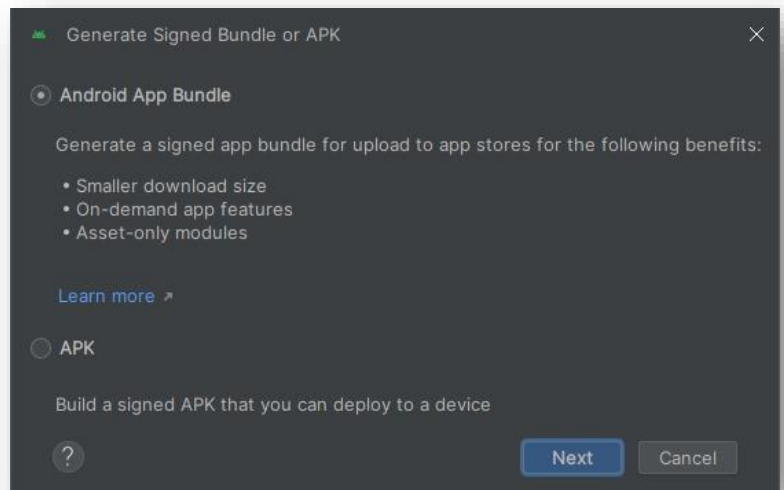
In this step, 14 **batch tester** are selected for testing the application for quality checks.

BUNDLE BUILDING



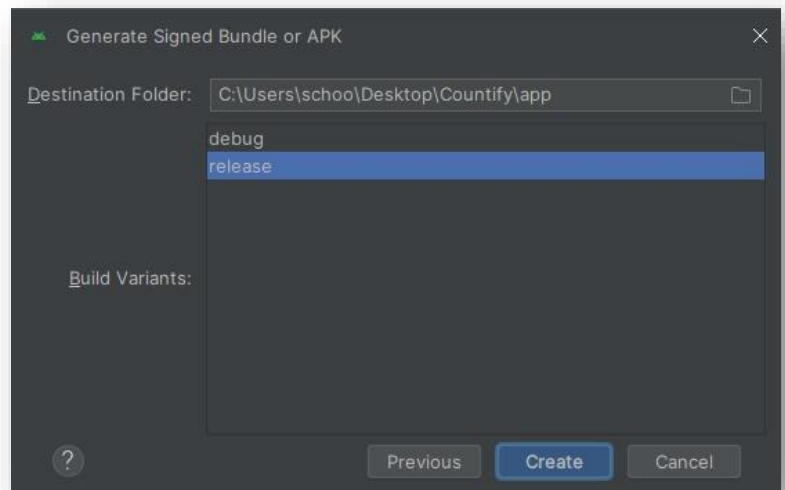
In android studios under the build drop down, to start building a **single bundle**, the **Generate Signed Bundle** option is clicked

Then the **Android App Bundle** option is picked as the advantages allow for a smaller download size, On-demand app features and Asset-only modules to be enabled



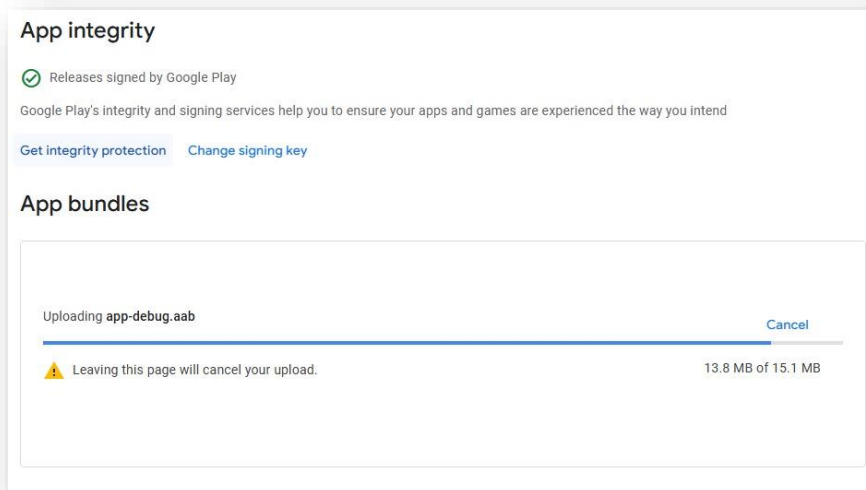
Then the key is generated by entering the input fields along with a key password.

Then the last step for making a single bundle is to select the **release** option as this will be the latest running version of the application.



APPLICATION RELEASE

Once the application's single bundle release is ready it can be selected to be uploaded.



File type	Version	API levels	Target SDK	Screen layouts	ABIs	Required features	
App bundle	1 (1.0)	26+	34	4	All	1	→

Release details




Release name

Latest releases

Release	Latest version	Track	Release status	Last updated	Countries / regions	Install base	
Countify	1	Internal testing	Draft	-	-	-	→

1 app [Create app](#)

Filter by **All**

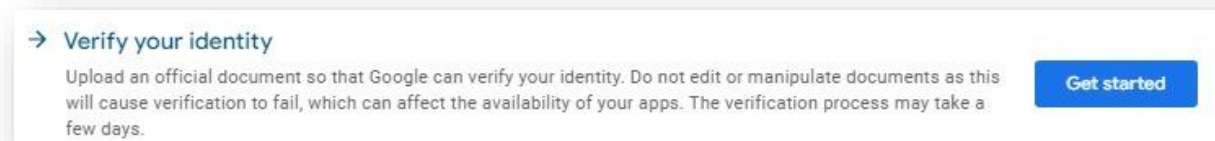
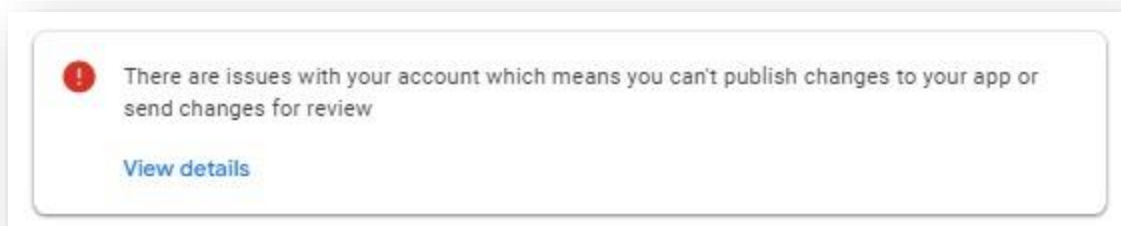
App	Installed audience	App status	Update status	Last updated	
 Countify st10036509.countify	0	Draft		Nov 4, 2024	  View app →

Show rows: 10 1 - 1 of 1 [|<](#) [<](#) [>](#) [>|](#)

Then we can see that the application *Countify* with its unique icon, has been listed with the latest version of the source code.

FULL RELEASE ERRORS

In other circumstances, *Countify* would be on the **Google Play Store**, but the error listed below prevents this from happening. The cause of the error is the lack of a verified account and to verify the account to resolve this error would reach past the hand-in date for the **OPSC POE**.



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. (GitHub , 2024)

- **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.

ST10048758 Merge pull request #65 from ST10036509/OF-119 3ad1c8c · yesterday 151 Commits

.github/workflows	Updated JAVA Version to v17	4 days ago
app	Updates to classes which contain counter model (FirestoreSe...	yesterday
gradle	Login & Register Unit Tests	2 days ago
.gitignore	Reformatted Warning for Login, Register, and SSO Pages & ...	last week
.gitignore.backup	Initial commit of Andoird Studio project	2 weeks ago
LICENSE	Initial commit	2 weeks ago
README.md	Update README.md	3 days ago
build.gradle.kts	Initial Authentication Service Developed (Untested)	last week
gradle.properties	Initial commit of Andoird Studio project	2 weeks ago
gradlew	Initial commit of Andoird Studio project	2 weeks ago
gradlew.bat	Initial commit of Andoird Studio project	2 weeks ago
settings.gradle.kts	Initial commit of Andoird Studio project	2 weeks ago
user-documentation.pdf	Added UserDocumentation to attach to readme	3 days ago

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. (GitHub, 2024)
- **CI/CD Pipeline:** The pipeline ensured that every merged change passed tests before being integrated into the main branch. (GitHub, 2024)
- As of 26/09/2024 the **Mockito** (Mockito, 2023) package used for mocking **AndroidX** fragments and classes had a depreciated package (**Byte Buddy**) (Winterhalter, 2024) which it relied on for simulating android environment with **Maven** so none of our tests could be run or pass.

✓ Merge pull request #51 from ST10036509/ST10032014-ReadMe Android CI #16: Commit c3c99f4 pushed by ST10032014	dev	3 days ago 4m 31s	...
✓ Update README.md Android CI #15: Pull request #51 opened by ST10032014	ST10032014-ReadMe	3 days ago 4m 17s	...
✓ Merge pull request #50 from ST10036509/ST10032014-patch-1 Android CI #14: Commit 9051d75 pushed by ST10032014	dev	3 days ago 4m 14s	...
✓ Added UserDocumentation to attach to readme Android CI #13: Pull request #50 opened by ST10032014	ST10032014-patch-1	3 days ago 4m 53s	...
✓ Merge pull request #47 from ST10036509/OF-89 Android CI #12: Commit 1d3b8bf pushed by ST10026396	dev	4 days ago 4m 11s	...
✓ OF-89 cleaned up counter creation ui Android CI #11: Pull request #47 opened by ST10026396	OF-89	4 days ago 4m 35s	...
✓ Merge pull request #46 from ST10036509/UPDATE_JAVA_17 Android CI #10: Commit 24477c1 pushed by ST10036509	dev	4 days ago 4m 28s	...
✓ Updated JAVA Version to v17 Android CI #9: Pull request #46 opened by ST10036509	UPDATE_JAVA_17	4 days ago 5m 10s	...

<div>✖ Merge pull request #65 from ST10036509/OF-119</div> <div>Android CI #46: Commit 3ad1c8c pushed by ST10048758</div>	dev	<div>yesterday</div> <div>🕒 4m 48s</div> <div>...</div>
<div>✖ Updates to classes which contain counter model (FirestoreSer...</div> <div>Android CI #45: Pull request #65 opened by ST10048758</div>	OF-119	<div>yesterday</div> <div>🕒 5m 4s</div> <div>...</div>
<div>✖ Merge pull request #64 from ST10036509/OF-114</div> <div>Android CI #44: Commit 9fabf05 pushed by ST10048758</div>	dev	<div>yesterday</div> <div>🕒 3m 30s</div> <div>...</div>
<div>✖ Changed counter model and changed CounterAdapter and C...</div> <div>Android CI #43: Pull request #64 opened by ST10048758</div>	OF-114	<div>yesterday</div> <div>🕒 3m 57s</div> <div>...</div>
<div>✖ Merge pull request #63 from ST10036509/OF-118</div> <div>Android CI #42: Commit 02aa7e4 pushed by ST10026396</div>	dev	<div>2 days ago</div> <div>🕒 4m 11s</div> <div>...</div>
<div>✖ OF-118</div> <div>Android CI #41: Pull request #63 opened by ST10026396</div>	OF-118	<div>2 days ago</div> <div>🕒 4m 3s</div> <div>...</div>
<div>✖ Merge pull request #62 from ST10036509/OF-116</div> <div>Android CI #40: Commit 58f5ab3 pushed by ST10026396</div>	dev	<div>2 days ago</div> <div>🕒 4m 0s</div> <div>...</div>
<div>✖ OF-116</div> <div>Android CI #39: Pull request #62 opened by ST10026396</div>	OF-116	<div>2 days ago</div> <div>🕒 4m 6s</div> <div>...</div>

Byte Buddy could not instrument all classes within the mock's type hierarchy

This problem should never occur for javac-compiled classes. This problem has been observed for classes that are:

- Compiled by older versions of scalac
- Classes that are part of the Android distribution
 - at org.mockito.internal.creation.bytebuddy.InlineBytecodeGenerator.triggerRetransformation(InlineBytecodeGenerator.java:280)
 - at org.mockito.internal.creation.bytebuddy.InlineBytecodeGenerator.mockClass(InlineBytecodeGenerator.java:213)
 - at org.mockito.internal.creation.bytebuddy.TypeCachingBytecodeGenerator.lambda\$mockClass\$0(TypeCachingBytecodeGenerator.java:47)
 - at net.bytebuddy.TypeCache.findOrInsert(TypeCache.java:168)
 - at net.bytebuddy.TypeCache\$WithInlineExpunction.findOrInsert(TypeCache.java:399)
 - at net.bytebuddy.TypeCache.findOrInsert(TypeCache.java:190)
 - at net.bytebuddy.TypeCache\$WithInlineExpunction.findOrInsert(TypeCache.java:410)
 - at org.mockito.internal.creation.bytebuddy.TypeCachingBytecodeGenerator.mockClass(TypeCachingBytecodeGenerator.java:40)
 - at org.mockito.internal.creation.bytebuddy.InlineDelegateByteBuddyMockMaker.createMockType(InlineDelegateByteBuddyMockMaker.java:391)

Logs showing Byte Buddy failing

ACCESS THE GITHUB REPO [HERE](#).

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:

- We implemented two of Firebase's REST APIs utilising their **Firebase Authentication** service accepting either **Firebase Tokens** or **OAuth 2.0 Tokens** and their NoSQL database service **Firestore** for handling RESTful remote cloud data storage. (Google, 2024) (Google, 2024)
- 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. (Kiss, 2024)
- Access the API [here](#).

REFERENCES

- Andorid. (2024). *Android Studio*. Retrieved from Developers:
https://developer.android.com/studio?gad_source=1&gclid=CjwKCAjw9eO3BhBNEiwAoc0-jSipK5gLtkil_1jT9epYd5mZyhSqpaAoNOGtZe7vP0dEKQLFLkGBSgRoCjawQAvD_BwE&gclsrc=aw.ds
- Android. (n.d.). *BiometricPrompt*. Retrieved from Developers:
<https://developer.android.com/reference/androidx/biometric/BiometricPrompt>
- GitHub . (2024). *About Git*. Retrieved from GitHub Docs:
<https://docs.github.com/en/get-started/using-git/about-git>
- GitHub. (2024). *Automating Projects using Actions*. Retrieved from GitHub Docs:
<https://docs.github.com/en/issues/planning-and-tracking-with-projects/automating-your-project/automating-projects-using-actions>
- GitHub. (2024). *The complete CI/CD solution*. Retrieved from GitHub:
<https://github.com/solutions/ci-cd>
- Google. (2024). *Firebase Database REST API*. Retrieved from Firebase:
<https://firebase.google.com/docs/reference/rest/database>
- Google. (2024). *Single sign-on* . Retrieved from Google Cloud:
<https://cloud.google.com/architecture/identity/single-sign-on>
- Google. (2024). *Using the Firestore REST API - Authentication and authorization*. Retrieved from Google Cloud: <https://cloud.google.com/firestore/docs/use-rest-api#:~:text=For%20authentication%2C%20the%20Firestore%20REST,requests%20from%20your%20application's%20users.>
- JetBrains. (2024). *Kotlin - Concise. Multiplatform. Fun*. Retrieved from Kotlinlang:
<https://kotlinlang.org/>
- Kiss, T. (2024). *Advice Slip JSON API*. Retrieved from Advice Slip:
<https://api.adviceslip.com/>
- Kreibich, J. (2010). *Using SQLite*. Sebastopol: O'Reily Media, Inc.
- Mockito . (2023). *How do I drink it?* Retrieved from Mockito : <https://site.mockito.org/>

Nudelman, G. (2013). *Android Design Patterns: Interaction Design Solutions for Developers*. Indianapolis: John Wiley & Sons, inc.

Wilson, J. (2016). *Creating Dynamic UIs with Android Fragments*. Birmingham: Packt Publishing Ltd.

Winterhalter, R. (2024). *Byte Buddy (without Dependencies)*. Retrieved from Maven Repository: <https://mvnrepository.com/artifact/net.bytebuddy/byte-buddy>