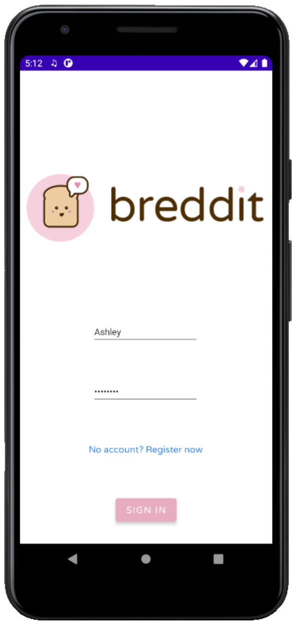
CS330 Final Project

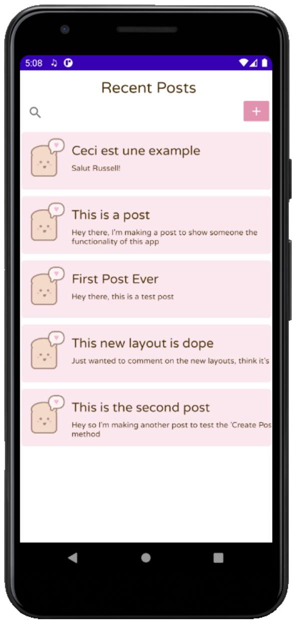
Liam Barrack - 002240046
Ashley Baker - 002223047

I. Basic Features

1. Configuration Change Handling

Our app handles screen rotations and has separate layout files that are optimized for horizontal screen orientation.





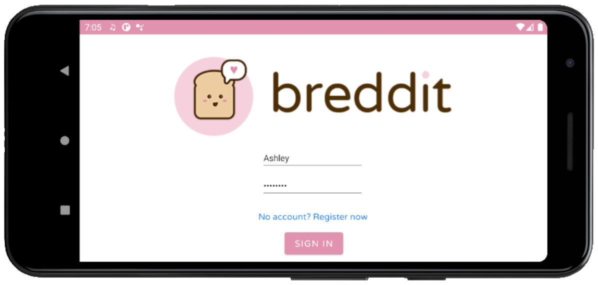
Ceci est une example
Salut Russell!

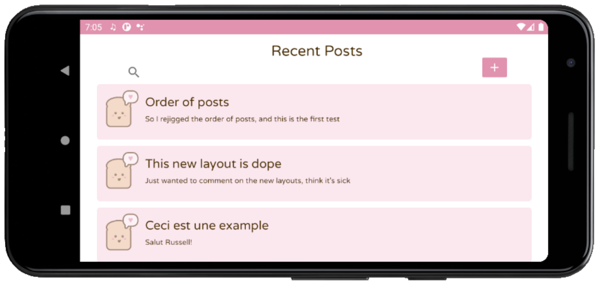
This is a post
Hey there, I'm making a post to show someone the functionality of this app

First Post Ever
Hey there, this is a test post

This new layout is dope
Just wanted to comment on the new layouts, think it's

This is the second post
Hey so I'm making another post to test the 'Create Pos method





Order of posts
So I rejigged the order of posts, and this is the first test

This new layout is dope
Just wanted to comment on the new layouts, think it's sick

Ceci est une example
Salut Russell!

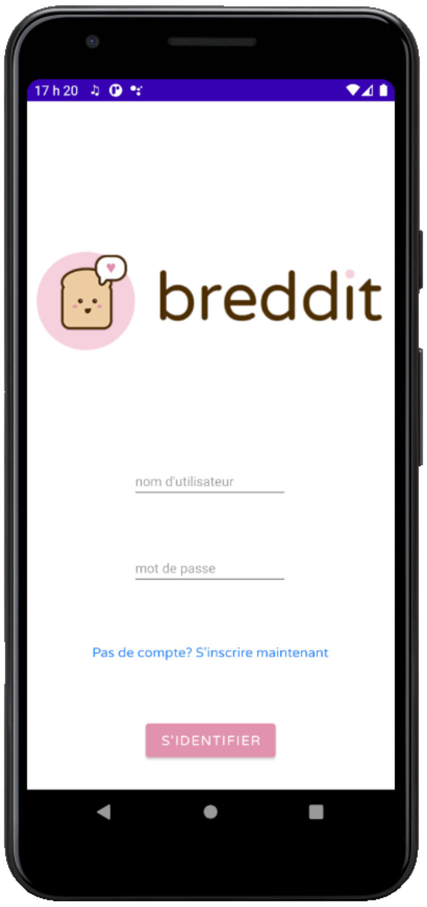
We kept persistent data as static members of MainActivity so that between instances of MainActivity, the data was preserved.

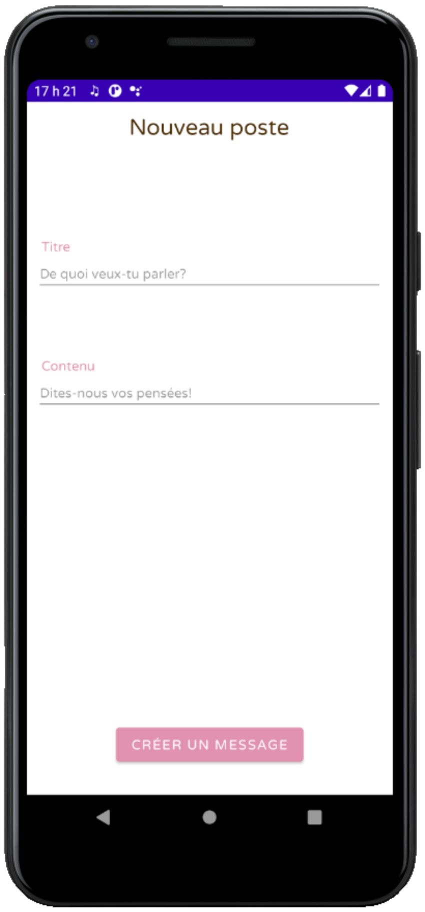
MainActivity.kt

```
companion object{
    var postAdapter = PostsAdapter(mutableListOf())
    lateinit var self: MainActivity
    var hasloggedin = false
    var me = "NULL"
    var currentQuery = ""
}
```

2. Localization

Our app is localized for both English (default) and French. The French localization will apply to any regional French (eg. Canadian French) as regional differences are unlikely in UI elements. Naturally, the localization does not apply to the actual forum posts.





3. Persistence

We used Firestore to store the posts and their respective comments, all of which are fetched in a thread to keep the UI up to date.

MainActivity.kt

```kotlin
Thread{
   while(true){
       if (currentQuery == ""){
           FirestoreTools.downloadPosts(postAdapter)
           Thread.sleep(500)
       }
   }
}.start()
"
   var currentQuery = ""
}
```

FirestoreTools.kt

```kotlin
fun downloadPosts(adapter: PostsAdapter){
   withcollection("allposts"){ collection ->
       collection.forEach start@{ doc ->
           withdocument(doc.get("id").toString(), "info"){ info ->
               for (post in adapter.posts){
                   if (post.id == doc.get("id").toString()){
                       return@withdocument
                   }
               }
               adapter.addpost(Post(
                   info .get("id").toString(),
                   info .get("userid").toString(),
                   info .get("op").toString(),
                   info .get("title").toString(),
                   info .get("content").toString(),
                   Timestamp.now()
               ))
           }
       }
   }
}
```

A separate thread fetches comments, but only runs while you are looking at the post.

ViewPostActivity.kt

```kotlin
Thread{
   while (running){
       FirestoreTools.downloadComments(adapter)
       Thread.sleep(500)
   }
}.start()
```

FirestoreTools.kt

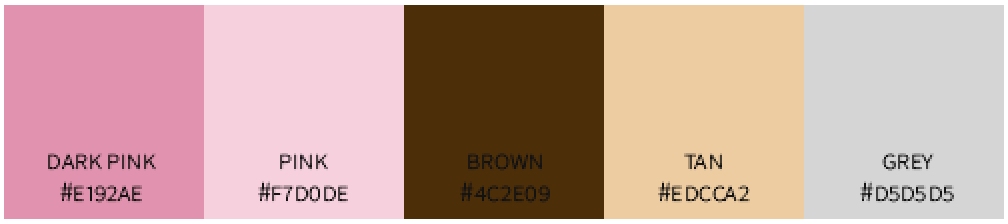
```kotlin
fun downloadComments(adapter: CommentsAdapter){
   withcollection(ViewPostActivity.post.id){ collection ->
       collection.forEach start@{ comment ->
           if (comment.id == "info") return@start
           for (com in adapter.comments){
               if (comment.get("id").toString() == com.id){
                   return@start
               }
           }
           val final = Comment(
               comment.get("id").toString(),
               comment.get("poster").toString(),
               comment.get("content").toString(),
               Timestamp.now()
           )
           adapter.addcomment(final)
           ViewPostActivity.post.comments.add(final)
       }
   }
}
```

4. Custom Styles and Themes

We created a custom theme based on our logo colours. Our sweet mascot KB was designed by Ashley.

Themes.xml

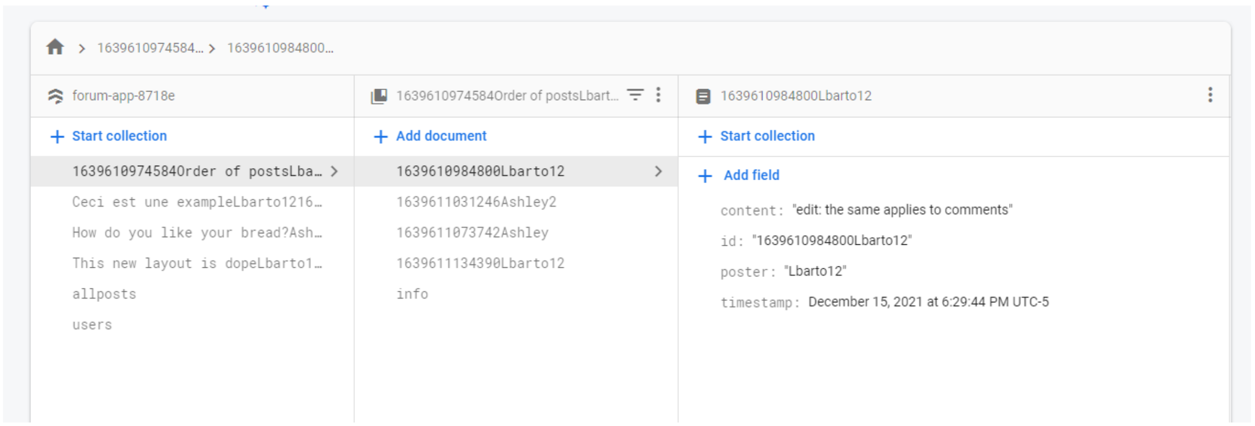
```xml
<resources xmlns:tools="http://schemas.android.com/tools">
   <!-- Base application theme. -->
   <style name="Theme.BredditLight"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
       <!-- Primary brand color. -->
       <item name="colorPrimary">@color/pink</item>
       <item name="colorPrimaryVariant">@color/dark_pink</item>
       <item name="colorOnPrimary">@color/white</item>
       <!-- Secondary brand color. -->
       <item name="colorSecondary">@color/grey</item>
       <item name="colorSecondaryVariant">@color/dark_pink</item>
       <item name="colorOnSecondary">@color/black</item>
       <!-- Status bar color. -->
       <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
       <!-- Customize your theme here. -->
   </style>
</resources>
```



II. Advanced Features

1. Cloud Firestore

Firestore stores each post as a collection, and each document in the collection is a different comment. References to the posts are stored in ‘allposts’ to allow for iteration over collections when reading them in.



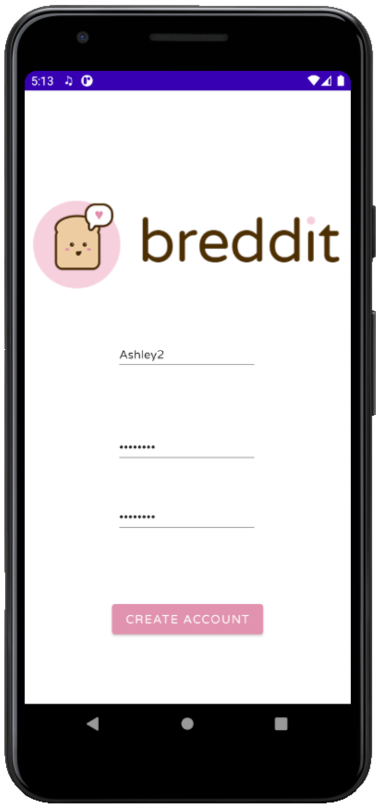
```
1639610974584Order of postsLba…    1639610984800Lbarto12     content: "edit: the same applies to comments"
Ceci est une exampleLbarto1216…    1639611031246Ashley2      id: "1639610984800Lbarto12"
How do you like your bread?Ash…    1639611073742Ashley       poster: "Lbarto12"
This new layout is dopeLbarto1…    1639611134390Lbarto12     timestamp: December 15, 2021 at 6:29:44 PM UTC-5
allposts                           info
users
```

2. Login and Authentication

We implemented a very basic login system that stores user credentials in the users collection. We are aware that this is not safe and that we should never store passwords in plain text.

FirestoreTools.kt

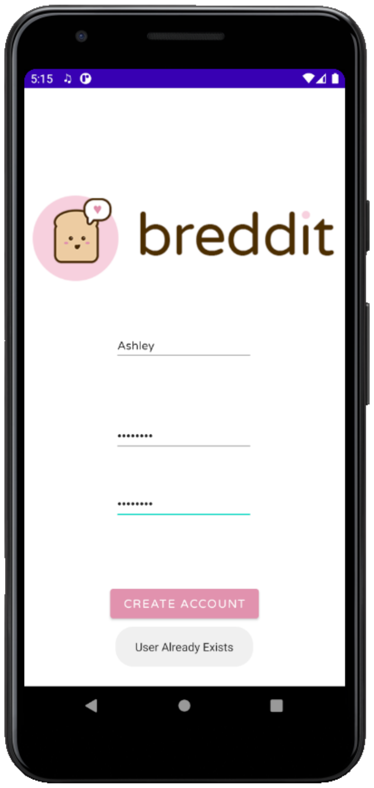
```kotlin
fun createUser(username: String, password: String){
   put("users", username, hashMapOf(
       "username" to username,
       "password" to password
   ))
}

fun userExists(username: String, callback: (result: Boolean) -> Unit){
   withcollection("users"){ users ->
       users.forEach { user ->
           if (user.get("username").toString() == username){
               callback.invoke(true)
               return@withcollection
           }
       }
       callback.invoke(false)
   }
}

fun credentialsValid(username: String, password: String, callback: (result:
Boolean) -> Unit){
   withcollection("users"){ users ->
       users.forEach { user ->
           if (user.get("username").toString() == username &&
               user.get("password").toString() == password){
               MainActivity.me = username
               callback.invoke(true)
               return@withcollection
           }
       }
       callback.invoke(false)
   }
}
```



Ashley2

••••••••

••••••••

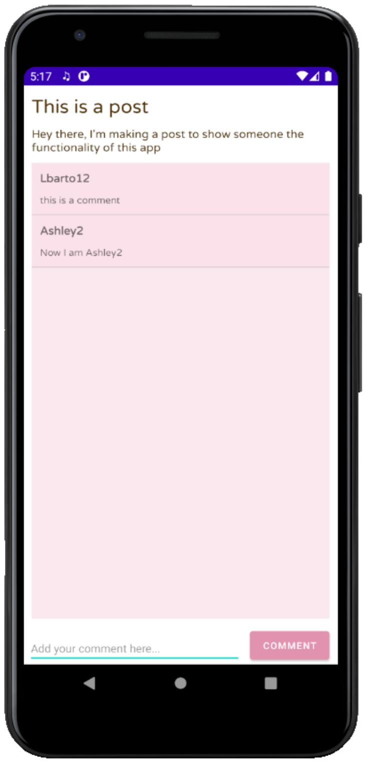


Ashley

••••••••

••••••••

User Already Exists



This is a post

Hey there, I'm making a post to show someone the functionality of this app

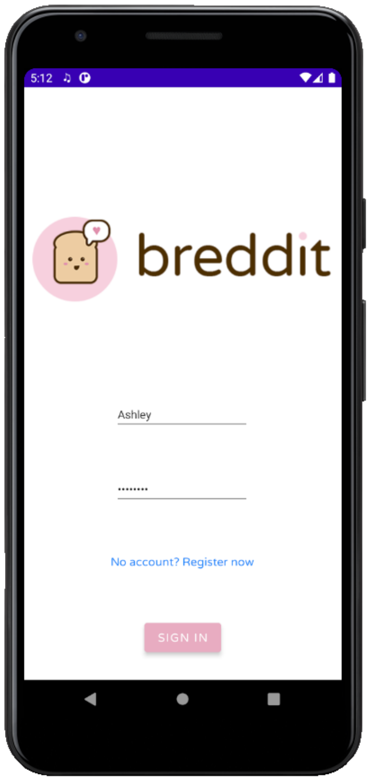
Lbarto12

this is a comment

Ashley2

Now I am Ashley2

Add your comment here...



Ashley

••••••••