

ADVANCED ANDROID APP DEVELOPMENT

(Project Semester January-May 2024)

Chat Application

Submitted by

Shaik Mahammad Rafi

Registration No 12013757

Programme and Section KO203

Course Code CSE227

Under the Guidance of

Dr. Subhita (20260)

Discipline of CSE/IT

Lovely School of Computer Science of Engineering

Lovely Professional University, Phagwara



DECLARATION

I, SHAIK MAHAMMAD RAFI, student of Computer Science & Engineering under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 17-04-2024

Signature: **Shaik Mahammad Rafi**

Registration.no.**12013757**

Name of the student: **Shaik Mahammad Rafi**

CERTIFICATE

This is to certify that SHAIK MAHAMMAD RAFI bearing Registration no. 12013757 has completed CSE227 project titled, Recipe application. To the best of my knowledge, the present work is the result of his/her original development, effort and study.

Signature and Name of the Supervisor

Designation of the Supervisor

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab.

Date: - 17-04-2024

ACKNOWLEDGEMENT

Primarily I'd thank God for being able to complete my project with success. Then I'd like to thank my mentor **Dr. Subhita (20260)**, whose valuable guidance has been the ones that helped me patch this project and make it full proof success in contribution towards the completion of this project.

Finally, I'd rather thanks to **Lovely Professional University**, who gave me this golden opportunity to learn many new things, to learn another aspect of life.

- Shaik Mahammad Rafi

CONTENTS:

Sr No.	Title	Page No.
1	Introduction	6
2	Objectives	7
3	Topics Covered in this project	8
4	ScreenShots	9
4	Functionalities	11
3	References	21
4	Bibliography	22

INTRODUCTION:

Introducing an end-to-end encrypted chat application built in Kotlin, designed to prioritize user privacy and security without compromising on usability. Leveraging modern cryptographic techniques, this application ensures that only the intended recipient can decrypt and read messages, providing a secure channel for communication.

Using Kotlin's robust features and libraries, the application implements strong encryption protocols such as AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman) to encrypt messages before transmission. Each user is assigned a unique key pair, ensuring confidentiality and integrity of messages exchanged.

Furthermore, the application employs key exchange mechanisms like Diffie-Hellman to securely negotiate encryption keys between users, minimizing the risk of eavesdropping or interception by malicious actors.

User experience is paramount, with a sleek and intuitive interface facilitating seamless communication while transparently handling encryption and decryption in the background. End-to-end encryption is the cornerstone of this application, empowering users to communicate freely and confidently, knowing their conversations are shielded from unauthorized access or surveillance.

OBJECTIVES:

The main objective of our recipe application is to empower users to explore, discover, and create culinary delights effortlessly. We aim to provide a user-friendly platform where individuals can access a diverse range of recipes, cooking videos, and culinary inspiration, enhancing their cooking experience and fostering a sense of culinary adventure and creativity.


- 1) Enhanced User Experience: Ensure that users have a seamless and enjoyable experience browsing, searching, and accessing recipes within the application. User can read the trending.
- 2) Increased Engagement: Encourage users to engage with the application by regularly exploring new recipes, watching cooking videos, and interacting with the content.
- 3) Content Variety and Quality: Curate a diverse range of recipes across different categories such as breakfast, lunch, dinner, and vegetarian options to cater to various dietary preferences and occasions.
- 4) Personalization: Allow users to personalize their experience by adding recipes to their favorites list, enabling them to easily access their preferred recipes for future reference.
- 5) User Retention and Loyalty: Implement features such as the ability to add recipes as favorites to encourage users to return to the application regularly and build long-term loyalty.

Topics Covered in this project:

- Scroll Views (Recycler View, Nested Scroll View)
- Intents (Both Implicit and Explicit)
- Progress Bar
- Fragments
- ViewPager2
- Material3 Bottom Navigation Drawer
- Card View
- Speech to Text converter
- Floating Action Button
- Animations (fade in, & fade out)
- Firebase Authentication (Mobile)
- Firebase Fire Store
- Firebase cloud Storage
- Firebase Cloud Messaging
- Notification Manager
- Custom Alert Dialog
- Room Database
- Shared Preferences
- Permissions (Internet, Media, Contacts)

Screen Shots:

Mobile Authentication



OTP Verification


We will send an One Time Password on this mobile number

Enter Mobile Number

+91 - 9391704193

GET OTP

Verify OTP



OTP Verification

Enter the OTP sent to

0

0

0

0


0

0

Didn't receive OTP? **RESEND OTP**

VERIFY

Profile Page



Name

Enter your Name

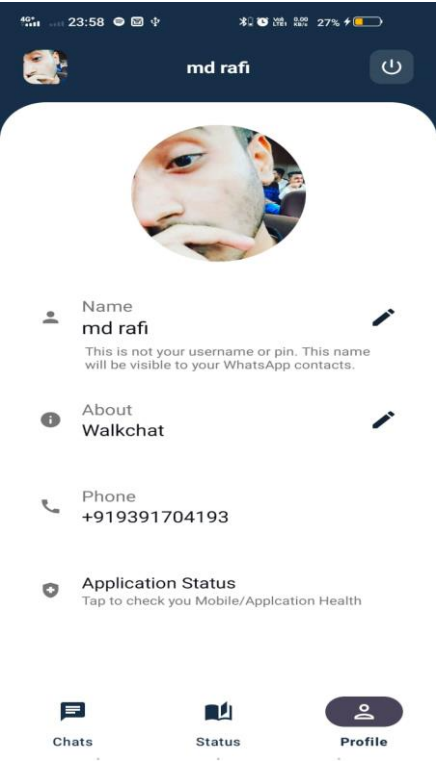
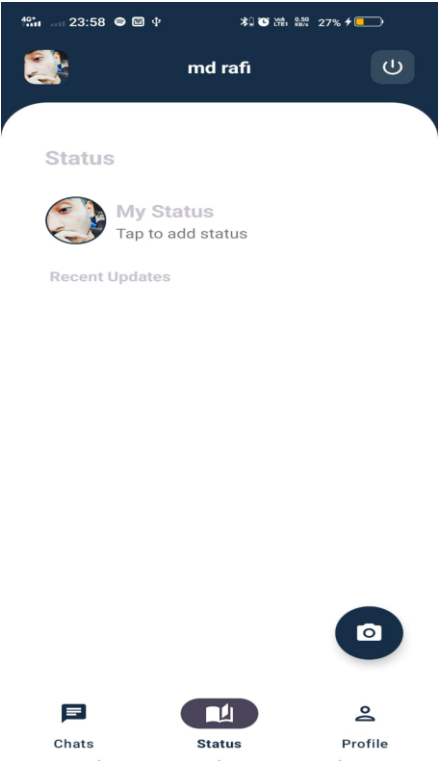
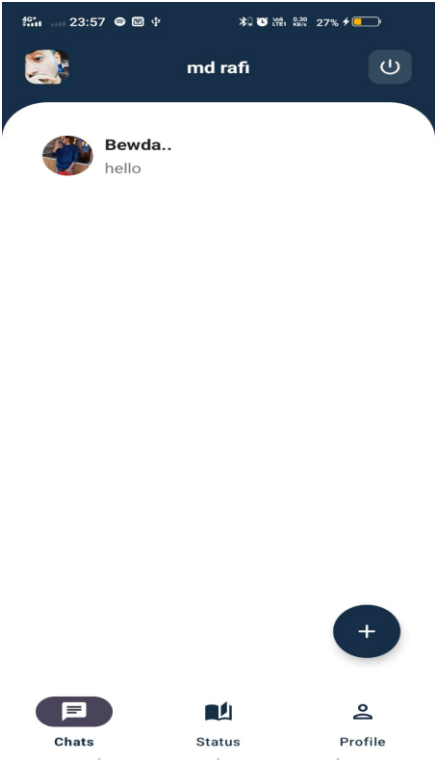
This is not your username or pin. This name will be visible to your WhatsApp contacts.

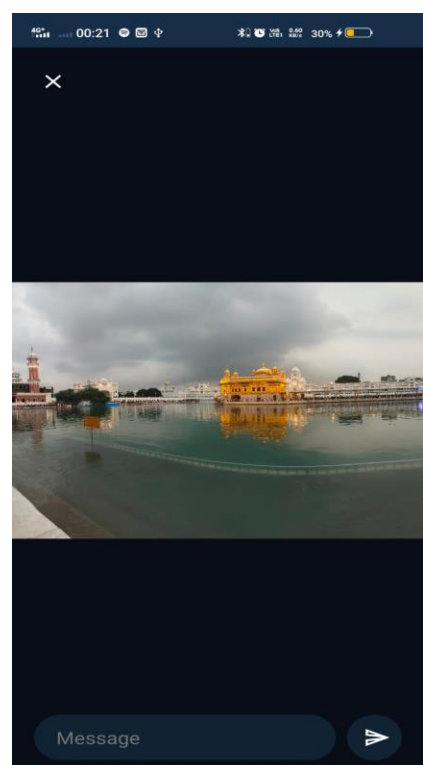
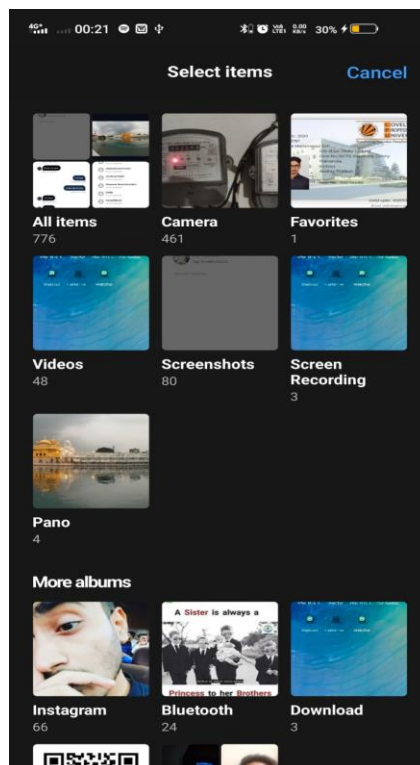
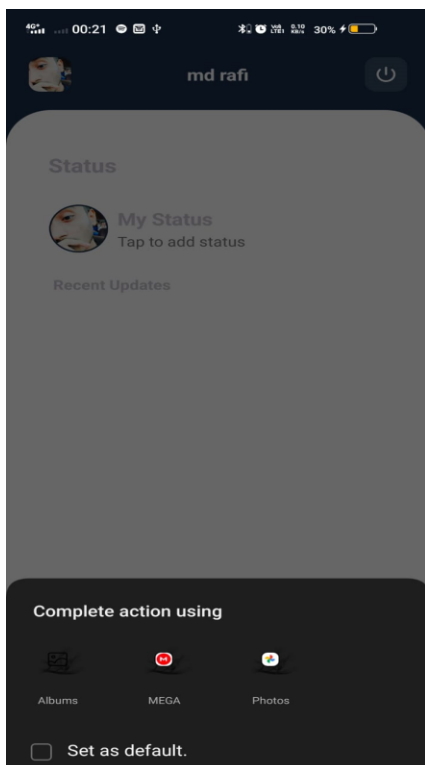
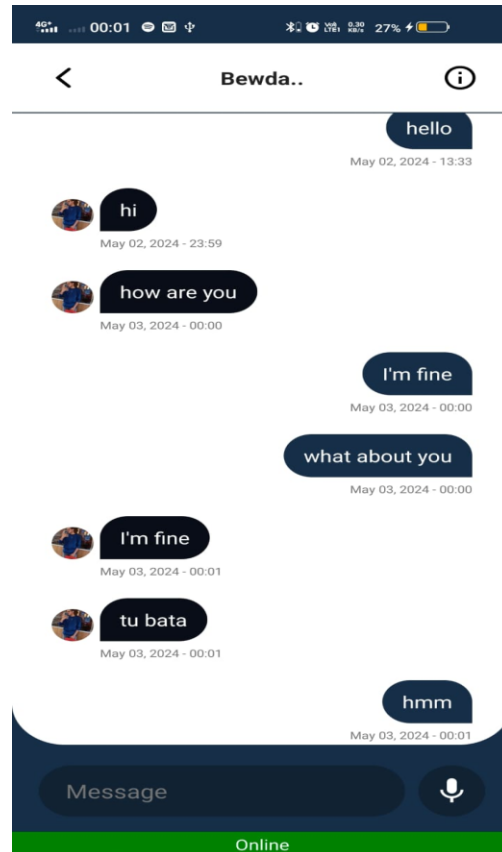
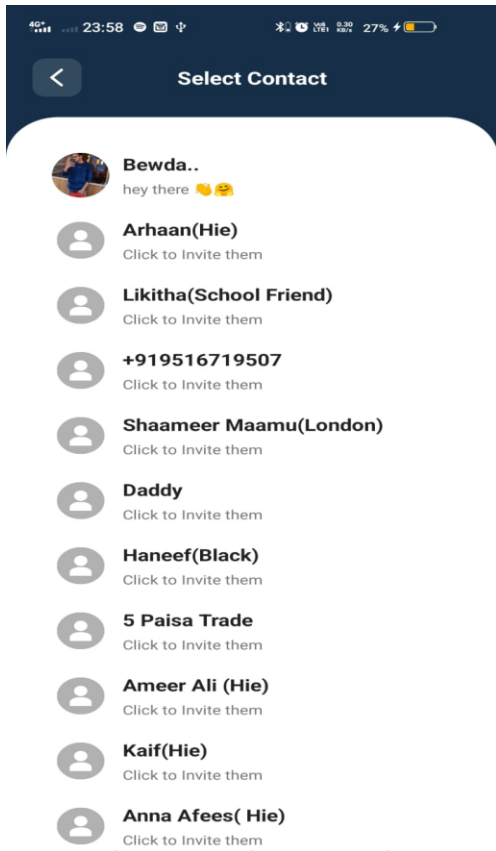
About

Your About

Phone

NEXT





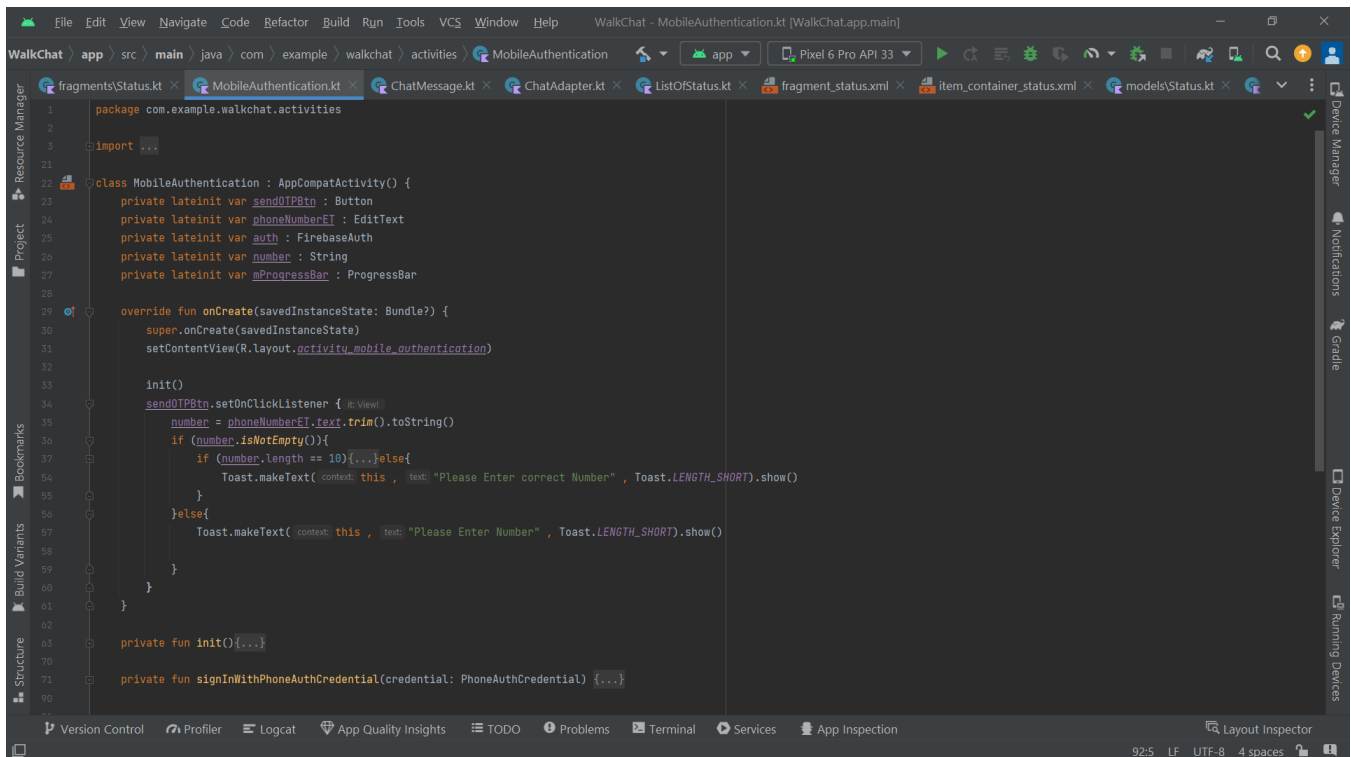
Functionalities:

Activity: 1 – Mobile Authentication

Key components include:

- Utilization of Firebase AUTH for user authentication.
- Integration of Phone Auth Options and Phone Auth Provider to send and verify OTPs.
- Handling of various states in the authentication process through callbacks, such as successful verification, failed verification, and OTP sent.

The activity features user feedback via toasts and UI updates using progress bars. The code ensures a smooth and secure user authentication process, enhancing the overall user experience.

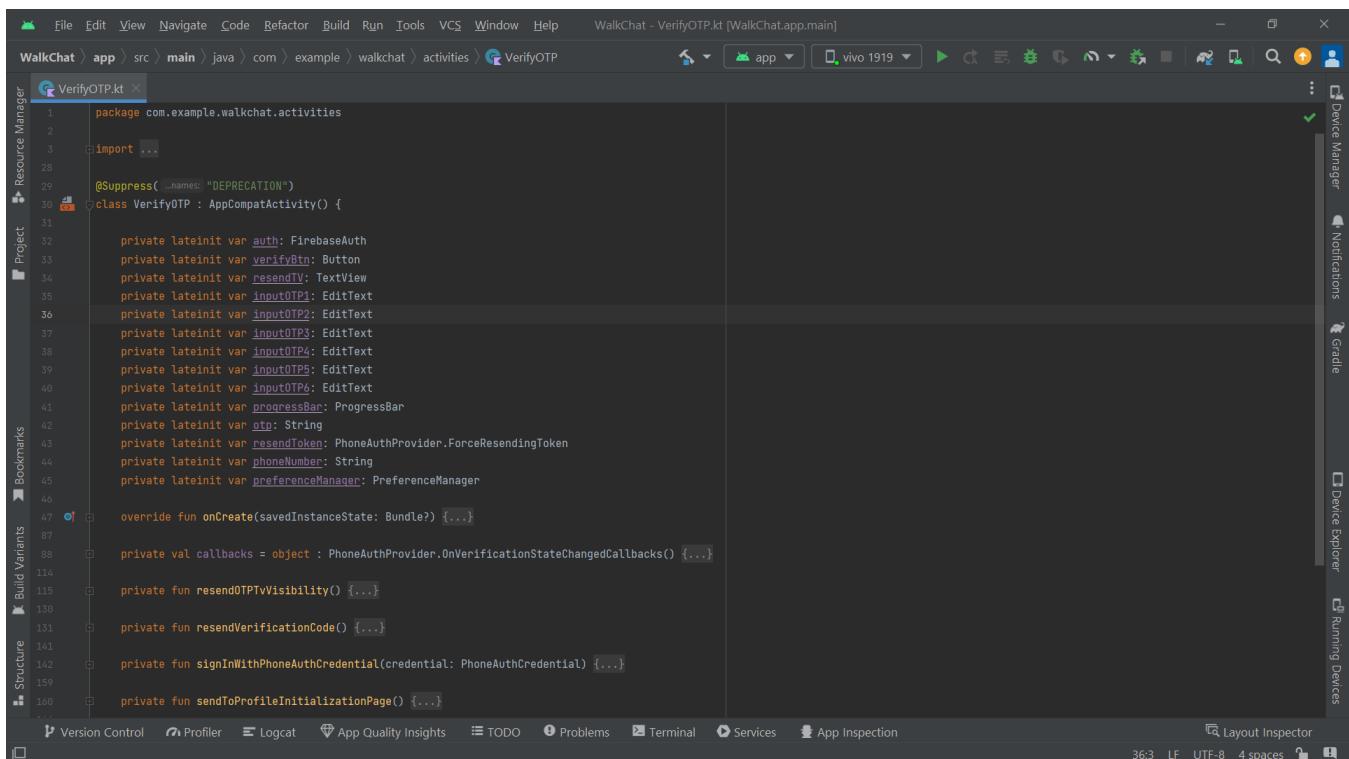


Activity: II – OTP Verification

Key features include:

- Integration of Firebase's authentication services, such as PhoneAuthProvider and FirebaseAuth, for OTP verification.
- Dynamic UI updates to enable/disable the resend OTP button after a cooldown period.
- Error handling for invalid OTP inputs and too many verification attempts.
- Seamless transition to the next activity upon successful verification, sending the user to the profile initialization page.

Additionally, the activity enhances user experience by organizing OTP input fields and providing smooth navigation between them using a custom EditTextWatcher class. Overall, the code demonstrates effective utilization of Firebase Phone Authentication for secure and efficient phone number verification in an Android application.



```
1 package com.example.walkchat.activities
2
3 import androidx.appcompat.app.AppCompatActivity
4
5 @Suppress("DEPRECATION")
6 class VerifyOTP : AppCompatActivity() {
7
8     private lateinit var auth: FirebaseAuth
9     private lateinit var verifyBtn: Button
10    private lateinit var resendTV: TextView
11    private lateinit var inputOTP1: EditText
12    private lateinit var inputOTP2: EditText
13    private lateinit var inputOTP3: EditText
14    private lateinit var inputOTP4: EditText
15    private lateinit var inputOTP5: EditText
16    private lateinit var inputOTP6: EditText
17    private lateinit var progressBar: ProgressBar
18    private lateinit var otp: String
19    private lateinit var resendToken: PhoneAuthProvider.ForceResendingToken
20    private lateinit var phoneNumber: String
21    private lateinit var preferenceManager: PreferenceManager
22
23    override fun onCreate(savedInstanceState: Bundle?) {
24        // ...
25    }
26
27    private val callbacks = object : PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
28        // ...
29    }
30
31    private fun resendOTPVisibility() {
32        // ...
33    }
34
35    private fun resendVerificationCode() {
36        // ...
37    }
38
39    private fun signInWithPhoneAuthCredential(credential: PhoneAuthCredential) {
40        // ...
41    }
42
43    private fun sendToProfileInitializationPage() {
44        // ...
45    }
46}
```

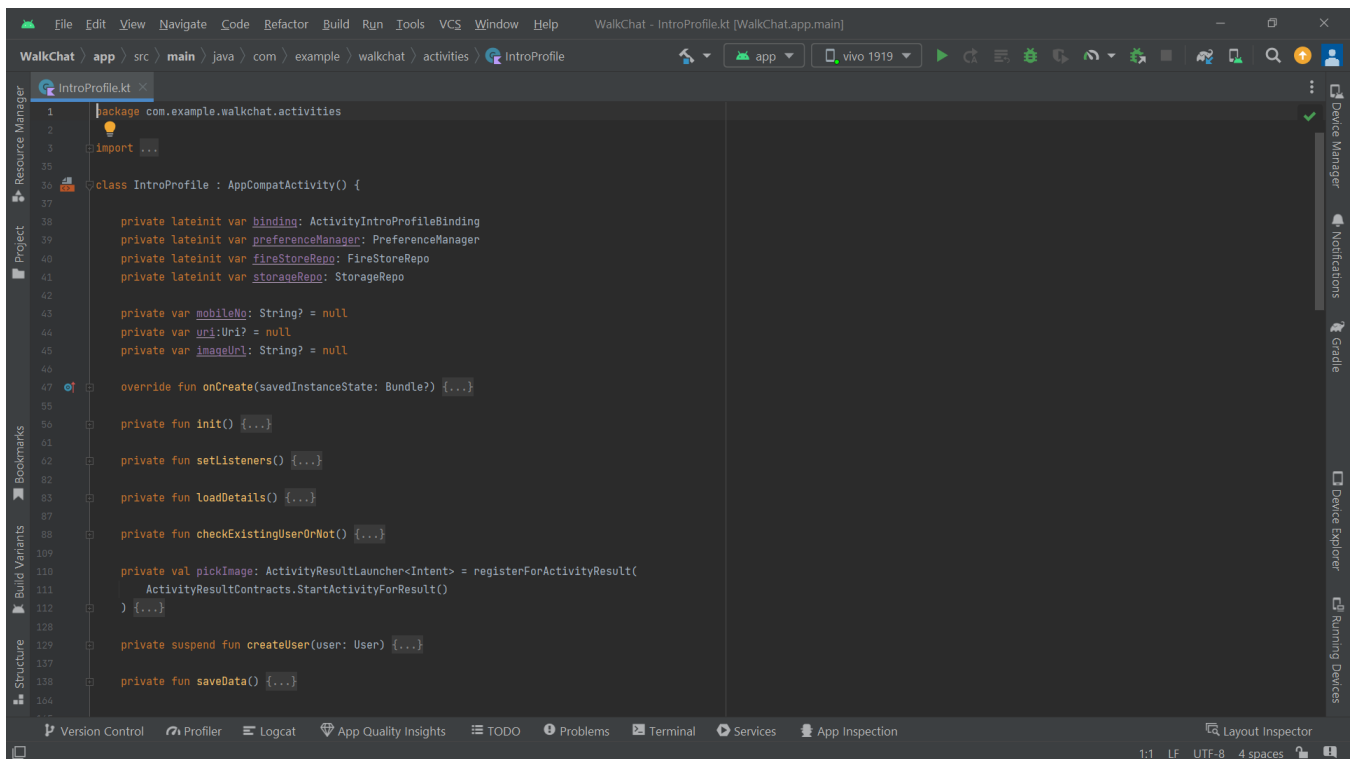
Activity: III – Profile Page

Key features include:

- Integration of Firebase services for user authentication and Firestore database.
- Use of coroutines for asynchronous tasks such as uploading images and updating user information.
- Implementation of activity result launcher for picking images from the device gallery.
- Custom dialog implementation for updating user information.
- Data validation to ensure that essential profile information such as name is provided before proceeding.

Management of UI state to show loading progress during data processing.

Overall, the activity provides a seamless user experience for initializing their profile within the application, ensuring essential user information is captured accurately and efficiently.

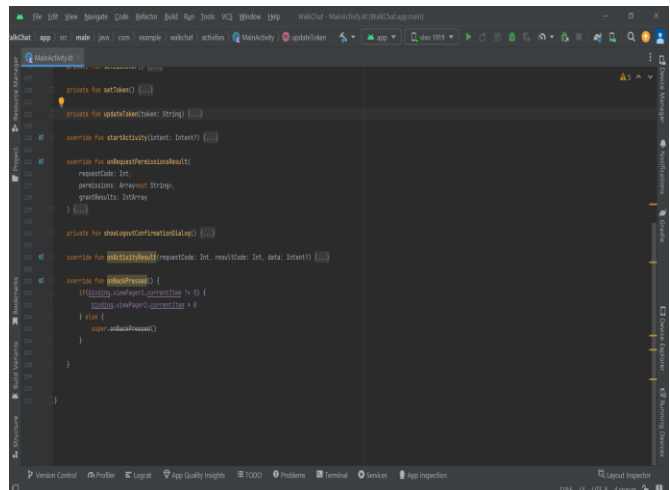
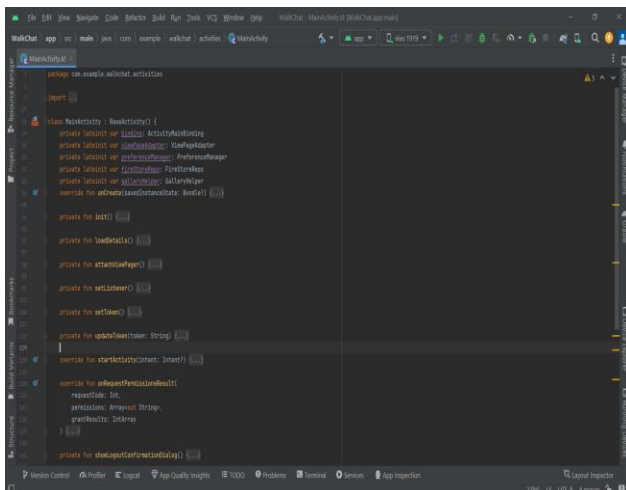


Activity: IV – Main Activity

Key features include:

- Integration of ViewPager2 to facilitate horizontal swiping between different sections.
- Bottom navigation setup for easy navigation between chat, status, and profile sections.
- Implementation of Firebase Messaging to update the FCM token and manage push notifications.
- Utilization of coroutines for asynchronous tasks such as updating the FCM token on Firestore.
- Custom logout confirmation dialog to ensure user confirmation before logging out.
- Integration of GalleryHelper to handle opening the device gallery for selecting images.
- Management of activity transitions to provide a smoother user experience.

Overall, the MainActivity enhances user engagement by providing a seamless interface for accessing various features within the application.

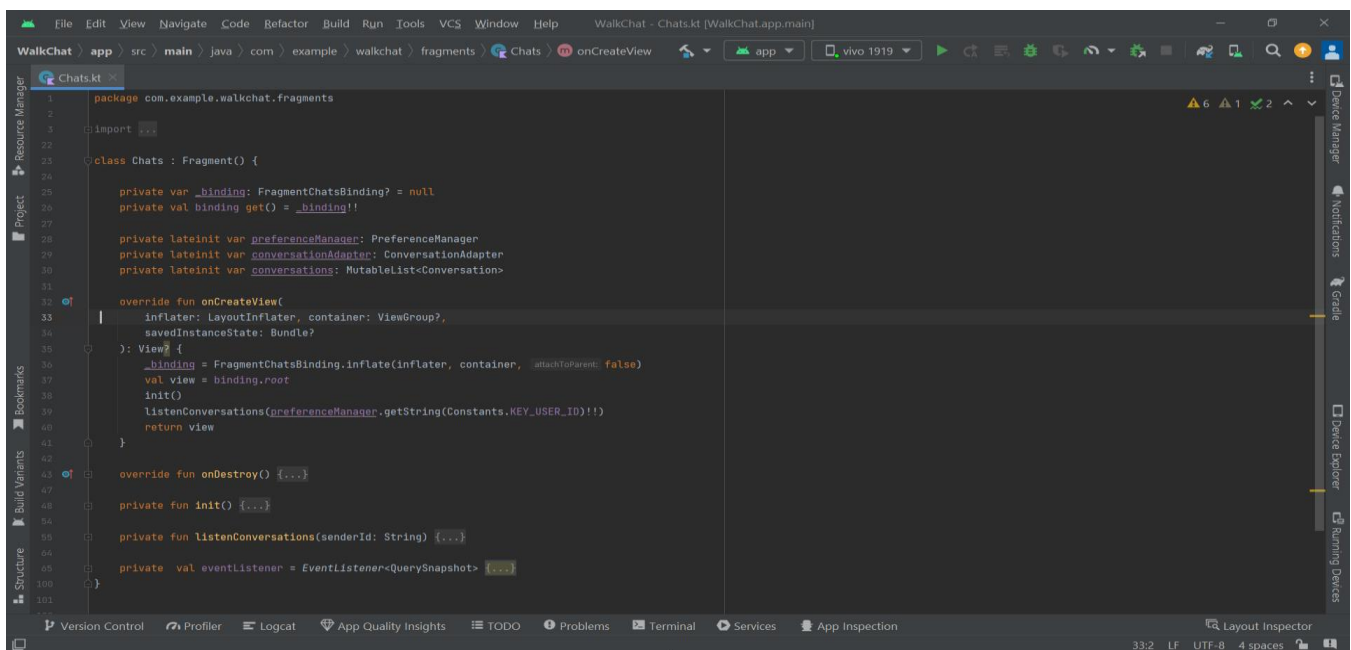


Fragment: I – Conversations

Key features include:

- Initialization of RecyclerView and ConversationAdapter for displaying conversations.
- Integration of Firestore to listen for real-time updates on conversations.
- Sorting conversations based on the timestamp of the most recent message.
- Handling document changes such as addition and modification of conversation details.
- Displaying progress bar during data loading and updating the UI accordingly.

Overall, the Chats fragment provides users with an up-to-date view of their conversations, ensuring a seamless communication experience within the application.

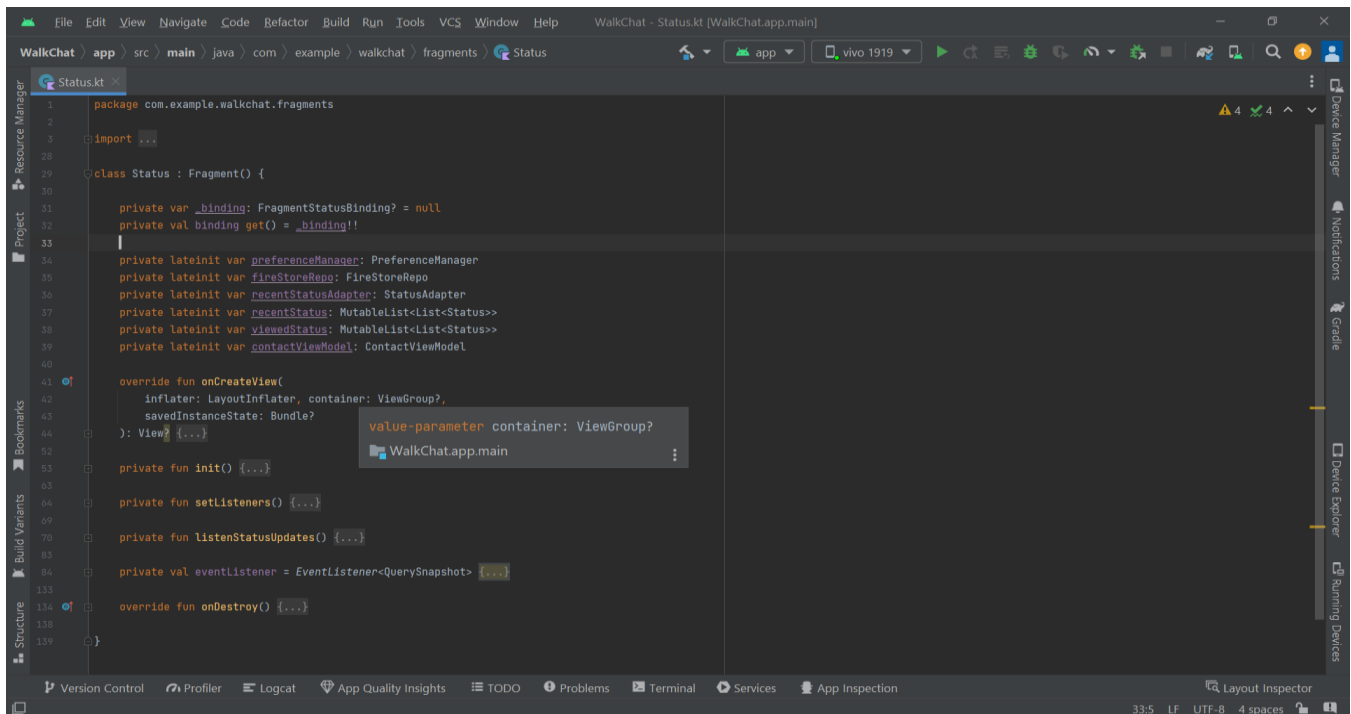


Fragment: II – Status

Key features include:

- Initialization of RecyclerView and StatusAdapter for displaying recent status updates.
- Integration of Firestore to listen for real-time updates on status changes.
- Sorting status updates based on the timestamp of the most recent update.
- Grouping status updates by user and sorting them within each group.
- Handling document changes such as addition of new status updates.

Overall, the Status fragment provides users with an up-to-date view of recent status updates from their contacts, ensuring they stay informed about their network's activities.

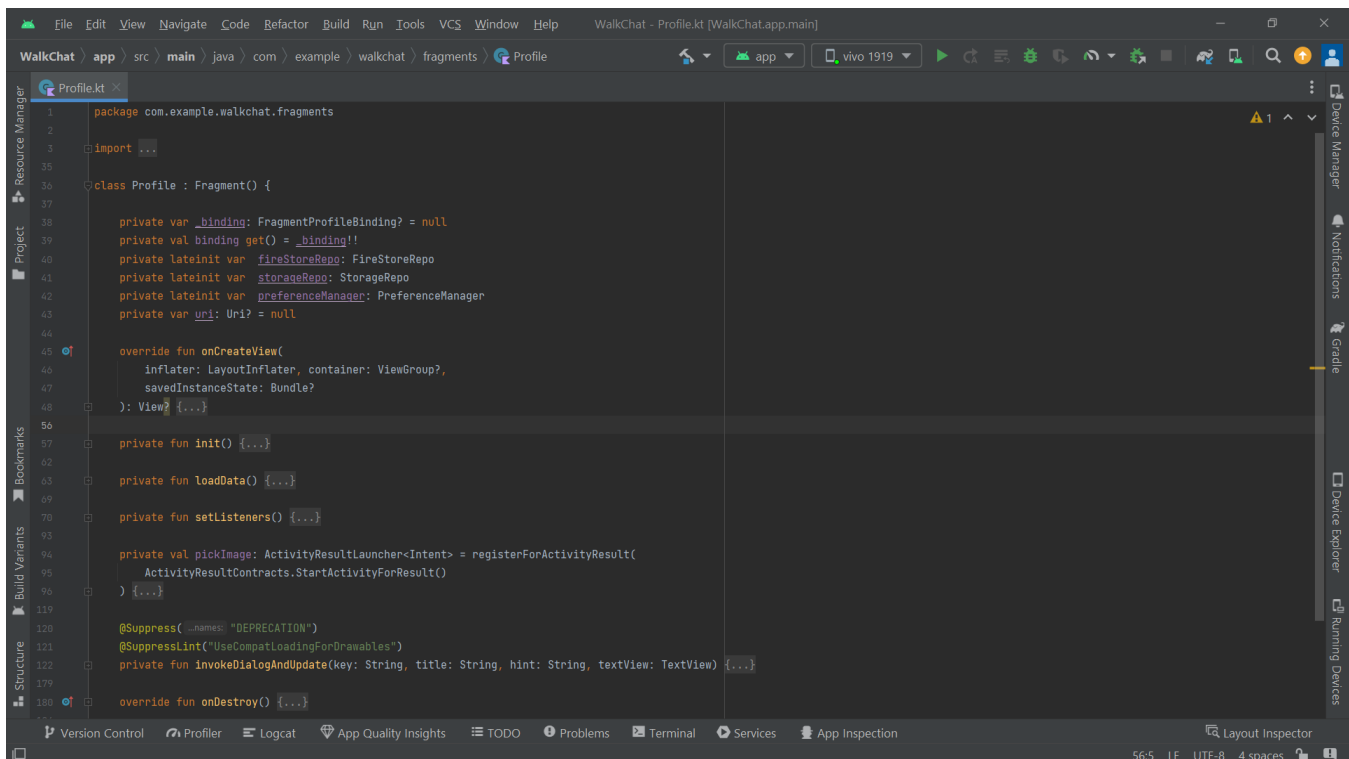


Fragment: III – Profile (User)

Key features include:

- Loading user profile data such as name, about, and mobile number from SharedPreferences.
- Providing functionality to update the profile picture by selecting an image from the device's gallery.
- Implementing dialogs for updating username and about information, with real-time updates to the UI and Firestore database.
- Utilizing Glide library for image loading and displaying default profile images if the user hasn't set one.
- Handling lifecycle events to prevent memory leaks.

Overall, the Profile fragment offers a user-friendly interface for managing profile information and updating it in real-time.

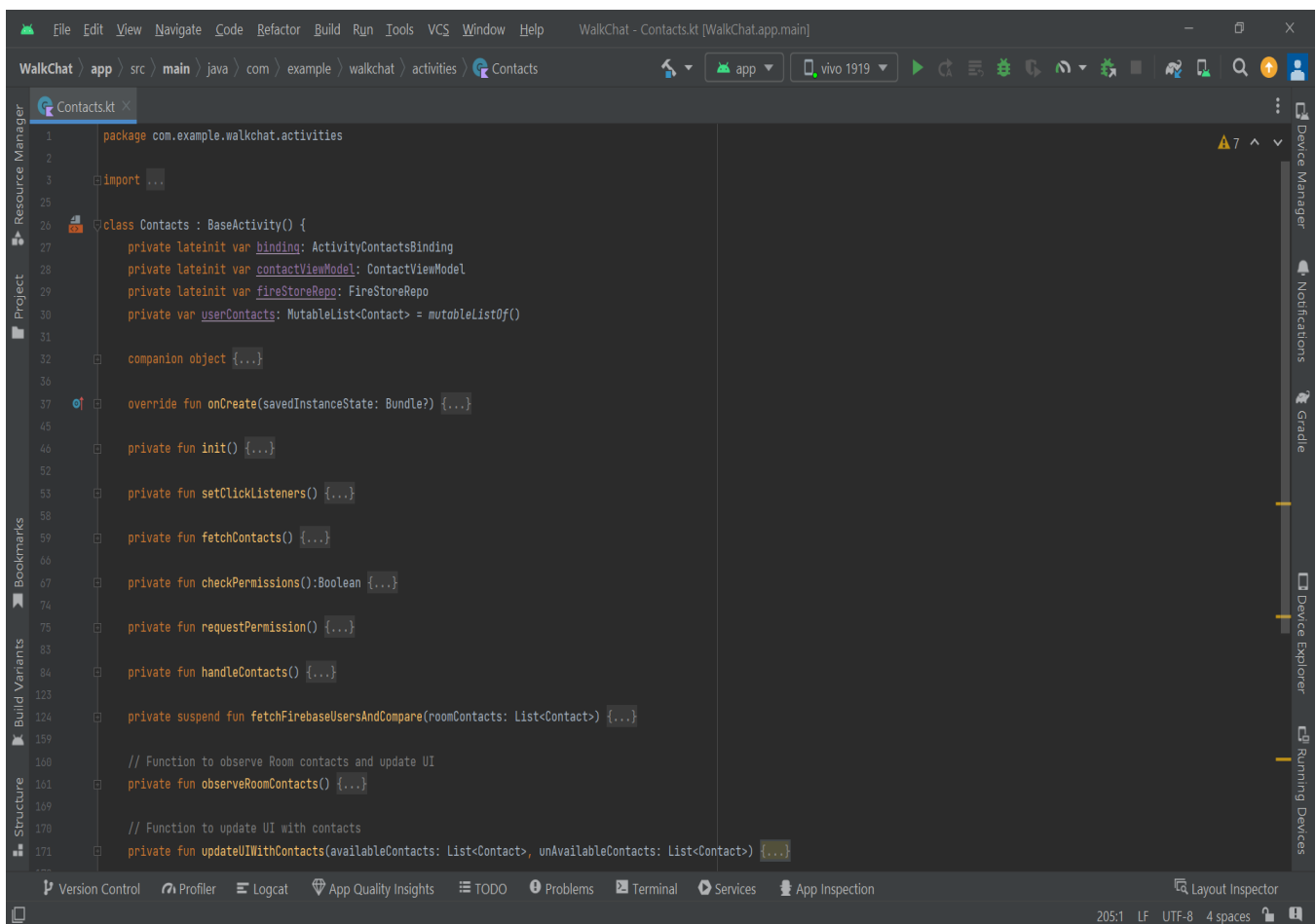


Activity: V – Contacts

Key features include:

- Requesting permission to access device contacts if not already granted.
- Handling the permission request result and proceeding with contact handling if granted.
- Fetching device contacts and comparing them with Firebase users to update their information in the Room database.
- Observing changes in Room database contacts and updating the UI accordingly.
- Displaying a progress bar while fetching and updating contacts to indicate loading.

Overall, the **Contacts** activity provides a seamless experience for users to interact with their contacts while efficiently managing data synchronization between local and remote sources.



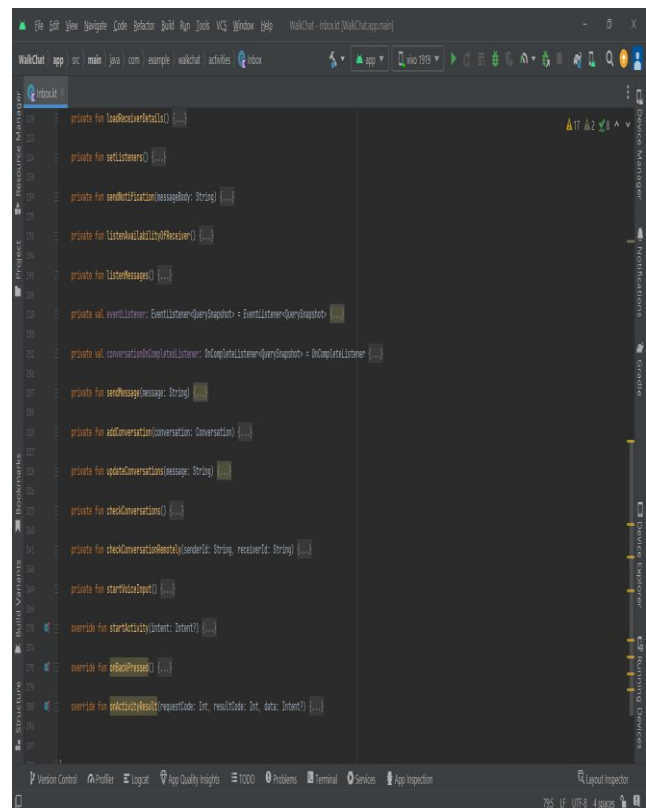
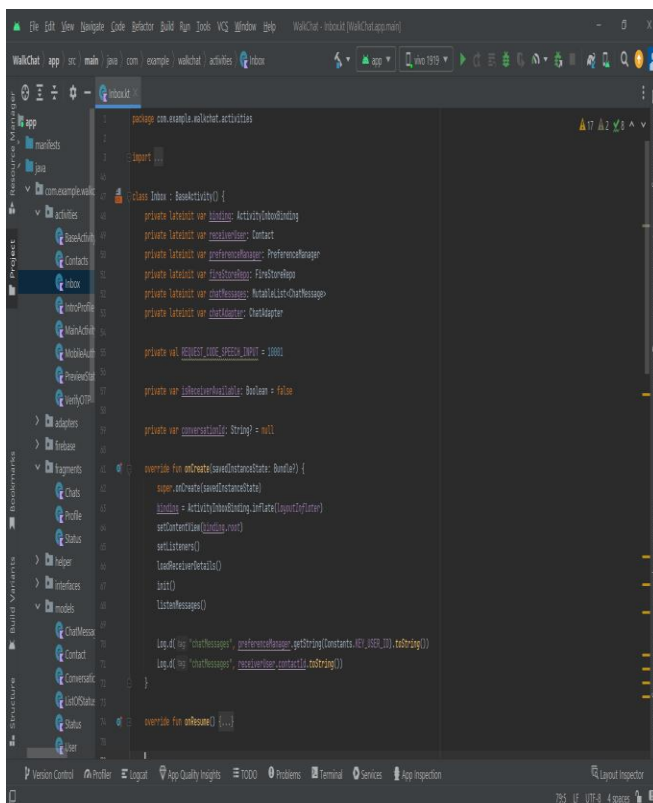
```
1 package com.example.walkchat.activities
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26 class Contacts : BaseActivity() {
27     private lateinit var binding: ActivityContactsBinding
28     private lateinit var contactViewModel: ContactViewModel
29     private lateinit var firestoreRepo: FirestoreRepo
30     private var userContacts: MutableList<Contact> = mutableListOf()
31
32     companion object { ... }
33
34
35
36
37     override fun onCreate(savedInstanceState: Bundle?) { ... }
38
39
40
41
42
43
44
45
46     private fun init() { ... }
47
48
49
50
51
52
53     private fun setClickListeners() { ... }
54
55
56
57
58
59     private fun fetchContacts() { ... }
60
61
62
63
64
65
66
67     private fun checkPermissions(): Boolean { ... }
68
69
70
71
72
73
74
75     private fun requestPermission() { ... }
76
77
78
79
80
81
82
83
84     private fun handleContacts() { ... }
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124     private suspend fun fetchFirebaseUsersAndCompare(roomContacts: List<Contact>) { ... }
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161 // Function to observe Room contacts and update UI
162 private fun observeRoomContacts() { ... }
163
164
165
166
167
168
169
170 // Function to update UI with contacts
171 private fun updateUIWithContacts(availableContacts: List<Contact>, unavailableContacts: List<Contact>) { ... }
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Activity: VI – Inbox

Key features include:

- Loading receiver details from the intent.
- Initializing necessary variables and adapters.
- Setting listeners for UI elements.
- Listening to messages using Firestore.
- Sending messages and updating conversations in Firestore.
- Handling speech input using speech recognition.
- Sending notifications to users when they are offline

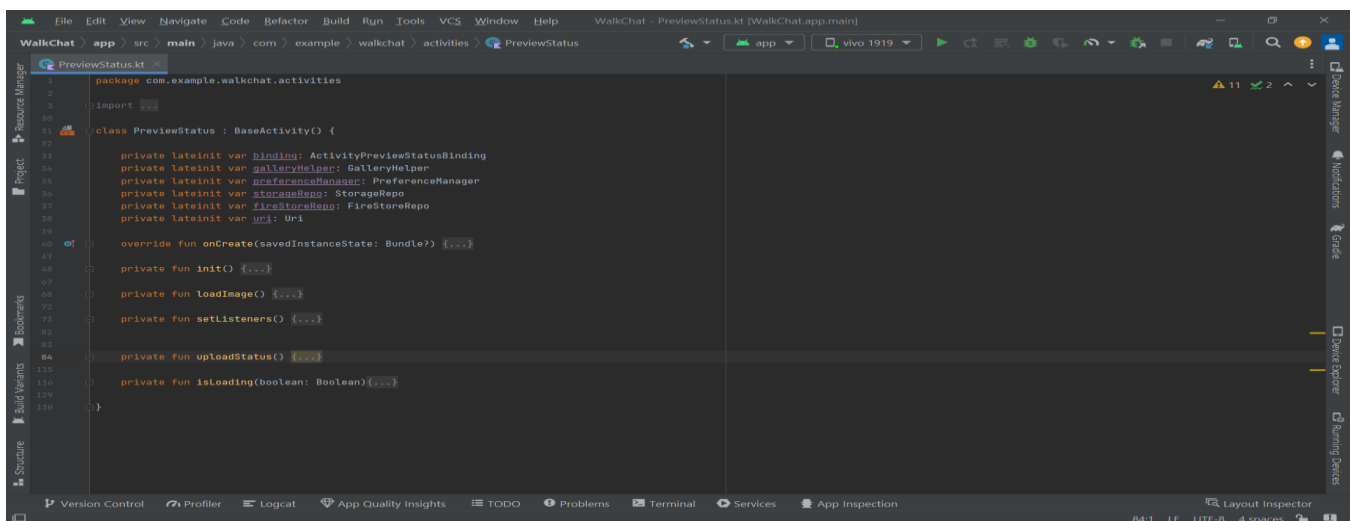
Overall, the Inbox activity provides a comprehensive interface for users to exchange messages and interact with each other within the application.



Activity: VII – Status

Key features Include:

- Initialization: It initializes necessary variables, such as galleryHelper, preferenceManager, storageRepo, and firestoreRepo. It also checks the intent to determine the mode of operation: either uploading a new status or viewing a preview.
- Load Image: If the mode is upload, it loads the image from the URI passed in the intent using Glide and displays it in the ImageView.
- Set Listeners: It sets click listeners for the send button and back button.
- Upload Status: When the send button is clicked, it initiates the upload process. It first shows a progress bar to indicate loading. Then, it uploads the status image to Firebase Storage and adds a corresponding entry to Firestore. After completion, it shows a toast message indicating success or failure and finishes the activity.
- isLoading: This function manages the visibility and enable/disable state of UI elements based on the loading state. When loading is true, it disables the send button, input message, and back button, and shows the progress bar.



References

- [1] Scroll View: <https://developer.android.com/reference/android/widget/ScrollView>
- [2] Rating Bar: <https://developer.android.com/reference/android/widget/RatingBar>
- [3] Intents: <https://developer.android.com/guide/components/intents-filters>
- [4] Progress Bar: <https://developer.android.com/reference/android/widget/ProgressBar>
- [5] Fragments: <https://developer.android.com/guide/fragments>
- [6] Camera Intent: <https://developer.android.com/training/camera/photobasics>
- [7] Splash Screen: <https://developer.android.com/guide/topics/ui/declaring-layout>
- [8] Firebase: <https://firebase.google.com/>
- [9] Creating Swipe Views with Tabs: <https://developer.android.com/training/animation/screen-slide>
- [10] Creating Bottom Navigation Drawer:
https://developer.android.com/guide/navigation/navigation-ui#bottom_nav
- [11] Card View: <https://developer.android.com/guide/topics/ui/layout/cardview>
- [12] Speech to Text Converter:
<https://developer.android.com/reference/android/speech/SpeechRecognizer>
- [13] Floating Action Button: <https://developer.android.com/guide/topics/ui/floating-action-button>

Bibliography

1. ANDROID APPLICATION DEVELOPMENT ALL-IN-ONE FOR DUMMIES by BARRY BURD, JOHN PAUL MUELLER, WILEY
2. ANDROID APPLICATION DEVELOPMENT by BARRY BURD, WILEY
3. ANDROID APPLICATION DEVELOPMENT by PRADEEP KOTHARI, DREAMTECH PRESS