

```

1 package com.marktrs.macapp.Fragment.Recruiter;
2
3
4 import android.os.Bundle;
5 import android.support.annotation.Nullable;
6 import android.support.v4.app.Fragment;
7 import android.support.v4.app.FragmentManager;
8 import android.support.v4.app.FragmentTransaction;
9 import android.view.LayoutInflater;
10 import android.view.View;
11 import android.view.ViewGroup;
12 import android.widget.Button;
13 import android.widget.EditText;
14 import android.widget.ImageButton;
15 import android.widget.Toast;
16
17 import com.google.firebase.auth.FirebaseAuth;
18 import com.google.firebase.auth.FirebaseUser;
19 import com.google.firebaseio.database.DatabaseReference;
20 import com.google.firebaseio.database.FirebaseDatabase;
21 import com.marktrs.macapp.Fragment.WorkerSetUpFragment;
22 import com.marktrs.macapp.Model.Job;
23 import com.marktrs.macapp.R;
24
25 /**
26  * A simple @link Fragment subclass.
27  */
28 public class AddNewJobFragment extends Fragment {
29
30
31     private Button confirmButton;
32     private EditText jobNameET;
33     private EditText symptomTypeET;
34     private EditText requiredEducationET;
35     private EditText requiredSkillET;
36     private EditText workplaceET;
37     private EditText paymentET;
38
39     private DatabaseReference mDatabase;
40     private FirebaseAuth mFirebaseAuth;
41     private FirebaseUser mFirebaseUser;
42
43     private ImageButton fab;
44
45     public AddNewJobFragment() {
46         // Required empty public constructor
47     }
48
49
50     @Override
51     public View onCreateView(LayoutInflater inflater, ViewGroup container,
52                             Bundle savedInstanceState) {
53         // Inflate the layout for this fragment
54         mDatabase = FirebaseDatabase.getInstance().getReference();
55         mFirebaseAuth = FirebaseAuth.getInstance();
56         mFirebaseUser = mFirebaseAuth.getCurrentUser();
57         return inflater.inflate(R.layout.fragment_add_new_job, container, false);
58     }
59
60     @Override
61     public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
62         super.onViewCreated(view, savedInstanceState);
63         jobNameET = (EditText) view.findViewById(R.id.job_name);
64         symptomTypeET = (EditText) view.findViewById(R.id.job_symptoms);
65         requiredEducationET = (EditText) view.findViewById(R.id.job_education);
66         requiredSkillET = (EditText) view.findViewById(R.id.job_skill);
67         workplaceET = (EditText) view.findViewById(R.id.job_workplace);
68         paymentET = (EditText) view.findViewById(R.id.job_payment);
69
70         confirmButton = (Button) view.findViewById(R.id.job_submit_button);
71         confirmButton.setOnClickListener(new View.OnClickListener() {
72             @Override
73             public void onClick(View view) {
74                 addJobToFirebase();
75                 PostedJobFragment postedJobFragment = new PostedJobFragment();
76                 FragmentManager manager = getFragmentManager();

```

```
77         FragmentTransaction transaction = manager.beginTransaction();
78         transaction.remove(AddNewJobFragment.this);
79         transaction.replace(R.id.fragment_area, postedJobFragment);
80         transaction.commit();
81         Toast.makeText(getApplicationContext(), "Successful !",
82                         Toast.LENGTH_SHORT).show();
83     }
84 }
85
86
87 public void addJobToFirebase() {
88     String key = mDatabase.child("Jobs").push().getKey();
89
90     Job job = new Job();
91     job.setJobId(key);
92     job.setJobName(jobNameET.getText().toString());
93     job.setSymptomType(symptomTypeET.getText().toString());
94     job.setRequiredEducation(requiredEducationET.getText().toString());
95     job.setRequiredSkill(requiredSkillET.getText().toString());
96     job.setWorkplace(workplaceET.getText().toString());
97     job.setPayment(paymentET.getText().toString());
98     job.setOwnerUID(mFirebaseUser.getUid());
99
100    mDatabase.child("Jobs").child(key).setValue(job);
101}
102
103 @Override
104 public void onDetach() {
105     super.onDetach();
106     fab = (ImageButton) getActivity().findViewById(R.id.fab);
107     fab.setVisibility(View.VISIBLE);
108 }
109
110 }
```

```
1 package com.marktrs.macapp.Fragment.Recruiter;
2
3 import android.content.Context;
4 import android.os.Bundle;
5 import android.support.v4.app.Fragment;
6 import android.support.v7.widget.GridLayoutManager;
7 import android.support.v7.widget.LinearLayoutManager;
8 import android.support.v7.widget.RecyclerView;
9 import android.util.Log;
10 import android.view.LayoutInflater;
11 import android.view.View;
12 import android.view.ViewGroup;
13 import android.widget.ImageButton;
14 import android.widget.TextView;
15
16 import com.google.firebase.auth.FirebaseAuth;
17 import com.google.firebase.auth.FirebaseUser;
18 import com.google.firebaseio.database.DataSnapshot;
19 import com.google.firebaseio.database.DatabaseError;
20 import com.google.firebaseio.database.DatabaseReference;
21 import com.google.firebaseio.database.FirebaseDatabase;
22 import com.google.firebaseio.database.ValueEventListener;
23 import com.marktrs.macapp.Model.Job;
24 import com.marktrs.macapp.Model.JobApplication;
25 import com.marktrs.macapp.R;
26
27 import java.util.ArrayList;
28 import java.util.HashMap;
29 import java.util.Map;
30
31 public class PostedJobFragment extends Fragment {
32
33     // TODO: Customize parameter argument names
34     private static final String ARG_COLUMN_COUNT = "column-count";
35     // TODO: Customize parameters
36     private int mColumnCount = 2;
37
38     private OnPostedFragmentInteractionListener mListener;
39
40     private FirebaseDatabase database;
41     private DatabaseReference jobsRef;
42     private DatabaseReference jobApplicationRef;
43
44     private ValueEventListener getAllJobListener;
45     private ValueEventListener getAllApplicationListener;
46
47     private ArrayList<Job> jobs;
48
49     private FirebaseAuth mFirebaseAuth;
50     private FirebaseUser mFirebaseUser;
51
52     private RecyclerView recyclerView;
53     private Map<String, Integer> jobApplicationCount;
54     private TextView noContentText;
55     private ImageButton fab;
56
57     /**
58      * Mandatory empty constructor for the fragment manager to instantiate the
59      * fragment (e.g. upon screen orientation changes).
60      */
61     public PostedJobFragment() {
62     }
63
64     // TODO: Customize parameter initialization
65     @SuppressWarnings("unused")
66     public static PostedJobFragment newInstance(int columnCount) {
67         PostedJobFragment fragment = new PostedJobFragment();
68         Bundle args = new Bundle();
69         args.putInt(ARG_COLUMN_COUNT, columnCount);
70         fragment.setArguments(args);
71         return fragment;
72     }
73
74     @Override
75     public void onCreate(Bundle savedInstanceState) {
76         super.onCreate(savedInstanceState);
```

```
77
78     database = FirebaseDatabase.getInstance();
79     mFirebaseAuth = FirebaseAuth.getInstance();
80     mFirebaseUser = mFirebaseAuth.getCurrentUser();
81
82     if (getArguments() != null) {
83         mColumnCount = getArguments().getInt(ARG_COLUMN_COUNT);
84     }
85 }
86
87 @Override
88 public View onCreateView(LayoutInflater inflater, ViewGroup container,
89                         Bundle savedInstanceState) {
90     View view = inflater.inflate(R.layout.fragment_postedjob_list, container, false);
91
92     // Set the adapter
93     if (view instanceof RecyclerView) {
94         Context context = view.getContext();
95         recyclerView = (RecyclerView) view;
96         if (mColumnCount <= 1) {
97             recyclerView.setLayoutManager(new LinearLayoutManager(context));
98         } else {
99             recyclerView.setLayoutManager(new GridLayoutManager(context, mColumnCount));
100        }
101    }
102    //TODO: Add posted job here
103
104    jobsRef = database.getReference("Jobs");
105    jobApplicationRef = database.getReference("JobApplications");
106
107    mFirebaseUser = mFirebaseAuth.getCurrentUser();
108
109    getAllApplicationListener = new ValueEventListener() {
110        @Override
111        public void onDataChange(DataSnapshot dataSnapshot) {
112            jobApplicationCount = new HashMap<>();
113            if (dataSnapshot.getChildrenCount() > 0) {
114                for (DataSnapshot dtSnapshot : dataSnapshot.getChildren()) {
115                    JobApplication application = dtSnapshot.getValue(JobApplication.class);
116                    String id = application.getJobId();
117                    if (jobApplicationCount.containsKey(id)) {
118                        jobApplicationCount.put(id, jobApplicationCount.get(id) + 1);
119                    } else {
120                        jobApplicationCount.put(id, 1);
121                    }
122                }
123                jobsRef.addListenerForSingleValueEvent(getAllJobListener);
124            }
125        }
126    }
127
128    @Override
129    public void onCancelled(DatabaseError databaseError) {
130    }
131    ;
132
133    getAllJobListener = new ValueEventListener() {
134        @Override
135        public void onDataChange(DataSnapshot dataSnapshot) {
136            jobs = new ArrayList<>();
137            for (DataSnapshot dtSnapshot : dataSnapshot.getChildren()) {
138                Job job = dtSnapshot.getValue(Job.class);
139                if (job.getOwnerUID().equals(mFirebaseUser.getUid())) {
140                    jobs.add(job);
141                }
142            }
143            if (jobs.isEmpty()) {
144                if (jobs.isEmpty()){
145                    noContentText = (TextView) getActivity().findViewById(R.id.no_content);
146                    noContentText.setText("You never announce any job \n\n Try to tap (+) button to add new
147 job");
148                    noContentText.setVisibility(View.VISIBLE);
149                }
150            } else {
151                recyclerView.setAdapter(new MyPostedJobRecyclerAdapter(jobs, jobApplicationCount,
152 mListener));
153            }
154        }
155    }
156
157    @Override
158    public void onStart() {
159        super.onStart();
160        getAllApplicationListener();
161    }
162
163    @Override
164    public void onStop() {
165        super.onStop();
166        getAllJobListener();
167    }
168
169    @Override
170    public void onPause() {
171        super.onPause();
172        getAllJobListener();
173    }
174
175    @Override
176    public void onResume() {
177        super.onResume();
178        getAllJobListener();
179    }
180
181    @Override
182    public void onDestroy() {
183        super.onDestroy();
184        getAllJobListener();
185    }
186
187    @Override
188    public void onRestart() {
189        super.onRestart();
190        getAllJobListener();
191    }
192
193    @Override
194    public void onBackPressed() {
195        super.onBackPressed();
196        Intent intent = new Intent(Intent.ACTION_MAIN);
197        intent.addCategory(Intent.CATEGORY_HOME);
198        startActivity(intent);
199    }
200
201    @Override
202    public void onConfigurationChanged(Configuration newConfig) {
203        super.onConfigurationChanged(newConfig);
204        recyclerView.setLayoutManager(new GridLayoutManager(this, mColumnCount));
205    }
206
207    @Override
208    public void onLowMemory() {
209        super.onLowMemory();
210        System.gc();
211    }
212
213    @Override
214    public void onTrimMemory(int level) {
215        super.onTrimMemory(level);
216        System.gc();
217    }
218
219    @Override
220    public void onTrimMemory(int level) {
221        super.onTrimMemory(level);
222        System.gc();
223    }
224
225    @Override
226    public void onTrimMemory(int level) {
227        super.onTrimMemory(level);
228        System.gc();
229    }
230
231    @Override
232    public void onTrimMemory(int level) {
233        super.onTrimMemory(level);
234        System.gc();
235    }
236
237    @Override
238    public void onTrimMemory(int level) {
239        super.onTrimMemory(level);
240        System.gc();
241    }
242
243    @Override
244    public void onTrimMemory(int level) {
245        super.onTrimMemory(level);
246        System.gc();
247    }
248
249    @Override
250    public void onTrimMemory(int level) {
251        super.onTrimMemory(level);
252        System.gc();
253    }
254
255    @Override
256    public void onTrimMemory(int level) {
257        super.onTrimMemory(level);
258        System.gc();
259    }
260
261    @Override
262    public void onTrimMemory(int level) {
263        super.onTrimMemory(level);
264        System.gc();
265    }
266
267    @Override
268    public void onTrimMemory(int level) {
269        super.onTrimMemory(level);
270        System.gc();
271    }
272
273    @Override
274    public void onTrimMemory(int level) {
275        super.onTrimMemory(level);
276        System.gc();
277    }
278
279    @Override
280    public void onTrimMemory(int level) {
281        super.onTrimMemory(level);
282        System.gc();
283    }
284
285    @Override
286    public void onTrimMemory(int level) {
287        super.onTrimMemory(level);
288        System.gc();
289    }
290
291    @Override
292    public void onTrimMemory(int level) {
293        super.onTrimMemory(level);
294        System.gc();
295    }
296
297    @Override
298    public void onTrimMemory(int level) {
299        super.onTrimMemory(level);
300        System.gc();
301    }
302
303    @Override
304    public void onTrimMemory(int level) {
305        super.onTrimMemory(level);
306        System.gc();
307    }
308
309    @Override
310    public void onTrimMemory(int level) {
311        super.onTrimMemory(level);
312        System.gc();
313    }
314
315    @Override
316    public void onTrimMemory(int level) {
317        super.onTrimMemory(level);
318        System.gc();
319    }
320
321    @Override
322    public void onTrimMemory(int level) {
323        super.onTrimMemory(level);
324        System.gc();
325    }
326
327    @Override
328    public void onTrimMemory(int level) {
329        super.onTrimMemory(level);
330        System.gc();
331    }
332
333    @Override
334    public void onTrimMemory(int level) {
335        super.onTrimMemory(level);
336        System.gc();
337    }
338
339    @Override
340    public void onTrimMemory(int level) {
341        super.onTrimMemory(level);
342        System.gc();
343    }
344
345    @Override
346    public void onTrimMemory(int level) {
347        super.onTrimMemory(level);
348        System.gc();
349    }
350
351    @Override
352    public void onTrimMemory(int level) {
353        super.onTrimMemory(level);
354        System.gc();
355    }
356
357    @Override
358    public void onTrimMemory(int level) {
359        super.onTrimMemory(level);
360        System.gc();
361    }
362
363    @Override
364    public void onTrimMemory(int level) {
365        super.onTrimMemory(level);
366        System.gc();
367    }
368
369    @Override
370    public void onTrimMemory(int level) {
371        super.onTrimMemory(level);
372        System.gc();
373    }
374
375    @Override
376    public void onTrimMemory(int level) {
377        super.onTrimMemory(level);
378        System.gc();
379    }
380
381    @Override
382    public void onTrimMemory(int level) {
383        super.onTrimMemory(level);
384        System.gc();
385    }
386
387    @Override
388    public void onTrimMemory(int level) {
389        super.onTrimMemory(level);
390        System.gc();
391    }
392
393    @Override
394    public void onTrimMemory(int level) {
395        super.onTrimMemory(level);
396        System.gc();
397    }
398
399    @Override
400    public void onTrimMemory(int level) {
401        super.onTrimMemory(level);
402        System.gc();
403    }
404
405    @Override
406    public void onTrimMemory(int level) {
407        super.onTrimMemory(level);
408        System.gc();
409    }
410
411    @Override
412    public void onTrimMemory(int level) {
413        super.onTrimMemory(level);
414        System.gc();
415    }
416
417    @Override
418    public void onTrimMemory(int level) {
419        super.onTrimMemory(level);
420        System.gc();
421    }
422
423    @Override
424    public void onTrimMemory(int level) {
425        super.onTrimMemory(level);
426        System.gc();
427    }
428
429    @Override
430    public void onTrimMemory(int level) {
431        super.onTrimMemory(level);
432        System.gc();
433    }
434
435    @Override
436    public void onTrimMemory(int level) {
437        super.onTrimMemory(level);
438        System.gc();
439    }
440
441    @Override
442    public void onTrimMemory(int level) {
443        super.onTrimMemory(level);
444        System.gc();
445    }
446
447    @Override
448    public void onTrimMemory(int level) {
449        super.onTrimMemory(level);
450        System.gc();
451    }
452
453    @Override
454    public void onTrimMemory(int level) {
455        super.onTrimMemory(level);
456        System.gc();
457    }
458
459    @Override
460    public void onTrimMemory(int level) {
461        super.onTrimMemory(level);
462        System.gc();
463    }
464
465    @Override
466    public void onTrimMemory(int level) {
467        super.onTrimMemory(level);
468        System.gc();
469    }
470
471    @Override
472    public void onTrimMemory(int level) {
473        super.onTrimMemory(level);
474        System.gc();
475    }
476
477    @Override
478    public void onTrimMemory(int level) {
479        super.onTrimMemory(level);
480        System.gc();
481    }
482
483    @Override
484    public void onTrimMemory(int level) {
485        super.onTrimMemory(level);
486        System.gc();
487    }
488
489    @Override
490    public void onTrimMemory(int level) {
491        super.onTrimMemory(level);
492        System.gc();
493    }
494
495    @Override
496    public void onTrimMemory(int level) {
497        super.onTrimMemory(level);
498        System.gc();
499    }
500
501    @Override
502    public void onTrimMemory(int level) {
503        super.onTrimMemory(level);
504        System.gc();
505    }
506
507    @Override
508    public void onTrimMemory(int level) {
509        super.onTrimMemory(level);
510        System.gc();
511    }
512
513    @Override
514    public void onTrimMemory(int level) {
515        super.onTrimMemory(level);
516        System.gc();
517    }
518
519    @Override
520    public void onTrimMemory(int level) {
521        super.onTrimMemory(level);
522        System.gc();
523    }
524
525    @Override
526    public void onTrimMemory(int level) {
527        super.onTrimMemory(level);
528        System.gc();
529    }
530
531    @Override
532    public void onTrimMemory(int level) {
533        super.onTrimMemory(level);
534        System.gc();
535    }
536
537    @Override
538    public void onTrimMemory(int level) {
539        super.onTrimMemory(level);
540        System.gc();
541    }
542
543    @Override
544    public void onTrimMemory(int level) {
545        super.onTrimMemory(level);
546        System.gc();
547    }
548
549    @Override
550    public void onTrimMemory(int level) {
551        super.onTrimMemory(level);
552        System.gc();
553    }
554
555    @Override
556    public void onTrimMemory(int level) {
557        super.onTrimMemory(level);
558        System.gc();
559    }
560
561    @Override
562    public void onTrimMemory(int level) {
563        super.onTrimMemory(level);
564        System.gc();
565    }
566
567    @Override
568    public void onTrimMemory(int level) {
569        super.onTrimMemory(level);
570        System.gc();
571    }
572
573    @Override
574    public void onTrimMemory(int level) {
575        super.onTrimMemory(level);
576        System.gc();
577    }
578
579    @Override
580    public void onTrimMemory(int level) {
581        super.onTrimMemory(level);
582        System.gc();
583    }
584
585    @Override
586    public void onTrimMemory(int level) {
587        super.onTrimMemory(level);
588        System.gc();
589    }
590
591    @Override
592    public void onTrimMemory(int level) {
593        super.onTrimMemory(level);
594        System.gc();
595    }
596
597    @Override
598    public void onTrimMemory(int level) {
599        super.onTrimMemory(level);
600        System.gc();
601    }
602
603    @Override
604    public void onTrimMemory(int level) {
605        super.onTrimMemory(level);
606        System.gc();
607    }
608
609    @Override
610    public void onTrimMemory(int level) {
611        super.onTrimMemory(level);
612        System.gc();
613    }
614
615    @Override
616    public void onTrimMemory(int level) {
617        super.onTrimMemory(level);
618        System.gc();
619    }
620
621    @Override
622    public void onTrimMemory(int level) {
623        super.onTrimMemory(level);
624        System.gc();
625    }
626
627    @Override
628    public void onTrimMemory(int level) {
629        super.onTrimMemory(level);
630        System.gc();
631    }
632
633    @Override
634    public void onTrimMemory(int level) {
635        super.onTrimMemory(level);
636        System.gc();
637    }
638
639    @Override
640    public void onTrimMemory(int level) {
641        super.onTrimMemory(level);
642        System.gc();
643    }
644
645    @Override
646    public void onTrimMemory(int level) {
647        super.onTrimMemory(level);
648        System.gc();
649    }
650
651    @Override
652    public void onTrimMemory(int level) {
653        super.onTrimMemory(level);
654        System.gc();
655    }
656
657    @Override
658    public void onTrimMemory(int level) {
659        super.onTrimMemory(level);
660        System.gc();
661    }
662
663    @Override
664    public void onTrimMemory(int level) {
665        super.onTrimMemory(level);
666        System.gc();
667    }
668
669    @Override
670    public void onTrimMemory(int level) {
671        super.onTrimMemory(level);
672        System.gc();
673    }
674
675    @Override
676    public void onTrimMemory(int level) {
677        super.onTrimMemory(level);
678        System.gc();
679    }
680
681    @Override
682    public void onTrimMemory(int level) {
683        super.onTrimMemory(level);
684        System.gc();
685    }
686
687    @Override
688    public void onTrimMemory(int level) {
689        super.onTrimMemory(level);
690        System.gc();
691    }
692
693    @Override
694    public void onTrimMemory(int level) {
695        super.onTrimMemory(level);
696        System.gc();
697    }
698
699    @Override
700    public void onTrimMemory(int level) {
701        super.onTrimMemory(level);
702        System.gc();
703    }
704
705    @Override
706    public void onTrimMemory(int level) {
707        super.onTrimMemory(level);
708        System.gc();
709    }
710
711    @Override
712    public void onTrimMemory(int level) {
713        super.onTrimMemory(level);
714        System.gc();
715    }
716
717    @Override
718    public void onTrimMemory(int level) {
719        super.onTrimMemory(level);
720        System.gc();
721    }
722
723    @Override
724    public void onTrimMemory(int level) {
725        super.onTrimMemory(level);
726        System.gc();
727    }
728
729    @Override
730    public void onTrimMemory(int level) {
731        super.onTrimMemory(level);
732        System.gc();
733    }
734
735    @Override
736    public void onTrimMemory(int level) {
737        super.onTrimMemory(level);
738        System.gc();
739    }
740
741    @Override
742    public void onTrimMemory(int level) {
743        super.onTrimMemory(level);
744        System.gc();
745    }
746
747    @Override
748    public void onTrimMemory(int level) {
749        super.onTrimMemory(level);
750        System.gc();
751    }
752
753    @Override
754    public void onTrimMemory(int level) {
755        super.onTrimMemory(level);
756        System.gc();
757    }
758
759    @Override
760    public void onTrimMemory(int level) {
761        super.onTrimMemory(level);
762        System.gc();
763    }
764
765    @Override
766    public void onTrimMemory(int level) {
767        super.onTrimMemory(level);
768        System.gc();
769    }
770
771    @Override
772    public void onTrimMemory(int level) {
773        super.onTrimMemory(level);
774        System.gc();
775    }
776
777    @Override
778    public void onTrimMemory(int level) {
779        super.onTrimMemory(level);
780        System.gc();
781    }
782
783    @Override
784    public void onTrimMemory(int level) {
785        super.onTrimMemory(level);
786        System.gc();
787    }
788
789    @Override
790    public void onTrimMemory(int level) {
791        super.onTrimMemory(level);
792        System.gc();
793    }
794
795    @Override
796    public void onTrimMemory(int level) {
797        super.onTrimMemory(level);
798        System.gc();
799    }
800
801    @Override
802    public void onTrimMemory(int level) {
803        super.onTrimMemory(level);
804        System.gc();
805    }
806
807    @Override
808    public void onTrimMemory(int level) {
809        super.onTrimMemory(level);
810        System.gc();
811    }
812
813    @Override
814    public void onTrimMemory(int level) {
815        super.onTrimMemory(level);
816        System.gc();
817    }
818
819    @Override
820    public void onTrimMemory(int level) {
821        super.onTrimMemory(level);
822        System.gc();
823    }
824
825    @Override
826    public void onTrimMemory(int level) {
827        super.onTrimMemory(level);
828        System.gc();
829    }
830
831    @Override
832    public void onTrimMemory(int level) {
833        super.onTrimMemory(level);
834        System.gc();
835    }
836
837    @Override
838    public void onTrimMemory(int level) {
839        super.onTrimMemory(level);
840        System.gc();
841    }
842
843    @Override
844    public void onTrimMemory(int level) {
845        super.onTrimMemory(level);
846        System.gc();
847    }
848
849    @Override
850    public void onTrimMemory(int level) {
851        super.onTrimMemory(level);
852        System.gc();
853    }
854
855    @Override
856    public void onTrimMemory(int level) {
857        super.onTrimMemory(level);
858        System.gc();
859    }
860
861    @Override
862    public void onTrimMemory(int level) {
863        super.onTrimMemory(level);
864        System.gc();
865    }
866
867    @Override
868    public void onTrimMemory(int level) {
869        super.onTrimMemory(level);
870        System.gc();
871    }
872
873    @Override
874    public void onTrimMemory(int level) {
875        super.onTrimMemory(level);
876        System.gc();
877    }
878
879    @Override
880    public void onTrimMemory(int level) {
881        super.onTrimMemory(level);
882        System.gc();
883    }
884
885    @Override
886    public void onTrimMemory(int level) {
887        super.onTrimMemory(level);
888        System.gc();
889    }
890
891    @Override
892    public void onTrimMemory(int level) {
893        super.onTrimMemory(level);
894        System.gc();
895    }
896
897    @Override
898    public void onTrimMemory(int level) {
899        super.onTrimMemory(level);
900        System.gc();
901    }
902
903    @Override
904    public void onTrimMemory(int level) {
905        super.onTrimMemory(level);
906        System.gc();
907    }
908
909    @Override
910    public void onTrimMemory(int level) {
911        super.onTrimMemory(level);
912        System.gc();
913    }
914
915    @Override
916    public void onTrimMemory(int level) {
917        super.onTrimMemory(level);
918        System.gc();
919    }
920
921    @Override
922    public void onTrimMemory(int level) {
923        super.onTrimMemory(level);
924        System.gc();
925    }
926
927    @Override
928    public void onTrimMemory(int level) {
929        super.onTrimMemory(level);
930        System.gc();
931    }
932
933    @Override
934    public void onTrimMemory(int level) {
935        super.onTrimMemory(level);
936        System.gc();
937    }
938
939    @Override
940    public void onTrimMemory(int level) {
941        super.onTrimMemory(level);
942        System.gc();
943    }
944
945    @Override
946    public void onTrimMemory(int level) {
947        super.onTrimMemory(level);
948        System.gc();
949    }
950
951    @Override
952    public void onTrimMemory(int level) {
953        super.onTrimMemory(level);
954        System.gc();
955    }
956
957    @Override
958    public void onTrimMemory(int level) {
959        super.onTrimMemory(level);
960        System.gc();
961    }
962
963    @Override
964    public void onTrimMemory(int level) {
965        super.onTrimMemory(level);
966        System.gc();
967    }
968
969    @Override
970    public void onTrimMemory(int level) {
971        super.onTrimMemory(level);
972        System.gc();
973    }
974
975    @Override
976    public void onTrimMemory(int level) {
977        super.onTrimMemory(level);
978        System.gc();
979    }
980
981    @Override
982    public void onTrimMemory(int level) {
983        super.onTrimMemory(level);
984        System.gc();
985    }
986
987    @Override
988    public void onTrimMemory(int level) {
989        super.onTrimMemory(level);
990        System.gc();
991    }
992
993    @Override
994    public void onTrimMemory(int level) {
995        super.onTrimMemory(level);
996        System.gc();
997    }
998
999    @Override
1000   public void onTrimMemory(int level) {
1001      super.onTrimMemory(level);
1002      System.gc();
1003  }
1004
1005  @Override
1006  public void onTrimMemory(int level) {
1007      super.onTrimMemory(level);
1008      System.gc();
1009  }
1010
1011  @Override
1012  public void onTrimMemory(int level) {
1013      super.onTrimMemory(level);
1014      System.gc();
1015  }
1016
1017  @Override
1018  public void onTrimMemory(int level) {
1019      super.onTrimMemory(level);
1020      System.gc();
1021  }
1022
1023  @Override
1024  public void onTrimMemory(int level) {
1025      super.onTrimMemory(level);
1026      System.gc();
1027  }
1028
1029  @Override
1030  public void onTrimMemory(int level) {
1031      super.onTrimMemory(level);
1032      System.gc();
1033  }
1034
1035  @Override
1036  public void onTrimMemory(int level) {
1037      super.onTrimMemory(level);
1038      System.gc();
1039  }
1040
1041  @Override
1042  public void onTrimMemory(int level) {
1043      super.onTrimMemory(level);
1044      System.gc();
1045  }
1046
1047  @Override
1048  public void onTrimMemory(int level) {
1049      super.onTrimMemory(level);
1050      System.gc();
1051  }
1052
1053  @Override
1054  public void onTrimMemory(int level) {
1055      super.onTrimMemory(level);
1056      System.gc();
1057  }
1058
1059  @Override
1060  public void onTrimMemory(int level) {
1061      super.onTrimMemory(level);
1062      System.gc();
1063  }
1064
1065  @Override
1066  public void onTrimMemory(int level) {
1067      super.onTrimMemory(level);
1068      System.gc();
1069  }
1070
1071  @Override
1072  public void onTrimMemory(int level) {
1073      super.onTrimMemory(level);
1074      System.gc();
1075  }
1076
1077  @Override
1078  public void onTrimMemory(int level) {
1079      super.onTrimMemory(level);
1080      System.gc();
1081  }
1082
1083  @Override
1084  public void onTrimMemory(int level) {
1085      super.onTrimMemory(level);
1086      System.gc();
1087  }
1088
1089  @Override
1090  public void onTrimMemory(int level) {
1091      super.onTrimMemory(level);
1092      System.gc();
1093  }
1094
1095  @Override
1096  public void onTrimMemory(int level) {
1097      super.onTrimMemory(level);
1098      System.gc();
1099  }
1100
1101  @Override
1102  public void onTrimMemory(int level) {
1103      super.onTrimMemory(level);
1104      System.gc();
1105  }
1106
1107  @Override
1108  public void onTrimMemory(int level) {
1109      super.onTrimMemory(level);
1110      System.gc();
1111  }
1112
1113  @Override
1114  public void onTrimMemory(int level) {
1115      super.onTrimMemory(level);
1116      System.gc();
1117  }
1118
1119  @Override
1120  public void onTrimMemory(int level) {
1121      super.onTrimMemory(level);
1122      System.gc();
1123  }
1124
1125  @Override
1126  public void onTrimMemory(int level) {
1127      super.onTrimMemory(level);
1128      System.gc();
1129  }
1130
1131  @Override
1132  public void onTrimMemory(int level) {
1133      super.onTrimMemory(level);
1134      System.gc();
1135  }
1136
1137  @Override
1138  public void onTrimMemory(int level) {
1139      super.onTrimMemory(level);
1140      System.gc();
1141  }
1142
1143  @Override
1144  public void onTrimMemory(int level) {
1145      super.onTrimMemory(level);
1146      System.gc();
1147  }
1148
1149  @Override
1150  public void onTrimMemory(int level) {
1151      super.onTrimMemory(level);
1152      System.gc();
1153  }
1154
1155  @Override
1156  public void onTrimMemory(int level) {
1157      super.onTrimMemory(level);
1158      System.gc();
1159  }
1160
1161  @Override
1162  public void onTrimMemory(int level) {
1163      super.onTrimMemory(level);
1164      System.gc();
1165  }
1166
1167  @Override
1168  public void onTrimMemory(int level) {
1169      super.onTrimMemory(level);
1170      System.gc();
1171  }
1172
1173  @Override
1174  public void onTrimMemory(int level) {
1175      super.onTrimMemory(level);
1176      System.gc();
1177  }
1178
1179  @Override
1180  public void onTrimMemory(int level) {
1181      super.onTrimMemory(level);
1182      System.gc();
1183  }
1184
1185  @Override
1186  public void onTrimMemory(int level) {
1187      super.onTrimMemory(level);
1188      System.gc();
1189  }
1190
1191  @Override
1192  public void onTrimMemory(int level) {
1193      super.onTrimMemory(level);
1194      System.gc();
1195  }
1196
1197  @Override
1198  public void onTrimMemory(int level) {
1199      super.onTrimMemory(level);
1200      System.gc();
1201  }
1202
1203  @Override
1204  public void onTrimMemory(int level) {
1205      super.onTrimMemory(level);
1206      System.gc();
1207  }
1208
1209  @Override
1210  public void onTrimMemory(int level) {
1211      super.onTrimMemory(level);
1212      System.gc();
1213  }
1214
1215  @Override
1216  public void onTrimMemory(int level) {
1217      super.onTrimMemory(level);
1218      System.gc();
1219  }
1220
1221  @Override
1222  public void onTrimMemory(int level) {
1223      super.onTrimMemory(level);
1224      System.gc();
1225  }
1226
1227  @Override
1228  public void onTrimMemory(int level) {
1229      super.onTrimMemory(level);
1230      System.gc();
1231  }
1232
1233  @Override
1234  public void onTrimMemory(int level) {
1235      super.onTrimMemory(level);
1236      System.gc();
1237  }
1238
1239  @Override
1240  public void onTrimMemory(int level) {
1241      super.onTrimMemory(level);
1242      System.gc();
1243  }
1244
1245  @Override
1246  public void onTrimMemory(int level) {
1247      super.onTrimMemory(level);
1248      System.gc();
1249  }
1250
1251  @Override
1252  public void onTrimMemory(int level) {
1253      super.onTrimMemory(level);
1254      System.gc();
1255  }
1256
1257  @Override
1258  public void onTrimMemory(int level) {
1259      super.onTrimMemory(level);
1260      System.gc();
1261  }
1262
1263  @Override
1264  public void onTrimMemory(int level) {
1265      super.onTrimMemory(level);
1266      System.gc();
1267  }
1268
1269  @Override
1270  public void onTrimMemory(int level) {
1271      super.onTrimMemory(level);
1272      System.gc();
1273  }
1274
1275  @Override
1276  public void onTrimMemory(int level) {
1277      super.onTrimMemory(level);
1278      System.gc();
1279  }
1280
1281  @Override
1282  public void onTrimMemory(int level) {
1283      super.onTrimMemory(level);
1284      System.gc();
1285  }
1286
1287  @Override
1288  public void onTrimMemory(int level) {
1289      super.onTrimMemory(level);
1290      System.gc();
1291  }
1292
1293  @Override
1294  public void onTrimMemory(int level) {
1295      super.onTrimMemory(level);
1296      System.gc();
1297  }
1298
1299  @Override
1300  public void onTrimMemory(int level) {
1301      super.onTrimMemory(level);
1302      System.gc();
1303  }
1304
1305  @Override
1306  public void onTrimMemory(int level) {
1307      super.onTrimMemory(level);
1308      System.gc();
1309  }
1310
1311  @Override
1312  public void onTrimMemory(int level) {
1313      super.onTrimMemory(level);
1314      System.gc();
1315  }
1316
1317  @Override
1318  public void onTrimMemory(int level) {
1319      super.onTrimMemory(level);
1320      System.gc();
1321  }
1322
1323  @Override
1324  public void onTrimMemory(int level) {
1325      super.onTrimMemory(level);
1326      System.gc();
1327  }
1328
1329  @Override
1330  public void onTrimMemory(int level) {
1331      super.onTrimMemory(level);
1332      System.gc();
1333  }
1334
1335  @Override
1336  public void onTrimMemory(int level) {
1337      super.onTrimMemory(level);
1338      System.gc();
1339  }
1340
1341  @Override
1342  public void onTrimMemory(int level) {
1343      super.onTrimMemory(level);
1344      System.gc();
1345  }
1346
1347  @Override
1348  public void onTrimMemory(int level) {
1349      super.onTrimMemory(level);
1350      System.gc();
1351  }
1352
1353  @Override
1354  public void onTrimMemory(int level) {
1355      super.onTrimMemory(level);
1356      System.gc();
1357  }
1358
1359  @Override
1360  public void onTrimMemory(int level) {
1361      super.onTrimMemory(level);
1362      System.gc();
1363  }
1364
1365  @Override
1366  public void onTrimMemory(int level) {
1367      super.onTrimMemory(level);
1368      System.gc();
1369  }
1370
1371  @Override
1372  public void onTrimMemory(int level) {
1373      super.onTrimMemory(level);
1374      System.gc();
1375  }
1376
1377  @Override
1378  public void onTrimMemory(int level) {
1379      super.onTrimMemory(level);
1380      System.gc();
1381  }
1382
1383  @Override
1384  public void onTrimMemory(int level) {
1385      super.onTrimMemory(level);
1386      System.gc();
1387  }
1388
1389  @Override
1390  public void onTrimMemory(int level) {
1391      super.onTrimMemory(level);
1392      System.gc();
1393  }
1394
1395  @Override
1396  public void onTrimMemory(int level) {
1397      super.onTrimMemory(level);
1398      System.gc();
1399  }
1400
1401  @Override
1402  public void onTrimMemory(int level) {
1403      super.onTrimMemory(level);
1404      System.gc();
1405  }
1406
1407  @Override
1408  public void onTrimMemory(int level) {
1409      super.onTrimMemory(level);
1410      System.gc();
1411  }
1412
1413  @Override
1414  public void onTrimMemory(int level) {
1415      super.onTrimMemory(level);
1416      System.gc();
1417  }
1418
1419  @Override
1420  public void onTrimMemory(int level) {
1421      super.onTrimMemory(level);
1422      System.gc();
1423  }
1424
1425  @Override
1426  public void onTrimMemory(int level) {
1427      super.onTrimMemory(level);
1428      System.gc();
1429  }
1430
1431  @Override
1432  public void onTrimMemory(int level) {
1433      super.onTrimMemory(level);
1434      System.gc();
1435  }
1436
1437  @Override
1438  public void onTrimMemory(int level) {
1439      super.onTrimMemory(level);
1440      System.gc();
1441  }
1442
1443  @Override
1444  public void onTrimMemory(int level) {
1445      super.onTrimMemory(level);
1446      System.gc();
1447  }
1448
1449  @Override
1450  public void onTrimMemory(int level) {
1451      super.onTrimMemory(level);
1452      System.gc();
1453  }
1454
1455  @Override
1456  public void onTrimMemory(int level) {
1457      super.onTrimMemory(level);
1458      System.gc();
1459  }
1460
1461  @Override
1462  public void onTrimMemory(int level) {
1463      super.onTrimMemory(level);
1464      System.gc();
1465  }
1466
1467  @Override
1468  public void onTrimMemory(int level) {
1469      super.onTrimMemory(level);
1470      System.gc();
1471  }
1472
1473  @Override
1474  public void onTrimMemory(int level) {
1475      super.onTrimMemory(level);
1476      System.gc();
1477  }
1478
1479  @Override
1480  public void onTrimMemory(int level) {
1481      super.onTrimMemory(level);
1482      System.gc();
1483  }
1484
1485  @Override
1486  public void onTrimMemory(int level) {
1487      super.onTrimMemory(level);
1488      System.gc();
1489  }
1490
1491  @Override
1492  public void onTrimMemory(int level) {
1493      super.onTrimMemory(level);
1494      System.gc();
1495  }
1496
1497  @Override
1498  public void onTrimMemory(int level) {
1499      super.onTrimMemory(level);
1500      System.gc();
1501  }
1502
1503  @Override
1504  public void onTrimMemory(int level) {
1505      super.onTrimMemory(level);
1506      System.gc();
1507  }
1508
1509  @Override
1510  public void onTrimMemory(int level) {
1511      super.onTrimMemory(level);
1512      System.gc();
1513  }
1514
1515  @Override
```

```
151             }
152         }
153
154         @Override
155         public void onCancelled(DatabaseError databaseError) {
156
157             }
158         };
159
160         jobApplicationRef.addValueEventListener(getAllApplicationListener);
161
162     }
163     return view;
164 }
165
166 @Override
167 public void onAttach(Context context) {
168     super.onAttach(context);
169     if (context instanceof OnPostedFragmentInteractionListener) {
170         mListener = (OnPostedFragmentInteractionListener) context;
171     } else {
172         throw new RuntimeException(context.toString()
173             + " must implement OnListFragmentInteractionListener");
174     }
175 }
176
177 @Override
178 public void onDetach() {
179     super.onDetach();
180     jobsRef.removeEventListener(getAllJobListener);
181 }
182
183 public interface OnPostedFragmentInteractionListener{
184     void onPressPostedJob(Job job);
185 }
186
187 }
188 }
```

```
1 package com.marktrs.macapp.Fragment.Recruiter;
2
3 import android.content.Context;
4 import android.os.Bundle;
5 import android.support.v4.app.Fragment;
6 import android.support.v7.widget.GridLayoutManager;
7 import android.support.v7.widget.LinearLayoutManager;
8 import android.support.v7.widget.RecyclerView;
9 import android.util.Log;
10 import android.view.LayoutInflater;
11 import android.view.View;
12 import android.view.ViewGroup;
13 import android.widget.TextView;
14
15 import com.google.firebaseio.database.DataSnapshot;
16 import com.google.firebaseio.database.DatabaseError;
17 import com.google.firebaseio.database.DatabaseReference;
18 import com.google.firebaseio.database.FirebaseDatabase;
19 import com.google.firebaseio.database.Query;
20 import com.google.firebaseio.database.ValueEventListener;
21 import com.marktrs.macapp.Model.Job;
22 import com.marktrs.macapp.Model.JobApplication;
23 import com.marktrs.macapp.Model.User;
24 import com.marktrs.macapp.R;
25 import com.marktrs.macapp.Fragment.Recruiter.dummy.DummyContent;
26 import com.marktrs.macapp.Fragment.Recruiter.dummy.DummyContent.DummyItem;
27
28 import java.util.ArrayList;
29 import java.util.HashMap;
30 import java.util.List;
31 import java.util.Map;
32
33 /**
34  * A fragment representing a list of Items.
35  * <p>
36  * Activities containing this fragment MUST implement the <@link OnListFragmentInteractionListener>
37  * interface.
38 */
39 public class JobApplicationArchiveFragment extends Fragment {
40
41     // TODO: Customize parameter argument names
42     private static final String ARG_COLUMN_COUNT = "column-count";
43     private static final String JOB_KEY = "jobKey";
44     // TODO: Customize parameters
45     private int mColumnCount = 1;
46     private OnListFragmentInteractionListener mListener;
47
48     private FirebaseDatabase database;
49     private DatabaseReference applicationRef;
50     private ValueEventListener getApplication;
51     private ArrayList<JobApplication> jobApplications;
52     private ArrayList<User> profileMap;
53
54     private RecyclerView recyclerView;
55
56     private TextView noContentText;
57
58     private Job job;
59
60     /**
61      * Mandatory empty constructor for the fragment manager to instantiate the
62      * fragment (e.g. upon screen orientation changes).
63      */
64     public JobApplicationArchiveFragment() {
65     }
66
67     // TODO: Customize parameter initialization
68     @SuppressWarnings("unused")
69     public static JobApplicationArchiveFragment newInstance(Job job) {
70         JobApplicationArchiveFragment fragment = new JobApplicationArchiveFragment();
71         Bundle args = new Bundle();
72         args.putSerializable(JOB_KEY, job);
73         fragment.setArguments(args);
74         return fragment;
75     }
76 }
```

```

77     @Override
78     public void onCreate(Bundle savedInstanceState) {
79         super.onCreate(savedInstanceState);
80
81         database = FirebaseDatabase.getInstance();
82         this.job = (Job) getArguments().getSerializable(JOB_KEY);
83
84         if (getArguments() != null) {
85             mColumnCount = getArguments().getInt(ARG_COLUMN_COUNT);
86         }
87     }
88
89     @Override
90     public View onCreateView(LayoutInflater inflater, ViewGroup container,
91                             Bundle savedInstanceState) {
92         View view = inflater.inflate(R.layout.fragment_jobapplicationarchive_list, container, false);
93
94         // Set the adapter
95         if (view instanceof RecyclerView) {
96             Context context = view.getContext();
97             recyclerView = (RecyclerView) view;
98             if (mColumnCount <= 1) {
99                 recyclerView.setLayoutManager(new LinearLayoutManager(context));
100            } else {
101                recyclerView.setLayoutManager(new GridLayoutManager(context, mColumnCount));
102            }
103
104            applicationRef = database.getReference("JobApplications");
105            profileMap = new ArrayList<>();
106
107            getApplication = new ValueEventListener() {
108                @Override
109                public void onDataChange(DataSnapshot dataSnapshot) {
110                    jobApplications = new ArrayList<>();
111                    if (dataSnapshot.getChildrenCount() > 0) {
112                        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
113                            JobApplication application = snapshot.getValue(JobApplication.class);
114                            if (application.getJobId().equals(job.getJobId())) {
115                                jobApplications.add(application);
116                            }
117                        }
118                        recyclerView.setAdapter(new MyJobApplicationArchiveRecyclerViewAdapter(jobApplications,
119                         mListener));
120                    }
121                    if (jobApplications.isEmpty()){
122                        noContentText = (TextView) getActivity().findViewById(R.id.no_content);
123                        noContentText.setText("There are no application on this time");
124                        noContentText.setVisibility(View.VISIBLE);
125                    }
126                }
127
128                @Override
129                public void onCancelled(DatabaseError databaseError) {
130
131                }
132
133                applicationRef.addValueEventListener(getApplication);
134            }
135            return view;
136        }
137
138        @Override
139        public void onAttach(Context context) {
140            super.onAttach(context);
141            if (context instanceof OnListFragmentInteractionListener) {
142                mListener = (OnListFragmentInteractionListener) context;
143            } else {
144                throw new RuntimeException(context.toString()
145                        + " must implement OnListFragmentInteractionListener");
146            }
147        }
148
149        @Override
150        public void onDetach() {
151            super.onDetach();

```

```
152     mListener = null;
153     applicationRef.removeEventListener(getApplicationContext());
154     noContentText.setVisibility(View.GONE);
155     noContentText.setText("");
156 }
157
158 /**
159 * This interface must be implemented by activities that contain this
160 * fragment to allow an interaction in this fragment to be communicated
161 * to the activity and potentially other fragments contained in that
162 * activity.
163 * <p/>
164 * See the Android Training lesson <a href=
165 * "http://developer.android.com/training/basics/fragments/communicating.html"
166 * >Communicating with Other Fragments</a> for more information.
167 */
168 public interface OnListFragmentInteractionListener {
169     // TODO: Update argument type and name
170     void onListFragmentInteraction(JobApplication jobApplication);
171 }
172 }
173 }
```

```
1 package com.marktrs.macapp.Fragment.Recruiter;
2
3 import android.support.v7.widget.RecyclerView;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.TextView;
8 import android.widget.Toast;
9 import com.marktrs.macapp.Model.Job;
10 import com.marktrs.macapp.R;
11
12 import java.util.ArrayList;
13 import java.util.Map;
14
15 public class MyPostedJobRecyclerViewAdapter extends RecyclerView.Adapter<MyPostedJobRecyclerViewAdapter.ViewHolder> {
16
17     private ArrayList<Job> jobs;
18     private Map<String, Integer> jobApplicationCount;
19     private PostedJobFragment.OnPostedFragmentInteractionListener mListener;
20
21     public MyPostedJobRecyclerViewAdapter(ArrayList<Job> jobs, Map<String, Integer> jobApplicationCount,
22     PostedJobFragment.OnPostedFragmentInteractionListener listener) {
23         this.jobs = jobs;
24         this.jobApplicationCount = jobApplicationCount;
25         this.mListener = listener;
26     }
27
28     @Override
29     public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
30         View view = LayoutInflater.from(parent.getContext())
31             .inflate(R.layout.fragment_postedjob, parent, false);
32         return new ViewHolder(view);
33     }
34
35     @Override
36     public void onBindViewHolder(final ViewHolder holder, int position) {
37         holder.mItem = jobs.get(position);
38         holder.mIdView.setText(jobs.get(position).getJobName());
39         holder.mContentView.setText(jobs.get(position).getWorkplace());
40         holder.jobSymptom.setText(jobs.get(position).getSymptomType());
41         if(null != jobApplicationCount.get(jobs.get(position).getJobId())){
42             holder.applicationCounter.setText(jobApplicationCount.get(jobs.get(position).getJobId()).toString());
43         }else {
44             holder.applicationCounter.setText("0");
45         }
46         holder.mView.setOnClickListener(new View.OnClickListener() {
47             @Override
48             public void onClick(View v) {
49                 if (null != mListener) {
50                     mListener.onPressPostedJob(holder.mItem);
51                 }
52             }
53         });
54     }
55
56     @Override
57     public int getItemCount() {
58         return jobs.size();
59     }
60
61     public class ViewHolder extends RecyclerView.ViewHolder {
62         public final View mView;
63         public final TextView mIdView;
64         public final TextView mContentView;
65         public final TextView jobSymptom;
66         public final TextView applicationCounter;
67         public Job mItem;
68
69         public ViewHolder(View view) {
70             super(view);
71             mView = view;
72             mIdView = (TextView) view.findViewById(R.id.id);
73             mContentView = (TextView) view.findViewById(R.id.content);
74             jobSymptom = (TextView) view.findViewById(R.id.job_symptom);
75             applicationCounter = (TextView) view.findViewById(R.id.application_count);
```

```
76      }
77
78     @Override
79     public String toString() {
80         return super.toString() + " '" + mContentView.getText() + "'";
81     }
82 }
83 }
84 }
```

```
1 package com.marktrs.macapp.Fragment.Recruiter;
2
3 import android.content.res.ColorStateList;
4 import android.graphics.Color;
5 import android.graphics.Paint;
6 import android.support.v7.widget.RecyclerView;
7 import android.text.Layout;
8 import android.util.Log;
9 import android.view.LayoutInflater;
10 import android.view.View;
11 import android.view.ViewGroup;
12 import android.widget.Button;
13 import android.widget.TextView;
14
15 import com.google.firebaseio.database.DataSnapshot;
16 import com.google.firebaseio.database.DatabaseError;
17 import com.google.firebaseio.database.DatabaseReference;
18 import com.google.firebaseio.database.FirebaseDatabase;
19 import com.google.firebaseio.database.ValueEventListener;
20 import com.marktrs.macapp.Fragment.Recruiter.JobApplicationAchiveFragment.OnListFragmentInteractionListener;
21 import com.marktrs.macapp.Fragment.Recruiter.dummy.DummyContent.DummyItem;
22 import com.marktrs.macapp.Model.JobApplication;
23 import com.marktrs.macapp.Model.User;
24 import com.marktrs.macapp.R;
25
26 import java.util.ArrayList;
27
28 /**
29  * @link RecyclerView.Adapter that can display a @link DummyItem and makes a call to the
30  * specified @link OnListFragmentInteractionListener.
31  * TODO: Replace the implementation with code for your data type.
32 */
33 public class MyJobApplicationAchiveRecyclerViewAdapter extends RecyclerView.Adapter<
34     MyJobApplicationAchiveRecyclerViewAdapter.ViewHolder> {
35
36     private final ArrayList<JobApplication> mValues;
37     private final OnListFragmentInteractionListener mListener;
38     private final String ACCEPTED = "Accepted";
39     private final String DENIED = "Denied";
40     private final String WAITING = "waiting";
41     DatabaseReference profileRef;
42     FirebaseDatabase database;
43     ValueEventListener getProfileListener;
44
45     public MyJobApplicationAchiveRecyclerViewAdapter(ArrayList<JobApplication> items, OnListFragmentInteractionListener
46         listener) {
47         mValues = items;
48         mListener = listener;
49     }
50
51     @Override
52     public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
53         View view = LayoutInflater.from(parent.getContext())
54             .inflate(R.layout.fragment_jobapplicationachive, parent, false);
55         return new ViewHolder(view);
56     }
57
58     @Override
59     public void onBindViewHolder(final ViewHolder holder, int position) {
60         holder.mItem = mValues.get(position);
61         database = FirebaseDatabase.getInstance();
62         profileRef = database.getReference("User").child(mValues.get(position).getWorkerId());
63         if (!holder.mItem.getStatus().equals(DENIED)) {
64             if (holder.mItem.getStatus().equals(ACCEPTED)) {
65                 holder.lineView.setVisibility(View.VISIBLE);
66                 holder.facebookView.setVisibility(View.VISIBLE);
67                 holder.buttonPanel.setVisibility(View.GONE);
68                 getProfileListener = new ValueEventListener() {
69                     @Override
70                     public void onDataChange(DataSnapshot dataSnapshot) {
71                         User user = dataSnapshot.getValue(User.class);
72                         holder.workerName.setText(user.getFirstName() + " " + user.getLastName());
73                         holder.workerName.setTextColor(Color.parseColor("#4CAF50"));
74                         holder.workerLocation.setText(user.getWorker().getLocation());
75                         holder.workerSymptom.setText(user.getWorker().getSymptom());
76                         holder.workerSkill.setText(user.getWorker().getSkill());
77                     }
78                 };
79             }
80         }
81     }
82
83     public void onBindViewHolder(ViewHolder holder, int position, List payloads) {
84         if (!mValues.get(position).getStatus().equals(DENIED)) {
85             if (payloads.contains(ListUpdatePayload.REMOVE_WORKER)) {
86                 holder.lineView.setVisibility(View.GONE);
87                 holder.facebookView.setVisibility(View.GONE);
88                 holder.buttonPanel.setVisibility(View.VISIBLE);
89             }
90         }
91     }
92
93     @Override
94     public int getItemCount() {
95         return mValues.size();
96     }
97
98     public void setOnListFragmentInteractionListener(OnListFragmentInteractionListener listener) {
99         mListener = listener;
100    }
101 }
```

```

75             holder.workerEducation.setText(user.getWorker().getEducator());
76             holder.workerLine.setText(user.getWorker().getLineId());
77             holder.workerFacebook.setText(user.getWorker().getFacebookId());
78         }
79
80         @Override
81         public void onCancelled(DatabaseError databaseError) {
82
83     };
84
85
86     } else if (holder.mItem.getStatus().equals(WAITING)) {
87         holder.buttonPanel.setVisibility(View.VISIBLE);
88         getProfileListener = new ValueEventListener() {
89             @Override
90             public void onDataChange(DataSnapshot dataSnapshot) {
91                 User user = dataSnapshot.getValue(User.class);
92                 holder.workerName.setText(user.getFirstName() + " " + user.getLastName());
93                 holder.workerLocation.setText(user.getWorker().getLocation());
94                 holder.workerSymptom.setText(user.getWorker().getSymptom());
95                 holder.workerSkill.setText(user.getWorker().getSkill());
96                 holder.workerEducation.setText(user.getWorker().getEducator());
97                 holder.workerLine.setText(user.getWorker().getLineId());
98                 holder.workerFacebook.setText(user.getWorker().getFacebookId());
99
100            holder.denyButton.setOnClickListener(new View.OnClickListener() {
101                @Override
102                public void onClick(View view) {
103                    holder.workerName.setText(DENIED);
104                    holder.workerName.setTextColor(Color.parseColor("#BDBDBD"));
105                    holder.workerName.setPaintFlags(holder.workerName.getPaintFlags() | Paint.
106 STRIKE_THRU_TEXT_FLAG);
107                    holder.buttonPanel.setVisibility(View.GONE);
108                    if (null != mListener) {
109                        holder.mItem.setStatus(DENIED);
110                        mListener.onListFragmentInteraction(holder.mItem);
111                    }
112                }
113            });
114
115            holder.acceptButton.setOnClickListener(new View.OnClickListener() {
116                @Override
117                public void onClick(View view) {
118                    holder.lineView.setVisibility(View.VISIBLE);
119                    holder.facebookView.setVisibility(View.VISIBLE);
120                    holder.acceptButton.setVisibility(View.GONE);
121                    holder.denyButton.setVisibility(View.GONE);
122                    if (null != mListener) {
123                        holder.mItem.setStatus(ACCEPTED);
124                        mListener.onListFragmentInteraction(holder.mItem);
125                    }
126                }
127            });
128        }
129
130        @Override
131        public void onCancelled(DatabaseError databaseError) {
132
133    };
134
135    }
136    profileRef.addListenerForSingleValueEvent(getProfileListener);
137 }else {
138     holder.workerName.setTextColor(Color.parseColor("#BDBDBD"));
139     holder.workerName.setPaintFlags(holder.workerName.getPaintFlags() | Paint.STRIKE_THRU_TEXT_FLAG);
140 }
141
142
143 @Override
144 public int getItemCount() {
145     return mValues.size();
146 }
147
148 public class ViewHolder extends RecyclerView.ViewHolder {
149     public final View mView;

```

```
150     public final TextView workerName;
151     public final TextView workerLocation;
152     public TextView workerSkill;
153     public TextView workerEducation;
154     public TextView workerSymptom;
155     public TextView workerLine;
156     public TextView workerFacebook;
157     public Button acceptButton;
158     public Button denyButton;
159     public View lineView;
160     public View facebookView;
161     public JobApplication mItem;
162     public View buttonPanel;
163
164     public ViewHolder(View view) {
165         super(view);
166         mView = view;
167         workerName = (TextView) view.findViewById(R.id.worker_name);
168         workerLocation = (TextView) view.findViewById(R.id.worker_location);
169         workerSymptom = (TextView) view.findViewById(R.id.worker_symptoms);
170         workerSkill = (TextView) view.findViewById(R.id.worker_skill);
171         workerEducation = (TextView) view.findViewById(R.id.worker_education);
172         acceptButton = (Button) view.findViewById(R.id.accept_button);
173         denyButton = (Button) view.findViewById(R.id.deny_button);
174         lineView = view.findViewById(R.id.line_view);
175         facebookView = view.findViewById(R.id.facebook_view);
176         workerLine = (TextView) view.findViewById(R.id.worker_line_id);
177         workerFacebook = (TextView) view.findViewById(R.id.worker_facebook_id);
178         buttonPanel = view.findViewById(R.id.button_panel);
179     }
180 }
181 }
182 }
```