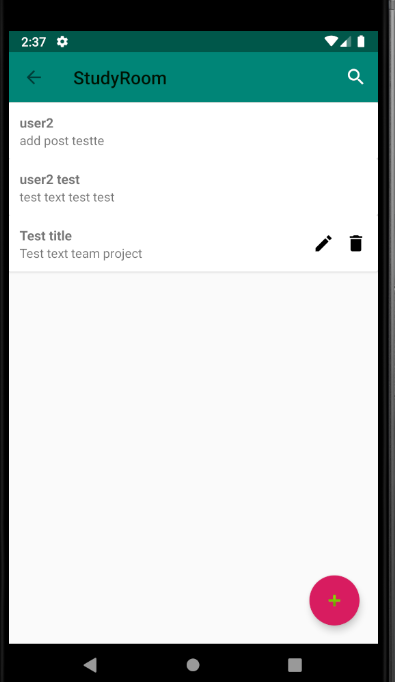
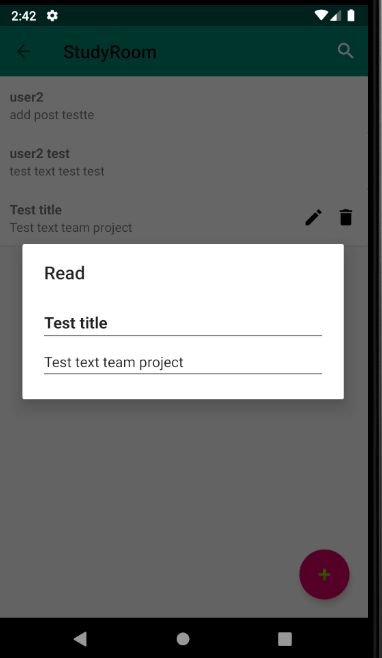
Lost and Found Bulletin Board section

Lost and Found Bulletin board section for students who have lost things and who have found others stuff will post their notice.

To make this section we created

UI part

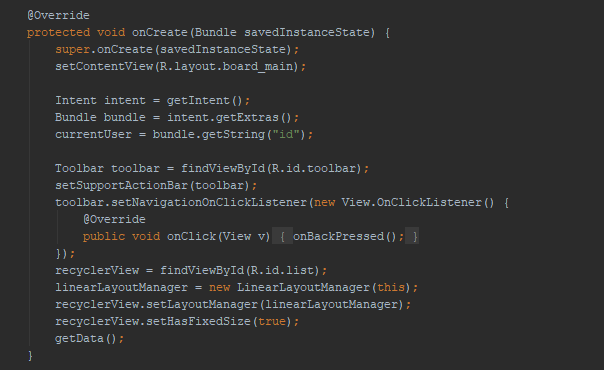
* board\_main.xml for entire UI
* dialog\_layout.xml for create, edit and read posts
* listview\_activity.xml for display single line of lists

Coding part

* Mainboard.java
* FirebaseAdapter.java
* FirebaseRecyclerAdapter.java
* PostAdapter.java
* Texts.java

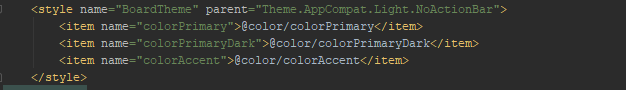
Mainboard.java

Mainboard.java for creating activity

onCreate method  


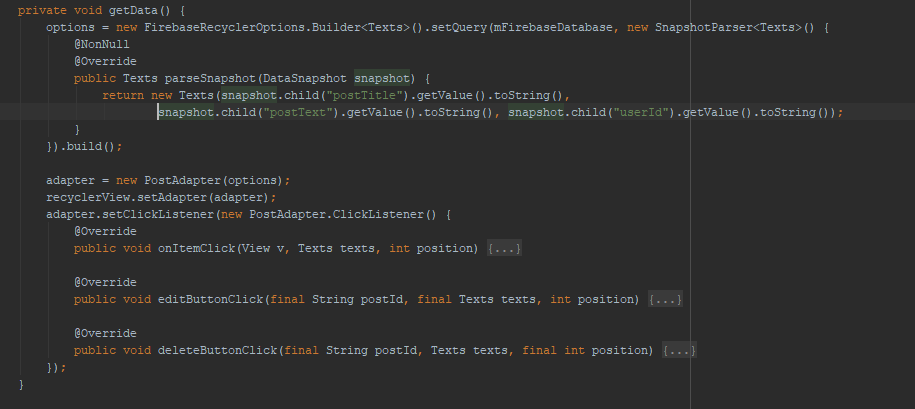
* Getting passed user id
* Creating toolbar :in this activity we used theme without toolbar for create customized toolbar





* We used RecyclerView to display posts because we have data collection elements change at runtime based on user action. So we need to work with
  + LayoutManager helps in positioning the items
  + RecyclerVIew.Adapter to handle data collection and bind it to view
* Calls getData() method for retrieving data from Google Firebase database

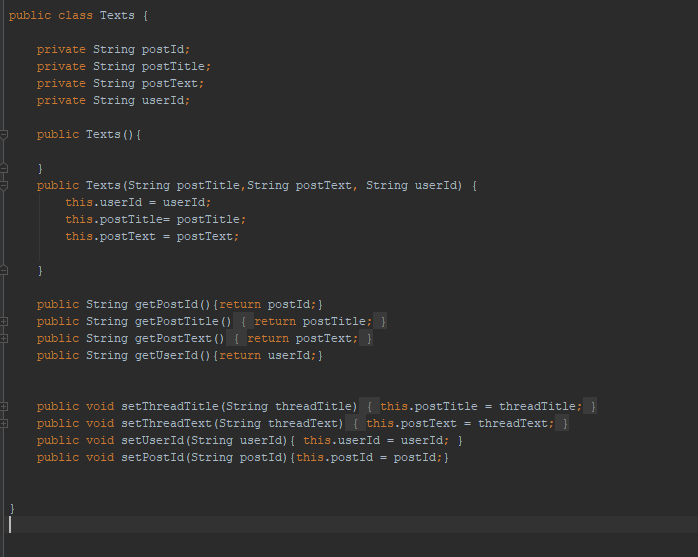
getData method

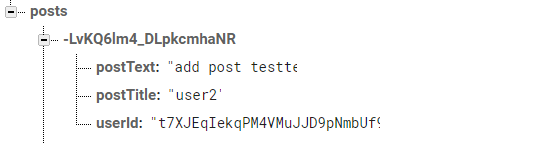


We are using Google Firebase Realtime Database to store and sync app data in realtime. Firebase Realtime Database is NoSQL cloud database stores data as one large JSON tree. More info <https://firebase.google.com/docs/database>

**Defining a data Model**

Every RecyclerView is backed by a source for data. We defined Texts class which represents the data model being displayed by the RecyclerView.

Texts.java 

* The getters and setters allows Firebase to map the data to field names 
* Empty constructor is required for Firebase automatic data mapping.

**Creating the RecyclerView.Adapter (PostAdapter.java)**

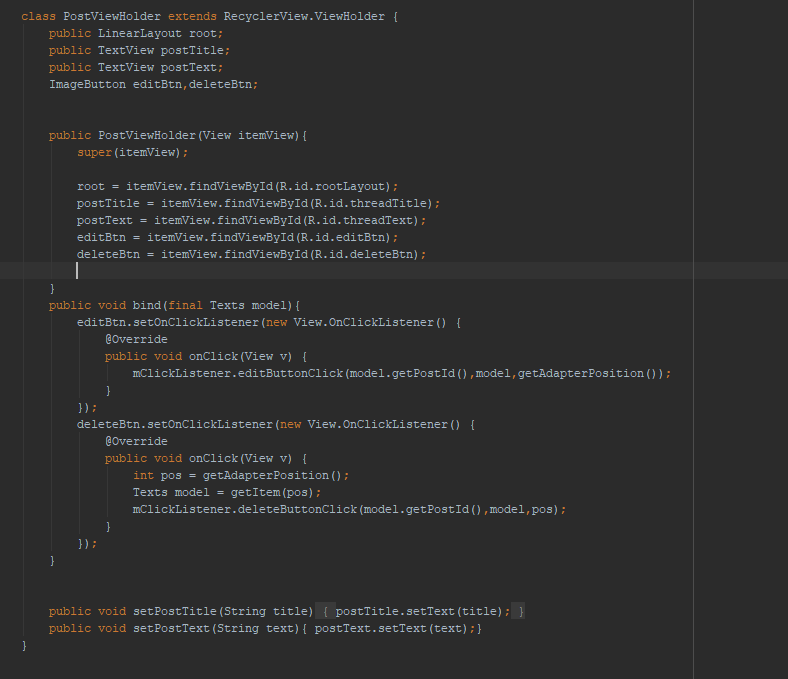
Creating adapter for populate the data into the RecyclerView. Firebase offers RecyclerView adapter for Realtime database:

* FirebaseRecyclerAdapter - binds a Query to a RecyclerView and responds to all real-time events included items being added, removed, moved, or changed.

We downloaded FirebaseRecyclerAdapter.java, FirebaseAdapter.java files from FirebaseUI github and customized for our StudyRoom app. Created custom FirebaseRecyclerAdapter named PostAdapter.java

public class PostAdapter extends FirebaseRecyclerAdapter<Texts, PostAdapter.PostViewHolder>

PostAdapter extends from FirebaseRecyclerAdapter class. Adapter requires the existence of ViewHolder object which describes and provides access to all views within each row. We create PostViewHolder and each row contains 2 TextView, 2 ImageButton.

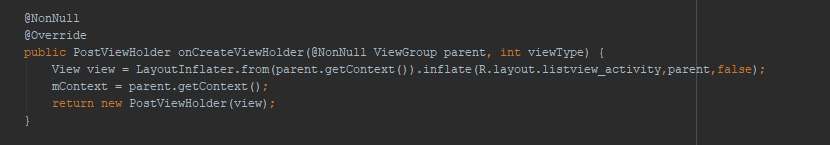


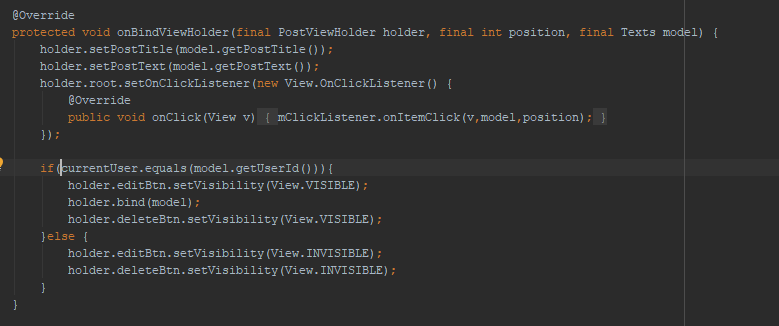
In PostAdapter class we created Clicklistener interface for user read,edit and deleting post.

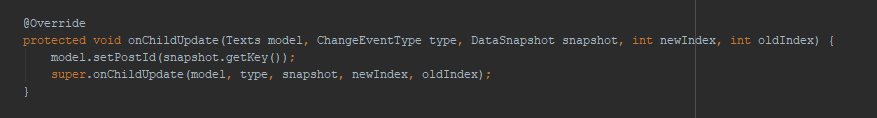
public interface ClickListener {  
 void onItemClick(View v,Texts texts, int position);  
 void editButtonClick(String postId, Texts texts, int position);  
 void deleteButtonClick(String postId, Texts texts, int position);  
}

public void setClickListener(ClickListener listener){  
 mClickListener =listener;  
}

onCreateViewHolder method to inflate the item layout and create the holder.



onBindViewHolder method to set the attributes based on data 

onChildUpdate method will notify child updated type 

filterCondition method for search

@Override  
protected boolean filterCondition(Texts model, String filterPattern) {  
 return model.getPostTitle().toLowerCase().contains(filterPattern) || model.getPostText().toLowerCase().contains(filterPattern);  
}

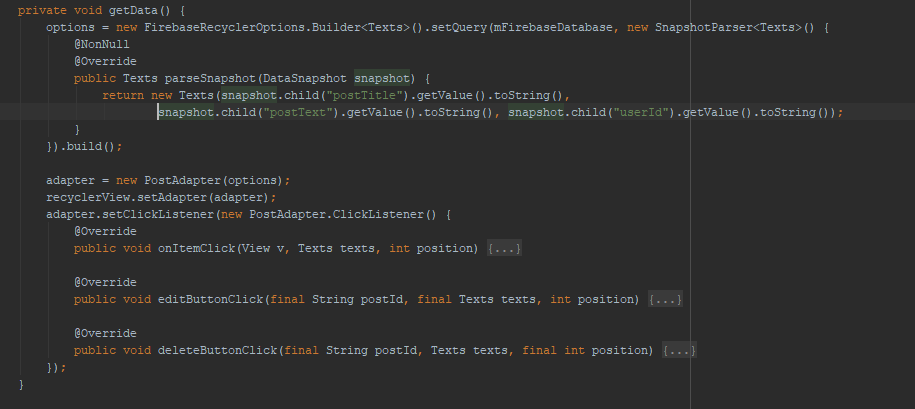
Lastly our PostAdapter class constructer will pass configure of the adapter

public PostAdapter(FirebaseRecyclerOptions<Texts> options){  
 super(options,true);  
}

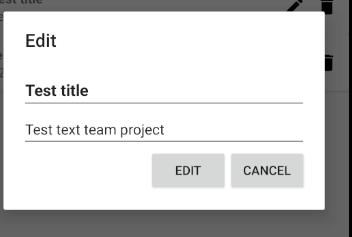
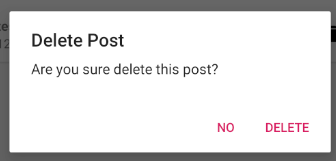
Now back to Mainboard.java

Retrieve an instance of database using getInstance() and reference the location

DatabaseReference mFirebaseDatabase = FirebaseDatabase.getInstance().getReference("posts");

getData() method 

* Configure the adapter by building FirebaseRecyclerOptions and customizing how data model class is parsed
* Creating PostAdapter object and binding the adapter to the RecyclerView
* Overriding Clicklistener interface methods for read,edit,delete
  + This click listener methods will create alertdialog

The FirebaseRecyclerAdapter uses an event listener to monitor changes to the Firebase query. To begin listening for data, call the startListening() method.

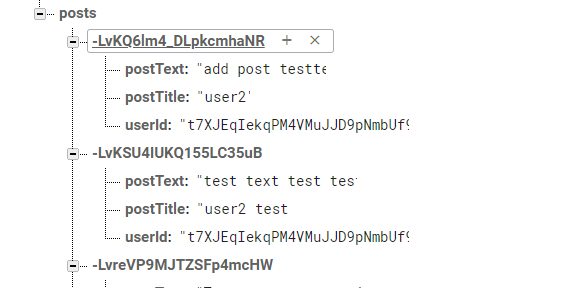
@Override  
protected void onStart() {  
 super.onStart();  
 adapter.startListening();  
}

The stopListening() call removes the event listener and all data in the adapter.

@Override  
protected void onStop() {  
 super.onStop();  
 adapter.stopListening();  
}

onClick method create new post

public void onClick(View v) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(MainBoard.this);  
 LayoutInflater inflater = (MainBoard.this).getLayoutInflater();  
 final View dialog = inflater.inflate(R.layout.dialog\_layout, null);  
 txtTitle = dialog.findViewById(R.id.editTxtTitle);  
 txtText = dialog.findViewById(R.id.editTxtText);  
 builder.setTitle("Create");  
 builder.setView(dialog);  
 final AlertDialog alertDialog = builder.show();  
 Button createBtn = alertDialog.findViewById(R.id.createBtn);  
 createBtn.setVisibility(View.VISIBLE);  
 createBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 String postTitle = txtTitle.getText().toString();  
 String postText = txtText.getText().toString();  
  
 String postId = mFirebaseDatabase.push().getKey();  
 Texts post = new Texts(postTitle, postText, currentUser);  
 mFirebaseDatabase.child(postId).setValue(post);  
 alertDialog.dismiss();  
 }  
 });  
 Button cancelBtn = alertDialog.findViewById(R.id.cancelBtn);  
 cancelBtn.setVisibility(View.VISIBLE);  
 cancelBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 alertDialog.dismiss();  
 }  
 });  
}

* onClick method will create alertdialog contains 2 editText widget for getting post Title,Text content.
* String postId = mFirebaseDatabase.push().getKey();
  + Generates new post id
* mFirebaseDatabase.child(postId).setValue(post);
  + New Generated key will new child of current location. When save an object the responses from any getters will be saved as children of new child. 

onCreateOptionsMenu()

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.menu, menu);  
  
 MenuItem search = menu.findItem(R.id.action\_search);  
  
 SearchView searchView = (SearchView) search.getActionView();  
 searchView.setQueryHint("Search");  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 return false;  
 }  
  
 @Override  
 public boolean onQueryTextChange(String newText) {  
 TextView noresultText = findViewById(R.id.emptyView);  
  
 adapter.getFilter().filter(newText);  
  
 return false;  
 }  
 });  
 return super.onCreateOptionsMenu(menu);  
}

* User to enter a search query and submit a request to a search provider. Shows a list of query suggestions or results, if available