Status Report: Instagram Clone

**Name:** Abdullah Bin Asad

**Application Name:** Instagram Clone

**Git Repository:** <https://github.com/Abdullah2Cool/Instagram-Clone>

**Functionality:** Login, register, see your profile info and store everything in Firebase

**Date:** September 16 – October 10

**Tutorial Series:** <https://www.youtube.com/watch?v=qpJRgr6HzAw&list=PLgCYzUzKIBE9XqkckEJJA0I1wVKbUAOdv>

**Specs:** Attaching a data model to the firebase database

**Major Challenge/Lesson:** Adding and getting data from firebase

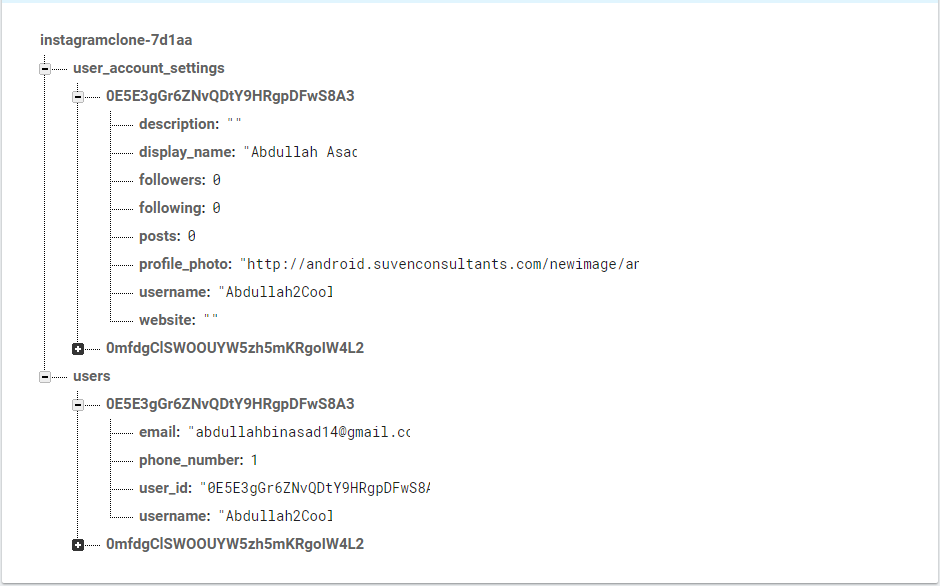
* You can create data models that allow you to hold the info locally and help store it in firebase
* These models also define the structure the way things are stored in firebase
* Here is an example of the User model (it must be named the same way in firebase)

public class User {  
 private String user\_id;  
 private long phone\_number;  
 private String email;  
 private String username;  
  
 public User(String user\_id, long phone\_number, String email, String username) {  
 this.user\_id = user\_id;  
 this.phone\_number = phone\_number;  
 this.email = email;  
 this.username = username;  
 }  
  
 public User () {  
  
 }  
  
 public String getUser\_id() {  
 return user\_id;  
 }  
  
 public void setUser\_id(String user\_id) {  
 this.user\_id = user\_id;  
 }  
  
 public long getPhone\_number() {  
 return phone\_number;  
 }  
  
 public void setPhone\_number(long phone\_number) {  
 this.phone\_number = phone\_number;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "user\_id='" + user\_id + '\'' +  
 ", phone\_number='" + phone\_number + '\'' +  
 ", email='" + email + '\'' +  
 ", username='" + username + '\'' +  
 '}';  
 }  
}

* It requires an empty constructor and all the default getters and setters
* The toString() method is optional
* Here is how you add a new user to the database:

public void addNewUser(String email, String username, String description, String website, String profie\_photoe) {  
 User user = new User(userID, 1, email, util\_StringManipulation.*condenseUsername*(username));  
  
 myRef.child(mContext.getString(R.string.*dbname\_users*))  
 .child(userID)  
 .setValue(user);  
  
 UserAccountSettings settings = new UserAccountSettings(  
 description,  
 username,  
 0,  
 0,  
 0,  
 profie\_photoe,  
 util\_StringManipulation.*condenseUsername*(username),  
 website  
 );  
  
 myRef.child(mContext.getString(R.string.*dbname\_user\_account\_settings*))  
 .child(userID)  
 .setValue(settings);  
}

* User account settings is another model that is not shown here which deals with other pieces of information such as #followers and the path to the profile photo
* As a result, here is what the structure of the database in firebase looks like:



* Firebase uses all the getter and setter methods to create appropriate fields and populate their value based on the user model we created
* In order to retrieve the user info, I created a third model called userSettings that contains a User object and a User\_Account\_Settings object:

package com.example.hafiz.instagramclone.Models;  
  
*/\*\*  
 \* Created by hafiz on 10/6/2017.  
 \*/*public class UserSettings {  
 private User user;  
 private UserAccountSettings userAccountSettings;  
  
 public UserSettings(User user, UserAccountSettings userAccountSettings) {  
 this.user = user;  
 this.userAccountSettings = userAccountSettings;  
 }  
  
 public UserSettings() {  
 }  
  
 public User getUser() {  
 return user;  
 }  
  
 public void setUser(User user) {  
 this.user = user;  
 }  
  
 public UserAccountSettings getUserAccountSettings() {  
 return userAccountSettings;  
 }  
  
 public void setUserAccountSettings(UserAccountSettings userAccountSettings) {  
 this.userAccountSettings = userAccountSettings;  
 }  
  
 @Override  
 public String toString() {  
 return "UserSettings{" +  
 "user=" + user +  
 ", userAccountSettings=" + userAccountSettings +  
 '}';  
 }  
}

* To get the info from firebase and populate the widgets, I use this function:
* */\*\*  
   \* Retrieves the account settings for the user currently logged in  
   \* Database: user\_account\_settings node  
   \** ***@param*** *dataSnapshot  
   \** ***@return*** *\*/*public UserSettings getUserSettings(DataSnapshot dataSnapshot) {  
   Log.*d*(*TAG*, "getUserAccountSettings: retrieving user account settings from firebase.");  
   UserAccountSettings settings = new UserAccountSettings();  
   User user = new User();  
    
   for (DataSnapshot ds : dataSnapshot.getChildren()) {  
    
   // user\_account\_settings node  
   if (ds.getKey().equals(mContext.getString(R.string.*dbname\_user\_account\_settings*))) {  
   Log.*d*(*TAG*, "getUserSettings: datasnapshot: " + ds);  
    
   try {  
   settings.setDisplay\_name(  
   ds.child(userID)  
   .getValue(UserAccountSettings.class)  
   .getDisplay\_name()  
   );  
   settings.setUsername(  
   ds.child(userID)  
   .getValue(UserAccountSettings.class)  
   .getUsername()  
   );  
   settings.setDescription(  
   ds.child(userID)  
   .getValue(UserAccountSettings.class)  
   .getDescription()  
   );  
   settings.setWebsite(  
   ds.child(userID)  
   .getValue(UserAccountSettings.class)  
   .getWebsite()  
   );  
   settings.setProfile\_photo(  
   ds.child(userID)  
   .getValue(UserAccountSettings.class)  
   .getProfile\_photo()  
   );  
   settings.setPosts(  
   ds.child(userID)  
   .getValue(UserAccountSettings.class)  
   .getPosts()  
   );  
   settings.setFollowing(  
   ds.child(userID)  
   .getValue(UserAccountSettings.class)  
   .getFollowing()  
   );  
   settings.setFollowers(  
   ds.child(userID)  
   .getValue(UserAccountSettings.class)  
   .getFollowers()  
   );  
   Log.*d*(*TAG*, "getUserSettings: retrieved user\_account\_settings information: " + settings.toString());  
   } catch (NullPointerException e) {  
   Log.*d*(*TAG*, "getUserAccountSettings: NullPointerException: " + e.getMessage());  
   }  
   }  
    
   // users node  
   if (ds.getKey().equals(mContext.getString(R.string.*dbname\_users*))) {  
   Log.*d*(*TAG*, "getUserSettings: datasnapshot: " + ds);  
   try {  
   user.setUsername(  
   ds.child(userID)  
   .getValue(User.class)  
   .getUsername()  
   );  
   user.setEmail(  
   ds.child(userID)  
   .getValue(User.class)  
   .getEmail()  
   );  
   user.setPhone\_number(  
   ds.child(userID)  
   .getValue(User.class)  
   .getPhone\_number()  
   );  
   user.setUser\_id(  
   ds.child(userID)  
   .getValue(User.class)  
   .getUser\_id()  
   );  
   Log.*d*(*TAG*, "getUserSettings: retrieved user information: " + user.toString());  
   } catch (NullPointerException e) {  
   Log.*d*(*TAG*, "getUserAccountSettings: NullPointerException: " + e.getMessage());  
   }  
   }  
   }  
    
   return new UserSettings(user, settings);  
  }
* This method fetches the info from FireBase and updates it locally in my models. Then I can use this info to populate widgets:

private void setProfileWidgets (UserSettings userSettings) {  
 Log.*d*(*TAG*, "setProfileWidgets: setting widgets with data retrieved from firebase database");  
  
 User user = userSettings.getUser();  
 UserAccountSettings settings = userSettings.getUserAccountSettings();  
  
 util\_UniversalImageLoader.*setImage*(settings.getProfile\_photo(), mProfilePhoto, null, "");  
 mDisplayName.setText(settings.getDisplay\_name());  
 mUsername.setText(user.getUsername());  
 mWebsite.setText(settings.getWebsite());  
 mDescription.setText(settings.getDescription());  
 mPosts.setText(String.*valueOf*(settings.getPosts()));  
 mFollowing.setText(String.*valueOf*(settings.getFollowing()));  
 mFollowers.setText(String.*valueOf*(settings.getFollowers()));  
 mProgressBar.setVisibility(View.*GONE*);  
}

Other things I’ve learned but not mentioned above:

* Loading images from different sources
* Building layouts and an overall understanding of how to create and nest reusable elements
* Logging in and logging out
* Signing up new users
* Sending verification emails and making sure they have been verified
* Querying the database and checking for certain conditions (if a username already exists)

**Sources:**

<https://www.youtube.com/watch?v=qpJRgr6HzAw&list=PLgCYzUzKIBE9XqkckEJJA0I1wVKbUAOdv>