```
public class LoginActivity extends AppCompatActivity {

    TwitterLoginButton loginButton;
    TwitterSession session;
    String token;
    String secret;
    Boolean isUserAuthorized = false;

    TextView infoTextView;

    SharedPreferences mShared;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Twitter.initialize(this);
        setContentView(R.layout.activity_login);
        infoTextView = findViewById(R.id.display);
        TwitterConfig config = new TwitterConfig.Builder(this)
                .logger(new DefaultLogger(Log.DEBUG))
                .twitterAuthConfig(new TwitterAuthConfig(

getString(R.string.com_twitter_sdk_android_CONSUMER_KEY),

getString(R.string.com_twitter_sdk_android_CONSUMER_SECRET)))
                .debug(true)
                .build();
        Twitter.initialize(config);

        getSessionInfo();
        if (!isUserAuthorized) {
            loginButton = findViewById(R.id.login_button);
            loginButton.setCallback(new Callback<TwitterSession>() {
                @Override
                public void success(Result<TwitterSession> result) {
                    session =
TwitterCore.getInstance().getSessionManager().getActiveSession();
                    TwitterAuthToken authToken =
session.getAuthToken();
                    saveSessionInfo(authToken.token,
authToken.secret);

                    Intent intent = new Intent(LoginActivity.this,
MainActivity.class);
                    intent.putExtra("userId", session.getUserId());
                    startActivity(intent);

                }

                @Override
                public void failure(TwitterException exception) {
                    Toast.makeText(LoginActivity.this, "Ошибка
авторизации!", Toast.LENGTH_LONG).show();
                }
            });
        } else {
            Intent intent = new Intent(LoginActivity.this,
MainActivity.class);
```

```
            session =
TwitterCore.getInstance().getSessionManager().getActiveSession();
            intent.putExtra("userId", session.getUserId());
            startActivity(intent);
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        // Pass the activity result to the login button.
        loginButton.onActivityResult(requestCode, resultCode, data);
    }

    private void saveSessionInfo(String curToken, String curSecret){
        mShared = getPreferences(MODE_PRIVATE);
        SharedPreferences.Editor mEditor = mShared.edit();
        mEditor.putBoolean("isUserAuthorized", true);
        mEditor.putString("token", curToken);
        mEditor.putString("secret", curSecret);
        mEditor.commit();
    }

    private void getSessionInfo(){
        mShared = getPreferences(MODE_PRIVATE);
        SharedPreferences.Editor mEditor = mShared.edit();
        isUserAuthorized = mShared.getBoolean("isUserAuthorized",
false);
        token = mShared.getString("token", "");
        secret = mShared.getString("secret", "");
    }
}


abstract class TimelineActivity extends AppCompatActivity {
    protected RecyclerView tweetsRecyclerView;
    protected TweetAdapter tweetAdapter;
    protected AsyncTask<Void, Void, Void> task;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Twitter.initialize(this);

        initToolbar();

        tweetsRecyclerView = findViewById(R.id.tweets_recycler_view);
        tweetsRecyclerView.setLayoutManager(new
LinearLayoutManager(this));
        tweetAdapter = new TweetAdapter(TimelineActivity.this);
        tweetsRecyclerView.setAdapter(tweetAdapter);

        initTask();
    }

    void initToolbar() {
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
    }

    abstract void initTask();

    void loadTweets() {
        task.execute();
    }
}


abstract class StandardTimelineActivity extends TimelineActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        loadTweets();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.info_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_home) {
            Intent intent = new Intent(this, MainActivity.class);
            startActivity(intent);
        }

        if (item.getItemId() == R.id.action_search) {
            Intent intent = new Intent(this, SearchActivity.class);
            startActivity(intent);
        }

        if (item.getItemId() == R.id.action_add_tweet) {
            final TwitterSession session =
TwitterCore.getInstance().getSessionManager()
                    .getActiveSession();
            final Intent intent = new
ComposerActivity.Builder(StandardTimelineActivity.this)
                    .session(session)
                    .darkTheme()
                    .createIntent();
            startActivity(intent);
        }
        return true;
    }
}


public class SearchActivity extends TimelineActivity {
    private EditText searchText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(R.layout.activity_search);
        super.onCreate(savedInstanceState);
    }

    @Override
    void initToolbar() {
        super.initToolbar();
        searchText = findViewById(R.id.request_edit_text);

        searchText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start,
int count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {}

            @Override
            public void afterTextChanged(Editable s) {
                task = new SearchTask(tweetAdapter, searchText);
                loadTweets();
            }
        });
    }

    @Override
    void initTask() {
        task = new SearchTask(tweetAdapter, searchText);
    }
}


public class MainActivity extends StandardTimelineActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(R.layout.activity_main);
        super.onCreate(savedInstanceState);
        getSupportActionBar().setTitle("Home");
    }

    @Override
    void initTask(){
        task = new TimelineTask(tweetAdapter, MainActivity.this);
    }
}



public class TweetAdapter extends
RecyclerView.Adapter<TweetAdapter.TweetViewHolder> {
    private static final String TWITTER_RESPONSE_FORMAT = "EEE MMM dd
HH:mm:ss ZZZZZ yyyy"; // Thu Oct 26 07:31:08 +0000 2017
    private static final String MONTH_DAY_FORMAT = "MMM d"; // Oct 26

    private List<Tweet> tweetList = new ArrayList<>();
    private Context context;
    public TweetAdapter(Context c) {
        context = c;
    }

    @Override
    public TweetViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.tweet_item_view, parent, false);
        return new TweetViewHolder(view);
    }

    @Override
    public void onBindViewHolder(TweetViewHolder holder, int position)
{
        holder.bind(tweetList.get(position));
    }

    @Override
    public int getItemCount() {
        return tweetList.size();
    }

    public void setItems(Collection<Tweet> tweets) {
        tweetList.addAll(tweets);
        notifyDataSetChanged();
    }

    public User getUser() {
        return tweetList.get(0).user;
    }

    public void clearItems() {
        tweetList.clear();
        notifyDataSetChanged();
    }

    class TweetViewHolder extends RecyclerView.ViewHolder {
        private ImageView userImageView;
        private TextView nameTextView;
        private TextView nickTextView;
        private TextView creationDateTextView;
        private TextView contentTextView;
        private ImageView tweetImageView;
        private TextView retweetsTextView;
        private TextView likesTextView;

        private ImageView isLiked;
        private ImageView isRetweeted;

        public TweetViewHolder(View itemView) {
            super(itemView);
            userImageView =
itemView.findViewById(R.id.profile_image_view);
            nameTextView =
itemView.findViewById(R.id.author_name_text_view);
            nickTextView =
itemView.findViewById(R.id.author_nick_text_view);
```

```
            creationDateTextView =
itemView.findViewById(R.id.creation_date_text_view);
            contentTextView =
itemView.findViewById(R.id.tweet_content_text_view);
            tweetImageView =
itemView.findViewById(R.id.tweet_image_view);
            retweetsTextView =
itemView.findViewById(R.id.retweets_text_view);
            likesTextView =
itemView.findViewById(R.id.likes_text_view);
            isRetweeted =
itemView.findViewById(R.id.retweet_image_view);
            isLiked = itemView.findViewById(R.id.like_image_view);
        }

        public void bind(Tweet tweet) {
            nameTextView.setText(tweet.user.name);
            nickTextView.setText(tweet.user.screenName);
            contentTextView.setText(tweet.text);

retweetsTextView.setText(String.valueOf(tweet.retweetCount));

likesTextView.setText(String.valueOf(tweet.favoriteCount));

            String creationDateFormatted =
getFormattedDate(tweet.createdAt);
            creationDateTextView.setText(creationDateFormatted);


Picasso.get().load(UserUtils.getProfileImageUrlHttps(tweet.user,

UserUtils.AvatarSize.REASONABLY_SMALL)).into(userImageView);

            if (tweet.favorited) {
                isLiked.setImageResource(R.drawable.like);
            } else {
                isLiked.setImageResource(R.drawable.not_like);
            }
            if (tweet.retweeted) {
                isRetweeted.setImageResource(R.drawable.retweet);
            }else {
                isRetweeted.setImageResource(R.drawable.not_retweet);
            }
            if (TweetMediaUtils.hasPhoto(tweet)) {
                String tweetPhotoUrl =
TweetMediaUtils.getPhotoEntity(tweet).mediaUrl;

Picasso.get().load(tweetPhotoUrl).into(tweetImageView);
            } else {
                tweetImageView.setVisibility(View.GONE);
            }

            linkifyProfile(tweet);
            setLikeAction(tweet);
        }

        private String getFormattedDate(String rawDate) {
            SimpleDateFormat utcFormat = new
SimpleDateFormat(TWITTER_RESPONSE_FORMAT, Locale.ROOT);
            SimpleDateFormat displayedFormat = new
SimpleDateFormat(MONTH_DAY_FORMAT, Locale.getDefault());
            try {
                Date date = utcFormat.parse(rawDate);
                return displayedFormat.format(date);
            } catch (ParseException e) {
                throw new RuntimeException(e);
            }
        }

        void linkifyProfile(final Tweet displayTweet) {
            if (displayTweet != null && displayTweet.user != null) {
                userImageView.setOnClickListener(v -> {
                    Intent intent = new Intent(context,
UserInfoActivity.class);
                    intent.putExtra("userId", displayTweet.user.id);
                    context.startActivity(intent);
                });

                nameTextView.setOnClickListener(v -> {
                    Intent intent = new Intent(context,
UserInfoActivity.class);
                    intent.putExtra("userId", displayTweet.user.id);
                    context.startActivity(intent);
                });
            }
        }

        void setLikeAction(final Tweet displayTweet) {
            if (displayTweet != null && displayTweet.user != null) {
                isLiked.setOnClickListener(v -> {
                    if (displayTweet.favorited) {
                        unfavorite(displayTweet);
                    } else {
                        favorite(displayTweet);
                    }
                });

                isRetweeted.setOnClickListener(v -> {
                        if (displayTweet.retweeted) {
                            unretweet(displayTweet);
                        } else {
                            retweet(displayTweet);
                        }
                });
            }
        }

        void favorite(Tweet tweet) {
            final TwitterSession session =
TwitterCore.getInstance().getSessionManager().getActiveSession();
            FavoriteService favoriteService =
TwitterCore.getInstance().getApiClient(session).getFavoriteService();
            Call<Tweet> tweetCall = favoriteService.create
                    (tweet.id, false);
            tweetCall.enqueue(new Callback<Tweet>() {
                @Override
                public void success(Result<Tweet> result) {
                    bind(result.data);
                }

                @Override
                public void failure(TwitterException exception) {
                }
            });
        }

        void unfavorite(Tweet tweet) {
            final TwitterSession session =
TwitterCore.getInstance().getSessionManager().getActiveSession();
            FavoriteService favoriteService =
TwitterCore.getInstance().getApiClient(session).getFavoriteService();
            Call<Tweet> tweetCall = favoriteService.destroy
                    (tweet.id, false);
            tweetCall.enqueue(new Callback<Tweet>() {
                @Override
                public void success(Result<Tweet> result) {
                    bind(result.data);
                }

                @Override
                public void failure(TwitterException exception) {
                }
            });
        }

        void retweet(Tweet tweet) {
            final TwitterSession session =
TwitterCore.getInstance().getSessionManager().getActiveSession();
            StatusesService statusesService =
TwitterCore.getInstance().getApiClient(session).getStatusesService();
            Call<Tweet> tweetCall = statusesService.retweet
                    (tweet.id, false);
            tweetCall.enqueue(new Callback<Tweet>() {
                @Override
                public void success(Result<Tweet> result) {
                    bind(result.data);
                }

                @Override
                public void failure(TwitterException exception) {
                }
            });
        }

        void unretweet(Tweet tweet) {
            final TwitterSession session =
TwitterCore.getInstance().getSessionManager().getActiveSession();
            StatusesService statusesService =
TwitterCore.getInstance().getApiClient(session).getStatusesService();
            Call<Tweet> tweetCall = statusesService.unretweet
                    (tweet.id, false);
            tweetCall.enqueue(new Callback<Tweet>() {
                @Override
                public void success(Result<Tweet> result) {
                    bind(result.data);
                }

                @Override
                public void failure(TwitterException exception) {
                }
            });
        }
    }
}


public class TimelineTask extends AsyncTask<Void, Void, Void> {
    TweetAdapter tweetAdapter;
    private MainActivity context;

    public TimelineTask(TweetAdapter adapter, MainActivity context) {
        tweetAdapter = adapter;
        this.context = context;
    }

    public TimelineTask(TweetAdapter adapter) {
        tweetAdapter = adapter;
    }

    @Override
    protected Void doInBackground(Void... temp) {
        final StatusesService statusesService =
TwitterCore.getInstance().getApiClient().getStatusesService();
        Call<List<Tweet>> list = statusesService
                .homeTimeline(800, null, null, false, false, false,
null);
        list.enqueue(new Callback<List<Tweet>>() {
            @Override
            public void success(Result<List<Tweet>> result) {
                tweetAdapter.setItems(result.data);
            }

            @Override
            public void failure(TwitterException exception) {
                Toast.makeText(context, "Отсутствует соединение!",
Toast.LENGTH_LONG).show();
            }
        });
        return null;
    }
}
```