# Android Code

Use of emojis:
https://www.flaticon.com/packs/emotions

## AddEntryFragment

```
package com.ublavins.emotion;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.net.Uri;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;

import android.provider.MediaStore;
import android.util.ArrayMap;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.SearchView;
import android.widget.Toast;


import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.LatLng;
```

```
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.android.material.button.MaterialButton;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.text.DateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Map;

import static android.app.Activity.RESULT_OK;


/**
 * A simple {@link Fragment} subclass.
 * Use the {@link AddEntryFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class AddEntryFragment extends Fragment implements OnMapReadyCallback {

    private static final int REQUEST_LOCATION = 1;
    private static final int REQUEST_IMAGE_CAPTURE = 100;
    private static final int REQUEST_IMAGE_PICK = 101;
    private ImageView happyView, okayView, stressView, sadView, angryView;
    private MapView mapView;
    private GoogleMap googleMap;
    private Geocoder geocoder;
    private Marker marker;
    private SearchView searchView;
```

```
private ImageButton currLocationButton;
private ImageView photoView;
private FusedLocationProviderClient fusedLocationClient;
private MaterialButton uploadPhoto, selectPhoto;
private TextInputLayout thoughtsLayout;
private TextInputEditText thoughtsText;
private String emotion = "";
private CheckBox happyCheck, okayCheck, stressCheck, sadCheck, angryCheck;
private FirebaseFirestore db;
private FirebaseUser mUser;
private FloatingActionButton addEntryButton;
private StorageReference mStorageRef;
private Uri mFilepath;

public AddEntryFragment() {
    // Required empty public constructor
}

public static AddEntryFragment newInstance() {
    AddEntryFragment fragment = new AddEntryFragment();
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mUser = FirebaseAuth.getInstance().getCurrentUser();
    db = FirebaseFirestore.getInstance();
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(getContext());
    mStorageRef = FirebaseStorage.getInstance().getReference();
    geocoder = new Geocoder(getContext(), Locale.ENGLISH);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_add_entry, container, false);
    searchView = view.findViewById(R.id.mapSearch);
    currLocationButton = view.findViewById(R.id.currLocationButton);
    addEntryButton = view.findViewById(R.id.addEntryFloatingButton);
    happyView = view.findViewById(R.id.happyImage);
    okayView = view.findViewById(R.id.okayImage);
    stressView = view.findViewById(R.id.stressImage);
    sadView = view.findViewById(R.id.sadImage);
    angryView = view.findViewById(R.id.angryImage);
```

```
setEmojiClick();
happyCheck = view.findViewById(R.id.happyCheck);
okayCheck = view.findViewById(R.id.okayCheck);
stressCheck = view.findViewById(R.id.stressCheck);
sadCheck = view.findViewById(R.id.sadCheck);
angryCheck = view.findViewById(R.id.angryCheck);
setCheckboxes();
thoughtsLayout = view.findViewById(R.id.thoughtsLayout);
thoughtsText = view.findViewById(R.id.thoughtsText);
selectPhoto = view.findViewById(R.id.selectPhotoButton);
uploadPhoto = view.findViewById(R.id.uploadPhotoButton);
photoView = view.findViewById(R.id.photoView);
mapView = view.findViewById(R.id.mapView);
mapView.onCreate(savedInstanceState);
mapView.onResume();

searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String s) {
        String location = searchView.getQuery().toString();
        List<Address> addresses = null;

        if (location != null && !location.isEmpty()) {
            Geocoder geocoder = new Geocoder(getActivity());
            try {
                addresses = geocoder.getFromLocationName(location, 1);
            } catch (IOException ex) {
                Log.d("Location", ex.toString());
            }
            Address address = addresses.get(0);
            LatLng latLng = new LatLng(address.getLatitude(), address.getLongitude());
            setMarker(new MarkerOptions().position(latLng).title(location));
            googleMap.animateCamera(CameraUpdateFactory
                    .newLatLngZoom(latLng, 18));
        }

        return true;
    }

    @Override
    public boolean onQueryTextChange(String s) {
        return false;
    }
});

currLocationButton.setOnClickListener(
```

```
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    if (ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION)
                            != PackageManager.PERMISSION_GRANTED
                            && ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_COARSE_LOCATION)
                            != PackageManager.PERMISSION_GRANTED) {

                        requestPermissions(new
String[]{android.Manifest.permission.ACCESS_COARSE_LOCATION,
                                        android.Manifest.permission.ACCESS_FINE_LOCATION},
                                REQUEST_LOCATION);

                        return;
                    }
                    getCurrentLocation();
                }
            }
        );

        selectPhoto.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    selectPhoto();
                }
            }
        );

        uploadPhoto.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    if (ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.CAMERA)
                            != PackageManager.PERMISSION_GRANTED) {

                        requestPermissions(new String[]{Manifest.permission.CAMERA},
                                REQUEST_IMAGE_CAPTURE);
                        return;
                    } else {
                        uploadPhoto();
                    }
                }
```

```
            }
        );

        addEntryButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        addEntry();
                    }
                }
        );

        mapView.getMapAsync(this);
        return view;
    }

    @Override
    public void onMapReady(GoogleMap gMap) {
        googleMap = gMap;
        if (ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_COARSE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            googleMap.setMyLocationEnabled(true);
        }
        googleMap.getUiSettings().setMyLocationButtonEnabled(true);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
            Bitmap imageBitmap = (Bitmap) data.getExtras().get("data");
            photoView.getLayoutParams().width = 800;
            photoView.getLayoutParams().height = 800;
            photoView.requestLayout();
            mFilepath = getImageUri(getContext(), imageBitmap);
            photoView.setImageBitmap(imageBitmap);
        } else if (requestCode == REQUEST_IMAGE_PICK && resultCode == RESULT_OK) {
            mFilepath = data.getData();
            Bitmap imageBitmap = null;
            try {
                photoView.getLayoutParams().width = 800;
                photoView.getLayoutParams().height = 800;
                photoView.requestLayout();
```

```
                imageBitmap =
MediaStore.Images.Media.getBitmap(getActivity().getContentResolver(), mFilepath);
                photoView.setImageBitmap(imageBitmap);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    // https://stackoverflow.com/questions/9890757/android-camera-data-intent-returns-null
    private Uri getImageUri(Context applicationContext, Bitmap photo) {
        ByteArrayOutputStream bytes = new ByteArrayOutputStream();
        photo.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
        String path = MediaStore.Images.Media.insertImage(getActivity().getContentResolver(),
photo, "", null);
        return Uri.parse(path);
    }

    @Override
    public void onRequestPermissionsResult(
            int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

        switch (requestCode) {
            case REQUEST_LOCATION: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    getCurrentLocation();
                }
                return;
            }
            case REQUEST_IMAGE_CAPTURE: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    uploadPhoto();
                }
                return;
            }
            case REQUEST_IMAGE_PICK: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    selectPhoto();
                }
                return;
            }

        }
```

```
    }

    private void selectPhoto() {
        Intent getPictureIntent = new Intent(Intent.ACTION_PICK);
        getPictureIntent.setType("image/*");
        startActivityForResult(getPictureIntent, REQUEST_IMAGE_PICK);
    }

    private void uploadPhoto() {
        // https://developer.android.com/training/camera/photobasics
        if (ActivityCompat.checkSelfPermission(getContext(), Manifest.permission.CAMERA)
                == PackageManager.PERMISSION_GRANTED) {
            Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            if (takePictureIntent.resolveActivity(getActivity().getPackageManager()) != null) {
                startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
            }
        }
    }

    private void getCurrentLocation() {
        fusedLocationClient.getLastLocation()
                .addOnSuccessListener(new OnSuccessListener<Location>() {
                    @Override
                    public void onSuccess(Location location) {
                        // Got last known location. In some rare situations this can be null.
                        if (location != null) {
                            LatLng latLng = new LatLng(location.getLatitude(),
location.getLongitude());
                            loadLocation(latLng.latitude, latLng.longitude);
                            setMarker(new MarkerOptions().position(latLng).title("Current Location"));
                            googleMap.animateCamera(CameraUpdateFactory
                                    .newLatLngZoom(latLng, 18));
                        }
                    }
                });
    }

    private void storePhoto(String uri) {
        if (mFilepath != null) {
            StorageReference storageReference = mStorageRef.child(uri);

            storageReference.putFile(mFilepath)
                    .addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                        @Override
                        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
```

```
            }
        });
    }
}

private void setEmojiClick() {
    happyView.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    happyCheck.setChecked(true);
                    okayCheck.setChecked(false);
                    stressCheck.setChecked(false);
                    sadCheck.setChecked(false);
                    angryCheck.setChecked(false);
                }
            }
    );
    okayView.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    okayCheck.setChecked(true);
                    happyCheck.setChecked(false);
                    stressCheck.setChecked(false);
                    sadCheck.setChecked(false);
                    angryCheck.setChecked(false);
                }
            }
    );
    stressView.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    stressCheck.setChecked(true);
                    happyCheck.setChecked(false);
                    okayCheck.setChecked(false);
                    sadCheck.setChecked(false);
                    angryCheck.setChecked(false);
                }
            }
    );
    sadView.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
```

```
                sadCheck.setChecked(true);
                happyCheck.setChecked(false);
                okayCheck.setChecked(false);
                stressCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    );
    angryView.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                angryCheck.setChecked(true);
                happyCheck.setChecked(false);
                okayCheck.setChecked(false);
                sadCheck.setChecked(false);
                stressCheck.setChecked(false);
            }
        }
    );
}

private void setCheckboxes() {
    happyCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (happyCheck.isChecked()) {
                okayCheck.setChecked(false);
                stressCheck.setChecked(false);
                sadCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    });
    okayCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (okayCheck.isChecked()) {
                happyCheck.setChecked(false);
                stressCheck.setChecked(false);
                sadCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    });
    stressCheck.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View view) {
            if (stressCheck.isChecked()) {
                happyCheck.setChecked(false);
                okayCheck.setChecked(false);
                sadCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    });
    sadCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (sadCheck.isChecked()) {
                happyCheck.setChecked(false);
                okayCheck.setChecked(false);
                stressCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    });
    angryCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (angryCheck.isChecked()) {
                happyCheck.setChecked(false);
                okayCheck.setChecked(false);
                sadCheck.setChecked(false);
                stressCheck.setChecked(false);
            }
        }
    });
}

private void loadLocation(double lat, double lon) {
    String address = "";
    try {
        address = geocoder.getFromLocation(lat, lon, 1
        ).get(0).getAddressLine(0);
        searchView.setQuery(address, false);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void addEntry() {
```

```
        if (validateEntry()) {
            Calendar now = Calendar.getInstance();
            String thoughts = thoughtsText.getText().toString();
            String lat = "";
            String lon = "";
            String imgUrl = "";
            String currentDate =
DateFormat.getDateInstance(DateFormat.SHORT).format(now.getTime());
            String currentTime =
DateFormat.getTimeInstance(DateFormat.SHORT).format(now.getTime());
            LatLng latLng;
            String location = "";
            Map<String, Object> entry = new ArrayMap<String, Object>();

            if (marker != null) {
                latLng = marker.getPosition();
                lat = String.valueOf(latLng.latitude);
                lon = String.valueOf(latLng.longitude);
                try {
                    location = geocoder.getFromLocation(
                            latLng.latitude, latLng.longitude, 1).get(0).getAddressLine(0);
                } catch (IOException e) {
                    e.printStackTrace();
                }

            }

            if (mFilepath != null) {
                imgUrl = "entries/" + mUser.getUid() + "/" + new Date().getTime() / 1000 + ".jpg";
                storePhoto(imgUrl);
            }

            entry.put("Emotion", emotion);
            entry.put("Thoughts", thoughts);
            entry.put("Lat", lat);
            entry.put("Lon", lon);
            entry.put("Date", currentDate);
            entry.put("Time", currentTime);
            entry.put("Timestamp", new Date().getTime() / 1000);
            entry.put("Photo", imgUrl);
            entry.put("Location", location);

            db.collection("Entries")
                    .document(mUser.getUid()).collection("entry")
                    .add(entry).addOnCompleteListener(
                    new OnCompleteListener<DocumentReference>() {
```

```
                    @Override
                    public void onComplete(@NonNull Task<DocumentReference> task) {
                        if (task.isSuccessful()) {
                            BottomNavigationView bottomNavBar =
getActivity().findViewById(R.id.mainNavBar);
                            bottomNavBar.setSelectedItemId(R.id.nav_home);
                            makeToast("Added entry to diary");
                            HomeFragment homeFragment = new HomeFragment();

getFragmentManager().beginTransaction().replace(R.id.mainFragmentFrame,
                                    homeFragment).commit();
                        } else {
                            makeToast("Request unsuccessful");
                        }
                    }
                }
            );

        } else {
            makeToast("Please select an emotion");
        }
    }

    private boolean validateEntry() {
        boolean isValid = true;

        if (happyCheck.isChecked()) emotion = "Happy";
        if (okayCheck.isChecked()) emotion = "Okay";
        if (stressCheck.isChecked()) emotion = "Stress";
        if (sadCheck.isChecked()) emotion = "Sad";
        if (angryCheck.isChecked()) emotion = "Angry";

        if (emotion.equals("")) isValid = false;

        return isValid;
    }

    private void setMarker(MarkerOptions markerOptions) {
        if (marker != null) {
            marker.remove();
            marker = googleMap.addMarker(markerOptions);
        } else {
            marker = googleMap.addMarker(markerOptions);
        }
    }
```

```
    private void makeToast(String msg) {
        Toast.makeText(getContext(), msg, Toast.LENGTH_SHORT).show();
    }

}
```

# AuthActivity

```
package com.ublavins.emotion;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class AuthActivity extends AppCompatActivity implements AuthCallback {

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

        FirebaseAuth mAuth = FirebaseAuth.getInstance();
        FirebaseUser mUser = mAuth.getCurrentUser();

        if (mUser != null) {
            startActivity(new Intent(AuthActivity.this, MainActivity.class));
            finish();
        } else {
            loginFragment();
        }
    }

    @Override
    public void loginFragment() {
        LoginFragment loginFrag = new LoginFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.fragmentFrame,
loginFrag).commit();
    }

    @Override
```

```
    public void registerFragment() {
        RegisterFragment registerFrag = new RegisterFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.fragmentFrame,
registerFrag)
                .addToBackStack(null).commit();
    }

    @Override
    public void resetPassFragment() {
        ResetPasswordFragment resetPasswordFragment = new ResetPasswordFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.fragmentFrame,
                resetPasswordFragment).addToBackStack(null).commit();
    }
}
```

# AuthCallback

```
package com.ublavins.emotion;

public interface AuthCallback {
    void loginFragment();
    void registerFragment();
    void resetPassFragment();
}
```

# ChartFragment

```
package com.ublavins.emotion;

import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;

import com.google.android.material.bottomnavigation.BottomNavigationView;


/**
```

```
 * A simple {@link Fragment} subclass.
 * Use the {@link ChartFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class ChartFragment extends Fragment {

    private BottomNavigationView chartNav;

    public ChartFragment() {
        // Required empty public constructor
    }

    public static ChartFragment newInstance(String param1, String param2) {
        ChartFragment fragment = new ChartFragment();
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_chart, container, false);
        chartNav = view.findViewById(R.id.chartNav);

        mapFrame();

        chartNav.setOnNavigationItemSelectedListener(
                new BottomNavigationView.OnNavigationItemSelectedListener() {
                    @Override
                    public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                        switch (menuItem.getItemId()) {
                            case R.id.chart_map:
                                mapFrame();
                                break;
                            case R.id.chart_graph:
                                graphFrame();
                                break;
                        }
                        return true;
                    }
                }
```

```
        );

        return view;
    }

    private void mapFrame() {
        MapChartFragment mapChartFragment = new MapChartFragment();
        getFragmentManager().beginTransaction().replace(R.id.chartFragementFrame,
mapChartFragment)
                .addToBackStack(null).commit();
    }

    private void graphFrame() {
        GraphChartFragment graphChartFragment = new GraphChartFragment();
        getFragmentManager().beginTransaction().replace(R.id.chartFragementFrame,
graphChartFragment)
                .addToBackStack(null).commit();

    }
}
```

# DiaryEntry

```
package com.ublavins.emotion;

public class DiaryEntry {
    private String entryId;
    private String entryEmotion;
    private int entryEmojiIcon;
    private String entryDate;
    private String entryTime;
    private String entryThoughts;
    private long entryTimestamp;
    private String entryLocation;

    public DiaryEntry() {}

    public DiaryEntry(String id, String emotion, int icon, String date, String time, String
thoughts, long timestamp, String location) {
        entryId = id;
        entryEmotion = emotion;
        entryEmojiIcon = icon;
        entryDate = date;
```

```
    entryTime = time;
    entryThoughts = thoughts;
    entryTimestamp = timestamp;
    entryLocation = location;
}

public String getId() {
    return entryId;
}

public void setId(String id) {
    entryId = id;
}

public int getIcon() {
    return entryEmojiIcon;
}

public void setIcon(int icon) {
    entryEmojiIcon = icon;
}

public String getDate() {
    return entryDate;
}

public void setDate(String date) {
    entryDate = date;
}

public String getTime() {
    return entryTime;
}

public void setTime(String time) {
    entryTime = time;
}

public String getThoughts() {
    return entryThoughts;
}

private void setLocation(String location) {
    entryLocation = location;
}
```

```
    public void setThoughts(String thoughts) {
        entryThoughts = thoughts;
    }

    public long getTimestamp() {
        return entryTimestamp;
    }

    public void setEntryTimestamp(long timestamp) {
        entryTimestamp = timestamp;
    }

    public void setEmotion(String emotion) {
        entryEmotion = emotion;
    }

    public String getEmotion() {
        return entryEmotion;
    }

    public String getLocation() {return entryLocation;}
}
```

# DiaryRecyclerAdapter

```
package com.ublavins.emotion;

import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;


import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
```

```
import java.util.ArrayList;

public class DiaryRecyclerAdapter extends
RecyclerView.Adapter<DiaryRecyclerAdapter.DiaryViewHolder> {
   private ArrayList<DiaryEntry> diaryEntryList;

   public DiaryRecyclerAdapter(ArrayList<DiaryEntry> entryList) {
      diaryEntryList = entryList;
   }

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

   @Override
   public void onBindViewHolder(@NonNull DiaryViewHolder holder, int position) {
      DiaryEntry diaryEntry = diaryEntryList.get(position);
      holder.itemView.setTag(diaryEntry.getId());
      holder.entryImage.setImageResource(diaryEntry.getIcon());
      holder.entryDate.setText(diaryEntry.getDate());
      holder.entryTime.setText(diaryEntry.getTime());
      holder.entryEmotion.setText(diaryEntry.getEmotion());
      holder.entryThoughts.setText(diaryEntry.getThoughts());
      holder.entryLocation.setText(getLocation(diaryEntry.getLocation()));

      switch(diaryEntry.getEmotion()) {
         case "Okay":
            holder.entryImage.setColorFilter(Color.parseColor("#999900"));
            break;
         case "Happy":
            holder.entryImage.setColorFilter(Color.parseColor("#ff669900"));
            break;
         case "Sad":
            holder.entryImage.setColorFilter(Color.parseColor("#ff0099cc"));
            break;
         case "Angry":
            holder.entryImage.setColorFilter(Color.parseColor("#ffcc0000"));
            break;
```

```
        }


    }

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

    public static class DiaryViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener{
        public ImageView entryImage;
        public TextView entryDate;
        public TextView entryTime;
        public TextView entryEmotion;
        public TextView entryThoughts;
        public TextView entryLocation;

        public DiaryViewHolder(@NonNull View itemView) {
            super(itemView);
            entryImage = itemView.findViewById(R.id.imageView);
            entryDate = itemView.findViewById(R.id.dateTextView);
            entryTime = itemView.findViewById(R.id.timeTextView);
            entryEmotion = itemView.findViewById(R.id.emotionTextView);
            entryThoughts = itemView.findViewById(R.id.thoughtsView);
            entryLocation = itemView.findViewById(R.id.locationText);
            itemView.setOnClickListener(this);
        }

        @Override
        public void onClick(View view) {
            FirebaseFirestore.getInstance().collection("Entries")
                    .document(FirebaseAuth.getInstance().getCurrentUser().getUid())
                    .collection("entry").document(view.getTag().toString()).get()
                    .addOnSuccessListener(
                    new OnSuccessListener<DocumentSnapshot>() {
                        @Override
                        public void onSuccess(DocumentSnapshot documentSnapshot) {
                            AppCompatActivity activity = (AppCompatActivity)view.getContext();
                            EntryFragment entryFragment = new EntryFragment(documentSnapshot);
                            activity.getSupportFragmentManager().beginTransaction()
                                    .replace(R.id.mainFragmentFrame, entryFragment)
                                    .addToBackStack(null).commit();
                        }
                    }
```

```
            );
        }

    }

    private String getLocation(String location) {
        String loc = "N/A";
        if (!location.isEmpty()) {
            loc =  location;
        }
        return loc;
    }

}
```

# EmotionMarker

```
package com.ublavins.emotion;

import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class EmotionMarker {

    private String markerLat;
    private String markerLon;
    private String markerEmotion;
    private String markerThoughts;
    private MarkerOptions markerOptions;

    public EmotionMarker() {
        markerOptions = new MarkerOptions();
    }

    public void setLat(String lat) {
        markerLat = lat;
    }

    public void setLon(String lon) {
        markerLon = lon;
    }

    public void setEmotion(String emotion) {
        markerEmotion = emotion;
```

```
    }

    public void setThoughts(String thoughts) {
        markerThoughts = thoughts;
    }

    public void setPosition() {
        if (!markerLat.isEmpty() && !markerLon.isEmpty()) {
            LatLng latLng = new LatLng(Double.parseDouble(markerLat),
                    Double.parseDouble(markerLon));
            markerOptions.position(latLng);
        }
    }

    public void setTitle() {
        if (markerEmotion != null && !markerEmotion.isEmpty()) {
            markerOptions.title(markerEmotion);
        }
    }

    public void setSnippet() {
        if (markerThoughts != null && !markerThoughts.isEmpty()) {
            markerOptions.snippet(markerThoughts);
        }
    }

    public MarkerOptions getMarker() {
        if (markerLat != null && !markerLat.isEmpty() &&
                markerLon != null && !markerLon.isEmpty()) {
            setPosition();
            setTitle();
            setSnippet();
        }
        return markerOptions;
    }

}
```

# EntryFragment

```
package com.ublavins.emotion;

import android.Manifest;
import android.content.pm.PackageManager;
```

```
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.net.Uri;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.SearchView;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.material.button.MaterialButton;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.squareup.picasso.Picasso;

import java.io.IOException;
import java.util.List;
import java.util.Locale;


/**
 * A simple {@link Fragment} subclass.
```

```
 * Use the {@link EntryFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class EntryFragment extends Fragment implements OnMapReadyCallback {

    private static final int REQUEST_LOCATION = 1;
    private DocumentSnapshot entry;
    private MaterialButton updateEntry, deleteEntry;
    private ImageView happyView, okayView, stressView, sadView, angryView;
    private CheckBox happyCheck, okayCheck, stressCheck, sadCheck, angryCheck;
    private TextInputEditText thoughtsText;
    private GoogleMap googleMap;
    private Marker marker;
    private MapView map;
    private SearchView searchView;
    private ImageButton currLocationButton;
    private FirebaseFirestore db;
    private FirebaseUser mUser;
    private FusedLocationProviderClient fusedLocationClient;
    private String emotionStr = "";
    private ImageView photoView;
    private Geocoder geocoder;

    public EntryFragment(DocumentSnapshot documentSnapshot) {
        entry = documentSnapshot;
    }

    public static EntryFragment newInstance(DocumentSnapshot documentSnapshot) {
        EntryFragment fragment = new EntryFragment(documentSnapshot);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mUser = FirebaseAuth.getInstance().getCurrentUser();
        db = FirebaseFirestore.getInstance();
        fusedLocationClient = LocationServices.getFusedLocationProviderClient(getContext());
        geocoder = new Geocoder(getContext(), Locale.ENGLISH);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_entry, container, false);
```

```
emotionStr = entry.get("Emotion").toString();
map = view.findViewById(R.id.mapView);
map.onCreate(savedInstanceState);
map.onResume();
map.getMapAsync(this);
happyView = view.findViewById(R.id.happyImage);
okayView = view.findViewById(R.id.okayImage);
stressView = view.findViewById(R.id.stressImage);
sadView = view.findViewById(R.id.sadImage);
angryView = view.findViewById(R.id.angryImage);
setEmojiClick();
happyCheck = view.findViewById(R.id.happyCheck);
okayCheck = view.findViewById(R.id.okayCheck);
stressCheck = view.findViewById(R.id.stressCheck);
sadCheck = view.findViewById(R.id.sadCheck);
angryCheck = view.findViewById(R.id.angryCheck);
thoughtsText = view.findViewById(R.id.thoughtsText);
searchView = view.findViewById(R.id.mapSearch);
currLocationButton = view.findViewById(R.id.currLocationButton);
updateEntry = view.findViewById(R.id.updateEntryButton);
deleteEntry = view.findViewById(R.id.deleteEntryButton);
photoView = view.findViewById(R.id.photoView);
if (!entry.getString("Lat").isEmpty() && !entry.getString("Lon").isEmpty()) {
   loadLocation(
         Double.parseDouble(entry.getString("Lat")),
         Double.parseDouble(entry.getString("Lon"))
   );
}
setPhoto();
setCheck();
setCheckboxes();
thoughtsText.setText(entry.get("Thoughts").toString());
setMap();


updateEntry.setOnClickListener(
      new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            updateEntry();
         }
      }
);

deleteEntry.setOnClickListener(
      new View.OnClickListener() {
```

```
            @Override
            public void onClick(View view) {
                delete();
            }
        }
    );


    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String s) {
            String location = searchView.getQuery().toString();
            List<Address> addresses = null;

            if (location != null && !location.isEmpty()) {
                Geocoder geocoder = new Geocoder(getActivity());
                try {
                    addresses = geocoder.getFromLocationName(location, 1);
                } catch (IOException ex) {
                    Log.d("Location", ex.toString());
                }
                Address address = addresses.get(0);
                LatLng latLng = new LatLng(address.getLatitude(), address.getLongitude());
                setMarker(new MarkerOptions().position(latLng).title(location));
                googleMap.animateCamera(CameraUpdateFactory
                        .newLatLngZoom(latLng, 18));
            }

            return true;
        }

        @Override
        public boolean onQueryTextChange(String s) {
            return false;
        }
    });

    currLocationButton.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    if (ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION)
                            != PackageManager.PERMISSION_GRANTED
                            && ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_COARSE_LOCATION)
```

```
                    != PackageManager.PERMISSION_GRANTED) {

                requestPermissions(new
String[]{android.Manifest.permission.ACCESS_COARSE_LOCATION,
                        android.Manifest.permission.ACCESS_FINE_LOCATION},
                    REQUEST_LOCATION);

                return;
            }
            getCurrentLocation();
        }
      }
    );
    return view;
  }

  @Override
  public void onMapReady(GoogleMap gMap) {
    googleMap = gMap;
    if (ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED
        && ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_COARSE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
      googleMap.setMyLocationEnabled(true);
    }
    googleMap.getUiSettings().setMyLocationButtonEnabled(true);
  }

  @Override
  public void onRequestPermissionsResult(
      int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

    switch (requestCode) {
      case REQUEST_LOCATION: {
        if (grantResults.length > 0
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
          getCurrentLocation();
        }
        return;
      }
    }
  }

  private void loadLocation(double lat, double lon) {
```

```
        String address = "";
        try {
            address = geocoder.getFromLocation(lat, lon, 1
            ).get(0).getAddressLine(0);
            searchView.setQuery(address, false);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void updateEntry() {
        if (validateEntry()) {
            String thoughts = thoughtsText.getText().toString();
            String lat = "";
            String lon = "";
            LatLng latLng;
            String location = "";

            if (marker != null) {
                latLng = marker.getPosition();
                lat = String.valueOf(latLng.latitude);
                lon = String.valueOf(latLng.longitude);
                try {
                    location = geocoder.getFromLocation(latLng.latitude, latLng.longitude, 1
                    ).get(0).getAddressLine(0);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            db.collection("Entries").document(mUser.getUid())
                    .collection("entry").document(entry.getId()).update(
                        "Emotion", emotionStr,
                        "Thoughts", thoughts,
                        "Lat", lat,
                        "Lon", lon,
                    "Location", location
            );
            HomeFragment homeFragment = new HomeFragment();
            getFragmentManager().beginTransaction().replace(R.id.mainFragmentFrame,
homeFragment)
                    .commit();
        }
    }

    private boolean validateEntry() {
```

```
        boolean isValid = true;
        if (happyCheck.isChecked()) emotionStr = "Happy";
        if (okayCheck.isChecked()) emotionStr = "Okay";
        if (stressCheck.isChecked()) emotionStr = "Stress";
        if (sadCheck.isChecked()) emotionStr = "Sad";
        if (angryCheck.isChecked()) emotionStr = "Angry";

        if (emotionStr.equals("")) isValid = false;

        return isValid;
    }

    private void delete() {

        if (!entry.getString("Photo").isEmpty()) {
            StorageReference storageReference =
FirebaseStorage.getInstance().getReference();
            StorageReference photo = storageReference.child(entry.getString("Photo"));
            photo.delete().addOnSuccessListener(
                    new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            deleteEntry();
                        }
                    }
            );
        } else {
            deleteEntry();
        }
    }

    private void deleteEntry() {
        db.collection("Entries").document(mUser.getUid()).collection("entry")
                .document(entry.getId()).delete().addOnSuccessListener(
                new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        HomeFragment homeFragment = new HomeFragment();

getFragmentManager().beginTransaction().replace(R.id.mainFragmentFrame,
homeFragment)
                                .commit();
                    }
                }
        );
    }
```

```
private void setEmojiClick() {
    happyView.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    happyCheck.setChecked(true);
                    okayCheck.setChecked(false);
                    stressCheck.setChecked(false);
                    sadCheck.setChecked(false);
                    angryCheck.setChecked(false);
                }
            }
    );
    okayView.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    okayCheck.setChecked(true);
                    happyCheck.setChecked(false);
                    stressCheck.setChecked(false);
                    sadCheck.setChecked(false);
                    angryCheck.setChecked(false);
                }
            }
    );
    stressView.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    stressCheck.setChecked(true);
                    happyCheck.setChecked(false);
                    okayCheck.setChecked(false);
                    sadCheck.setChecked(false);
                    angryCheck.setChecked(false);
                }
            }
    );
    sadView.setOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    sadCheck.setChecked(true);
                    happyCheck.setChecked(false);
                    okayCheck.setChecked(false);
                    stressCheck.setChecked(false);
```

```
                angryCheck.setChecked(false);
            }
        }
    );
    angryView.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                angryCheck.setChecked(true);
                happyCheck.setChecked(false);
                okayCheck.setChecked(false);
                sadCheck.setChecked(false);
                stressCheck.setChecked(false);
            }
        }
    );
}

private void setCheckboxes() {
    happyCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (happyCheck.isChecked()) {
                okayCheck.setChecked(false);
                stressCheck.setChecked(false);
                sadCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    });
    okayCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (okayCheck.isChecked()) {
                happyCheck.setChecked(false);
                stressCheck.setChecked(false);
                sadCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    });
    stressCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (stressCheck.isChecked()) {
                happyCheck.setChecked(false);
```

```
                okayCheck.setChecked(false);
                sadCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    });
    sadCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (sadCheck.isChecked()) {
                happyCheck.setChecked(false);
                okayCheck.setChecked(false);
                stressCheck.setChecked(false);
                angryCheck.setChecked(false);
            }
        }
    });
    angryCheck.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (angryCheck.isChecked()) {
                happyCheck.setChecked(false);
                okayCheck.setChecked(false);
                sadCheck.setChecked(false);
                stressCheck.setChecked(false);
            }
        }
    });
}

private void setCheck() {
    String emotion = entry.get("Emotion").toString();
    switch (emotion) {
        case "Happy":
            happyCheck.setChecked(true);
            break;
        case "Okay":
            okayCheck.setChecked(true);
            break;
        case "Stress":
            stressCheck.setChecked(true);
            break;
        case "Sad":
            sadCheck.setChecked(true);
            break;
        case "Angry":
```

```
            angryCheck.setChecked(true);
            break;
        }
    }

    private void setMap() {
        db.collection("Entries").document(mUser.getUid())
                .collection("entry").document(entry.getId()).get().addOnSuccessListener(
                new OnSuccessListener<DocumentSnapshot>() {
                    @Override
                    public void onSuccess(DocumentSnapshot documentSnapshot) {
                        String lat = documentSnapshot.get("Lat").toString();
                        String lon = documentSnapshot.get("Lon").toString();

                        if (!lat.equals("") && !lon.equals("")) {
                            LatLng latLng = new LatLng(Double.parseDouble(lat),
Double.parseDouble(lon));
                            marker = googleMap.addMarker(new
MarkerOptions().position(latLng).title("Location"));

googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15));
                        }
                    }
                }
        );
    }

    private void getCurrentLocation() {
        fusedLocationClient.getLastLocation()
                .addOnSuccessListener(new OnSuccessListener<Location>() {
                    @Override
                    public void onSuccess(Location location) {
                        // Got last known location. In some rare situations this can be null.
                        if (location != null) {
                            LatLng latLng = new LatLng(location.getLatitude(),
location.getLongitude());
                            loadLocation(latLng.latitude, latLng.longitude);
                            setMarker(new MarkerOptions().position(latLng).title("Current Location"));
                            googleMap.animateCamera(CameraUpdateFactory
                                    .newLatLngZoom(latLng, 18));
                        }
                    }
                });
    }

    private void setMarker(MarkerOptions markerOptions) {
```

```
        if (marker != null) {
            marker.remove();
            marker = googleMap.addMarker(markerOptions);
        } else {
            marker = googleMap.addMarker(markerOptions);
        }
    }

    private void setPhoto() {
        if (entry.get("Photo") != null) {
            if (!entry.getString("Photo").isEmpty()) {
                StorageReference storageReference =
FirebaseStorage.getInstance().getReference();
                StorageReference photo = storageReference.child(entry.getString("Photo"));
                photo.getDownloadUrl().addOnSuccessListener(
                        new OnSuccessListener<Uri>() {
                            @Override
                            public void onSuccess(Uri uri) {
                                photoView.getLayoutParams().height = 800;
                                photoView.getLayoutParams().width = 800;
                                photoView.requestLayout();
                                Picasso.get().load(uri).into(photoView);
                            }
                        }
                );
            }
        }
    }
}
```

# GraphChartFragment

```
package com.ublavins.emotion;

import android.graphics.Color;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.util.ArrayMap;
import android.util.Log;
import android.view.LayoutInflater;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;

import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.charts.PieChart;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;
import com.github.mikephil.charting.data.BarEntry;
import com.github.mikephil.charting.data.PieData;
import com.github.mikephil.charting.data.PieDataSet;
import com.github.mikephil.charting.data.PieEntry;
import com.github.mikephil.charting.formatter.IndexAxisValueFormatter;
import com.github.mikephil.charting.utils.ColorTemplate;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
import com.tiper.MaterialSpinner;

import java.util.ArrayList;
import java.util.Map;


/**
 * A simple {@link Fragment} subclass.
 * Use the {@link GraphChartFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class GraphChartFragment extends Fragment {

    private PieChart pieChart;
    private BarChart barChart;
    private MaterialSpinner menuSpinner;
    private FirebaseFirestore db;
    private static final String[] CHARTS = {"Pie", "Bar"};
    private int count;
    private Map<String, Integer> emotion = new ArrayMap<>();

    public GraphChartFragment() {
        // Required empty public constructor
    }
```

```
public static GraphChartFragment newInstance(String param1, String param2) {
    GraphChartFragment fragment = new GraphChartFragment();
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    db = FirebaseFirestore.getInstance();
    count = 0;
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_graph_chart, container, false);
    menuSpinner = view.findViewById(R.id.chartType);
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(getContext(),
            android.R.layout.simple_dropdown_item_1line, CHARTS);
    menuSpinner.setAdapter(adapter);
    menuSpinner.setSelection(0);
    pieChart = view.findViewById(R.id.pieChart);
    barChart = view.findViewById(R.id.barChart);
    loadPieChart();
    loadBarChart();
    db.collection("Entries").document(
            FirebaseAuth.getInstance().getCurrentUser().getUid())
            .collection("entry").get().addOnCompleteListener(
            new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            if (!emotion.containsKey(document.get("Emotion").toString())) {
                                emotion.put(document.get("Emotion").toString(), 1);
                                count++;
                            } else {
                                emotion.put(
                                        document.get("Emotion").toString(),
                                        emotion.get(document.get("Emotion").toString()) + 1
                                );
                                count++;
                            }
                        }
```

```
                    if (count > 0) {
                        ArrayList<String> keys = new ArrayList<>();
                        ArrayList<PieEntry> vals = new ArrayList<>();
                        ArrayList<BarEntry> bars = new ArrayList<>();

                        for (String key : emotion.keySet()) { keys.add(key);}
                        for (int i = 0; i < keys.size(); i++) {
                            vals.add(new PieEntry((float)emotion.get(keys.get(i))/count,
keys.get(i)));
                            bars.add(new BarEntry(i, emotion.get(keys.get(i))));
                        }

                        Log.d("Test", keys.toString());

                        PieDataSet dataSet = new PieDataSet(vals, "Emotions");
                        BarDataSet barDataSet = new BarDataSet(bars, "Emotions");
                        barDataSet.setColors(ColorTemplate.COLORFUL_COLORS);
                        BarData bData = new BarData(barDataSet);
                        barChart.setData(bData);
                        XAxis xAxis = barChart.getXAxis();
                        xAxis.setValueFormatter(new IndexAxisValueFormatter(keys));
                        xAxis.setPosition(XAxis.XAxisPosition.TOP);
                        xAxis.setDrawGridLines(false);
                        xAxis.setLabelCount(keys.size());


                        dataSet.setSliceSpace(3f);
                        dataSet.setSelectionShift(5f);
                        dataSet.setColors(ColorTemplate.COLORFUL_COLORS);

                        PieData pData = new PieData(dataSet);
                        pData.setValueTextSize(10f);
                        pData.setValueTextColor(Color.YELLOW);

                        pieChart.setData(pData);
                        pieChart.setDrawHoleEnabled(false);
                        pieChart.invalidate();
                        barChart.invalidate();
                    } else {
                        pieChart.setNoDataText("No data available");
                    }

                }
            }
        }
```

```
        );

        menuSpinner.setOnItemSelectedListener(
                new MaterialSpinner.OnItemSelectedListener() {
                    @Override
                    public void onItemSelected(MaterialSpinner materialSpinner, View view, int i,
long l) {
                        if (materialSpinner.getSelectedItem() == "Pie") {
                            barChart.setVisibility(View.INVISIBLE);
                            pieChart.animateXY(1500, 1500);
                            pieChart.setVisibility(View.VISIBLE);
                        } else {
                            pieChart.setVisibility(View.INVISIBLE);
                            barChart.animateY(1500);
                            barChart.setVisibility(View.VISIBLE);
                        }
                    }

                    @Override
                    public void onNothingSelected(MaterialSpinner materialSpinner) {
                    }
                }
        );
        return view;
    }

    private void loadPieChart() {
        pieChart.setUsePercentValues(true);
        pieChart.getDescription().setEnabled(false);
        pieChart.setExtraOffsets(5, 10, 5, 5);
        pieChart.setDragDecelerationFrictionCoef(0.95f);
        pieChart.setVisibility(View.VISIBLE);
        pieChart.animateXY(1500, 1500);
        pieChart.setNoDataText("");
        Legend l = pieChart.getLegend();
        l.setVerticalAlignment(Legend.LegendVerticalAlignment.TOP);
        l.setHorizontalAlignment(Legend.LegendHorizontalAlignment.CENTER);
    }

    private void loadBarChart() {
        barChart.setVisibility(View.INVISIBLE);
        barChart.getDescription().setEnabled(false);
        barChart.setNoDataText("");
    }
}
```

# HomeFragment

```
package com.ublavins.emotion;

import android.os.Build;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
import com.tiper.MaterialSpinner;

import java.util.ArrayList;
import java.util.List;


public class HomeFragment extends Fragment {

    private FloatingActionButton addEntry;
    private RecyclerView diaryRecyclerView;
    private RecyclerView.Adapter diaryAdapter;
    private RecyclerView.LayoutManager diaryLayoutManager;
```

```
private FirebaseFirestore db;
private List<DiaryEntry> entries = new ArrayList<>();
private MaterialSpinner emotionSpinner;
private ProgressBar homeProgress;
private ImageView noEntryImage;
private TextView noEntryText;
private static final String[] EMOTIONS = {"All", "Happy", "Okay", "Stress", "Sad", "Angry"};
private boolean loaded = true;

public HomeFragment() {
   // Required empty public constructor
}

public static HomeFragment newInstance() {
   HomeFragment fragment = new HomeFragment();
   return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   db = FirebaseFirestore.getInstance();
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
   // Inflate the layout for this fragment
   View view = inflater.inflate(R.layout.fragment_home, container, false);
   final ArrayList<DiaryEntry> entryList = new ArrayList<>();
   diaryRecyclerView = view.findViewById(R.id.diaryRecyclerView);
   diaryRecyclerView.setHasFixedSize(true);
   homeProgress = view.findViewById(R.id.homeProgress);
   noEntryImage = view.findViewById(R.id.noEntryImage);
   noEntryText = view.findViewById(R.id.noEntryText);
   emotionSpinner = view.findViewById(R.id.emotionSpinner);
   emotionSpinner.setHintAnimationEnabled(false);
   ArrayAdapter<String> adapter = new ArrayAdapter<String>(getContext(),
         android.R.layout.simple_dropdown_item_1line, EMOTIONS);
   emotionSpinner.setAdapter(adapter);
   emotionSpinner.setSelection(0);
   diaryLayoutManager = new LinearLayoutManager(getContext());
   diaryAdapter = new DiaryRecyclerAdapter(entryList);

   db.collection("Entries").document(
         FirebaseAuth.getInstance().getCurrentUser().getUid())
```

```
        .collection("entry").orderBy("Timestamp",
        Query.Direction.DESCENDING).get().addOnCompleteListener(
        new OnCompleteListener<QuerySnapshot>() {
            @RequiresApi(api = Build.VERSION_CODES.N)
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful() && loaded) {
                    if (!task.getResult().isEmpty()) {
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            DiaryEntry entry = new DiaryEntry(
                                    document.getId(),
                                    document.getString("Emotion"),
                                    getIcon(document.get("Emotion").toString()),
                                    document.get("Date").toString(),
                                    document.get("Time").toString(),
                                    document.get("Thoughts").toString(),
                                    document.getLong("Timestamp"),
                                    document.getString("Location")
                            );
                            entryList.add(entry);
                            entries.add(entry);
                        }
                        diaryRecyclerView.setLayoutManager(diaryLayoutManager);
                        diaryRecyclerView.setAdapter(diaryAdapter);
                        homeProgress.invalidate();
                        homeProgress.setVisibility(View.INVISIBLE);
                        emotionSpinner.setVisibility(View.VISIBLE);
                        loaded = false;
                    } else {
                        homeProgress.invalidate();
                        homeProgress.setVisibility(View.INVISIBLE);
                        noEntryImage.setVisibility(View.VISIBLE);
                        noEntryText.setVisibility(View.VISIBLE);
                    }
                }
                if (!loaded) {
                    homeProgress.invalidate();
                    homeProgress.setVisibility(View.INVISIBLE);
                }
            }
        }
);

emotionSpinner.setOnItemSelectedListener(
        new MaterialSpinner.OnItemSelectedListener() {
            @Override
```

```
                public void onItemSelected(MaterialSpinner materialSpinner, View view, int i,
long l) {
                    filterEmotion(materialSpinner.getSelectedItem().toString());
                }

                @Override
                public void onNothingSelected(MaterialSpinner materialSpinner) {
                }
            }
        );

        addEntry = (FloatingActionButton)view.findViewById(R.id.addEntryFloatingButton);

        addEntry.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        BottomNavigationView bottomNavigationView = getActivity().
                                findViewById(R.id.mainNavBar);
                        bottomNavigationView.setSelectedItemId(R.id.nav_add_entry);
                    }
                }
        );
        return view;
    }

    private void filterEmotion(String emotion) {
        int icon = getIcon(emotion);
        List<DiaryEntry> filterEntries = new ArrayList<>();
        diaryLayoutManager = new LinearLayoutManager(getContext());
        diaryAdapter = new DiaryRecyclerAdapter((ArrayList<DiaryEntry>) filterEntries);
        if (emotion.equals("All")) {
            for (DiaryEntry entry : entries) filterEntries.add(entry);
        } else {
            for (DiaryEntry entry : entries) {
                if (entry.getIcon() == icon) {
                    filterEntries.add(entry);
                }
            }
        }
        diaryRecyclerView.setLayoutManager(diaryLayoutManager);
        diaryRecyclerView.setAdapter(diaryAdapter);
    }

    private int getIcon(String emotion) {
        int icon = 0;
```

```
        switch (emotion) {
            case "Happy":
                icon = R.drawable.ic_happy;
                break;
            case "Okay":
                icon = R.drawable.ic_okay;
                break;
            case "Stress":
                icon = R.drawable.ic_stress;
                break;
            case "Sad":
                icon = R.drawable.ic_sad;
                break;
            case "Angry":
                icon = R.drawable.ic_angry;
                break;
        }
        return icon;
    }

}
```

# LoginFragment

```
package com.ublavins.emotion;

import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.util.Patterns;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.button.MaterialButton;
import com.google.android.material.textfield.TextInputEditText;
```

```
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;



/**
 * A simple {@link Fragment} subclass.
 * Use the {@link LoginFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class LoginFragment extends Fragment {

    private MaterialButton loginButton;
    private MaterialButton registerButton;
    private MaterialButton forgotPassButton;
    private TextInputLayout emailInput;
    private TextInputLayout passwordInput;
    private AuthCallback callback;
    private FirebaseAuth mAuth;
    private TextInputEditText email, password;
    private ProgressBar progressBar;

    public LoginFragment() {
        // Required empty public constructor
    }

    public static LoginFragment newInstance() {
        LoginFragment fragment = new LoginFragment();
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        callback = (AuthCallback) getActivity();
        mAuth = FirebaseAuth.getInstance();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_login, container, false);
        progressBar = view.findViewById(R.id.homeProgress);
        email = view.findViewById(R.id.emailLogin);
        password = view.findViewById(R.id.passwordLogin);
```

```
        emailInput = view.findViewById(R.id.emailInputLayout);
        passwordInput = view.findViewById(R.id.passwordInputLayout);
        loginButton = view.findViewById(R.id.loginButton);
        loginButton.setOnClickListener(
                new View.OnClickListener() {
                    public void onClick (View v) {
                        login(); }
                    }
                );
        registerButton = view.findViewById(R.id.registerButton);
        registerButton.setOnClickListener(
                new View.OnClickListener() {
                    public void onClick (View v) {
                        callback.registerFragment(); }
                }
        );
        forgotPassButton = view.findViewById(R.id.forgetPassButton);
        forgotPassButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        callback.resetPassFragment();
                    }
                }
        );
        return view;
    }

    public void login() {
        if (validateLogin()) {
            progressBar.setVisibility(View.VISIBLE);
            String emailText = email.getText().toString();
            String passText = password.getText().toString();
            mAuth.signInWithEmailAndPassword(emailText, passText)
                    .addOnCompleteListener(getActivity(), new OnCompleteListener<AuthResult>()
{
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                progressBar.invalidate();
                                progressBar.setVisibility(View.INVISIBLE);
                                // makeToast("User exists");
                                Intent intent =  new Intent(getActivity(), MainActivity.class);
                                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                startActivity(intent);
                                getActivity().finish();
```

```
                    } else {
                        progressBar.setVisibility(View.INVISIBLE);
                        makeToast("User does not exist");
                    }
                }
            });
        }
    }

    private boolean validateLogin() {
        boolean isValid = true;
        String emailText = email.getText().toString();
        String passText = password.getText().toString();


        if (emailText.isEmpty() || !Patterns.EMAIL_ADDRESS.matcher(emailText).matches()) {
            emailInput.setError("Enter a valid email address");
            isValid = false;
        } else {
            emailInput.setError(null);
        }

        if (passText.isEmpty()) {
            passwordInput.setError("Enter a password");
            isValid = false;
        } else {
            passwordInput.setError(null);
        }
        return isValid;
    }

    private void makeToast(String msg) {
        Toast.makeText(getContext(), msg, Toast.LENGTH_SHORT).show();
    }
}
```

# MainActivity

```
package com.ublavins.emotion;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;

public class MainActivity extends AppCompatActivity implements MainCallback {

    private BottomNavigationView bottomNavBar;

    private FirebaseAuth mAuth;
    private FirebaseUser mUser;
    private FirebaseFirestore db;
    private DocumentSnapshot snap;

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

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.logoutButtonMain:
                logout();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    private void logout() {
        mAuth.signOut();
        Intent intent = new Intent(this, AuthActivity.class);
```

```
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(intent);
        this.finish();
    }

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


        bottomNavBar = findViewById(R.id.mainNavBar);

        mAuth = FirebaseAuth.getInstance();
        mUser = mAuth.getCurrentUser();
        db = FirebaseFirestore.getInstance();

        final DocumentReference docRef = db.collection("Users").document(mUser.getUid());
        docRef.get().addOnCompleteListener(
                new OnCompleteListener<DocumentSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                        if (task.isSuccessful()) {
                            DocumentSnapshot docSnap = task.getResult();
                            if (docSnap.exists()) {
                                snap = docSnap;
                            }
                        }
                    }
                }
        );

        docRef.addSnapshotListener(
                new EventListener<DocumentSnapshot>() {
                    @Override
                    public void onEvent(@Nullable DocumentSnapshot documentSnapshot,
@Nullable FirebaseFirestoreException e) {
                        if (documentSnapshot != null && documentSnapshot.exists()) {
                            snap = documentSnapshot;
                        }
                    }
                }
        );

        homeFragment();
```

```
        bottomNavBar.setOnNavigationItemSelectedListener(
                new BottomNavigationView.OnNavigationItemSelectedListener() {
                    @Override
                    public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                        switch (menuItem.getItemId()) {
                            case R.id.nav_home:
                                homeFragment();
                                break;
                            case R.id.nav_add_entry:
                                addEntryFragment();
                                break;
                            case R.id.nav_charts:
                                chartFragment();
                                break;
                            case R.id.nav_profile:
                                profileFragment();
                                break;
                        }
                        return true;
                    }
                }
        );
    }

    @Override
    public void homeFragment() {
        HomeFragment homeFrag = new HomeFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.mainFragmentFrame,
homeFrag)
                .addToBackStack(null).commit();
    }

    @Override
    public void addEntryFragment() {
        AddEntryFragment addEntryFragment = new AddEntryFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.mainFragmentFrame,
                addEntryFragment).addToBackStack(null).commit();
    }

    @Override
    public void chartFragment() {
        ChartFragment chartFrag = new ChartFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.mainFragmentFrame,
chartFrag)
                .addToBackStack(null).commit();
    }
```

```
    @Override
    public void profileFragment() {
        ProfileFragment profileFrag = new ProfileFragment(snap);
        getSupportFragmentManager().beginTransaction().replace(R.id.mainFragmentFrame,
profileFrag)
                .addToBackStack(null).commit();
    }

}
```

# MainCallback

```
package com.ublavins.emotion;

public interface MainCallback {
    void homeFragment();
    void addEntryFragment();
    void chartFragment();
    void profileFragment();
}
```

# MapChartFragment

```
package com.ublavins.emotion;

import android.Manifest;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.location.Location;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.google.android.gms.location.FusedLocationProviderClient;
```

```
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.BitmapDescriptor;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.Arrays;
import java.util.List;


/**
 * A simple {@link Fragment} subclass.
 * Use the {@link MapChartFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class MapChartFragment extends Fragment implements OnMapReadyCallback,
        ActivityCompat.OnRequestPermissionsResultCallback {

    private static final int REQUEST_LOCATION = 1;
    private GoogleMap googleMap;
    private MapView mapView;
    private FusedLocationProviderClient fusedLocationClient;
    private FirebaseFirestore db;
    private List<String> EMOTIONS = Arrays.asList("Happy", "Okay", "Stress", "Sad",
"Angry");

    public MapChartFragment() {
        // Required empty public constructor
    }

    public static MapChartFragment newInstance() {
        MapChartFragment fragment = new MapChartFragment();
        return fragment;
    }
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(getContext());
    db = FirebaseFirestore.getInstance();
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_map_chart, container, false);
    mapView = view.findViewById(R.id.mapEmotions);
    mapView.onCreate(savedInstanceState);
    mapView.onResume();
    loadMap();
    mapView.getMapAsync(this);
    db.collection("Entries").document(
            FirebaseAuth.getInstance().getCurrentUser().getUid())
            .collection("entry").get().addOnCompleteListener(
            new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            if (!document.get("Lat").toString().equals("") &&
                            !document.get("Lon").toString().equals("")) {
                                if (EMOTIONS.contains(document.getString("Emotion"))) {
                                    setMarker(getMarker(
                                            document.get("Lat").toString(),
                                            document.get("Lon").toString(),
                                            document.get("Emotion").toString(),
                                            document.getString("Thoughts")
                                    ));
                                }
                            }
                        }
                    }
                }
            }
    );
    return view;
}

@Override
public void onMapReady(GoogleMap gMap) {
```

```
        googleMap = gMap;
        // Check if ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION
permissions have been set
        if (ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_COARSE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {
            googleMap.setMyLocationEnabled(true);
        }
        googleMap.getUiSettings().setMyLocationButtonEnabled(true);
    }

    private void loadMap() {
        // Check ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION
permissions
        // If not set request permissions then request permissions
        if (ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(getContext(),
Manifest.permission.ACCESS_COARSE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {

            requestPermissions(new
String[]{android.Manifest.permission.ACCESS_COARSE_LOCATION,
                            android.Manifest.permission.ACCESS_FINE_LOCATION},
                    REQUEST_LOCATION);

            return;
        }
        fusedLocationClient.getLastLocation()
                .addOnSuccessListener(new OnSuccessListener<Location>() {
                    @Override
                    public void onSuccess(Location location) {
                        if (location != null) {
                            LatLng latLng = new LatLng(location.getLatitude(),
                                    location.getLongitude());
                            googleMap.animateCamera(CameraUpdateFactory
                                    .newLatLngZoom(latLng, 12));
                        }
                    }
                });
    }
```

```
@Override
public void onRequestPermissionsResult(
        int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

    switch (requestCode) {
        case REQUEST_LOCATION: {
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                loadMap();
            }
            return;
        }

    }
}

private MarkerOptions getMarker(String lat, String lon, String emotion, String thoughts) {
    EmotionMarker emotionMarker = new EmotionMarker();
    emotionMarker.setLat(lat);
    emotionMarker.setLon(lon);
    emotionMarker.setEmotion(emotion);
    emotionMarker.setThoughts(thoughts);
    MarkerOptions marker = emotionMarker.getMarker();
    marker.icon(getIcon(emotion));
    return marker;
}

private BitmapDescriptor getIcon(String emotion) {
    Bitmap icon = BitmapFactory.decodeResource(getResources(), getBitmap(emotion));
    Bitmap bitmap = Bitmap.createScaledBitmap(icon, 80, 80, false);
    return BitmapDescriptorFactory.fromBitmap(bitmap);
}

private int getBitmap(String emotion) {
    int icon = 0;
    switch (emotion) {
        case "Happy":
            icon = R.drawable.happy;
            break;
        case "Okay":
            icon = R.drawable.okay;
            break;
        case "Stress":
            icon = R.drawable.stress;
            break;
        case "Sad":
```

```
                icon = R.drawable.sad;
                break;
            case "Angry":
                icon = R.drawable.angry;
                break;
        }
        return icon;
    }

    private void setMarker(MarkerOptions markerOptions) {
        googleMap.addMarker(markerOptions);
    }
}
```

# ProfileFragment

```
package com.ublavins.emotion;

import android.app.DatePickerDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;

import android.util.Patterns;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.DatePicker;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.button.MaterialButton;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.Calendar;
```

```
/**
 * A simple {@link Fragment} subclass.
 * Use the {@link ProfileFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class ProfileFragment extends Fragment {

    private MaterialButton editProfile, saveProfile, deleteAccount;
    private TextInputEditText fname, lname, email, dob, gender;
    private TextInputLayout fnameInput, lnameInput, emailInput, dobInput, genderInput;
    private FirebaseAuth mAuth;
    private DocumentSnapshot docSnap;
    private String fnameStr, lnameStr, emailStr, dobStr, genderStr;

    public ProfileFragment(DocumentSnapshot snap) {
        // Required empty public constructor
        docSnap = snap;
    }

    public static ProfileFragment newInstance(DocumentSnapshot snap) {
        ProfileFragment fragment = new ProfileFragment(snap);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mAuth = FirebaseAuth.getInstance();
        fnameStr = docSnap.get("FirstName").toString();
        lnameStr = docSnap.get("LastName").toString();
        emailStr = docSnap.get("Email").toString();
        dobStr = docSnap.get("Dob").toString();
        genderStr = docSnap.get("Gender").toString();
    }

    // Will need to think of caching firebase store
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_profile, container, false);

        fname = view.findViewById(R.id.fnameText);
        lname = view.findViewById(R.id.lnameText);
        email = view.findViewById(R.id.emailText);
```

```
        dob = view.findViewById(R.id.dobText);
        gender = view.findViewById(R.id.genderText);
        fnameInput = view.findViewById(R.id.fnameInputLayout);
        fnameInput.setHintAnimationEnabled(false);
        lnameInput = view.findViewById(R.id.lnameInputLayout);
        lnameInput.setHintAnimationEnabled(false);
        emailInput = view.findViewById(R.id.emailInputLayout);
        emailInput.setHintAnimationEnabled(false);
        dobInput = view.findViewById(R.id.dobInputLayout);
        dobInput.setHintAnimationEnabled(false);
        genderInput = view.findViewById(R.id.genderInputLayout);
        genderInput.setHintAnimationEnabled(false);
        editProfile = view.findViewById(R.id.editButton);
        saveProfile = view.findViewById(R.id.saveButton);
        deleteAccount = view.findViewById(R.id.deleteButton);
        fname.setText(fnameStr);
        lname.setText(lnameStr);
        email.setText(emailStr);
        dob.setText(dobStr);
        gender.setText(genderStr);

        deleteAccount.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity(),
R.style.AlertDialogStyle);
                        builder.setTitle("Delete account?");
                        builder.setPositiveButton("DELETE", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {
                                String uuid = mAuth.getUid();
                                FirebaseFirestore.getInstance().collection("Users")
                                        .document(mAuth.getUid()).delete();
                                FirebaseFirestore.getInstance().collection("Entries")
                                        .document(mAuth.getUid()).delete();
                                mAuth.getCurrentUser().delete();
                                mAuth.signOut();
                                Intent intent = new Intent(getActivity(), AuthActivity.class);
                                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                startActivity(intent);
                                getActivity().finish();
                            }
                        });
                        builder.setNegativeButton("CANCEL", new DialogInterface.OnClickListener()
{
```

```
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {

                    }
                });
                builder.show();
            }
        }
    );

    dob.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (dob.isFocusable()){
                    getBirthDate();
                }
            }
        }
    );

    editProfile.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (editProfile.getVisibility() == View.VISIBLE) {
                    editProfile.setVisibility(View.INVISIBLE);
                    saveProfile.setVisibility(View.VISIBLE);
                    fname.setFocusable(true);
                    fname.setFocusableInTouchMode(true);
                    lname.setFocusable(true);
                    lname.setFocusableInTouchMode(true);
                    email.setFocusable(true);
                    email.setFocusableInTouchMode(true);
                    dob.setFocusableInTouchMode(true);
                    gender.setFocusable(true);
                    gender.setFocusableInTouchMode(true);
                    Toast.makeText(getContext(), "Edit profile fields",
Toast.LENGTH_SHORT).show();
                }
            }
        }
    );

    saveProfile.setOnClickListener(
        new View.OnClickListener() {
```

```
            @Override
            public void onClick(View view) {
                if (saveProfile.getVisibility() == View.VISIBLE) {
                    if (validateUpdate()) {
                        if (!email.getText().toString().equals(mAuth.getCurrentUser().getEmail()))
{
                            Toast.makeText(getContext(), "TEST", Toast.LENGTH_SHORT);

FirebaseAuth.getInstance().getCurrentUser().updateEmail(email.getText().toString())
                                .addOnCompleteListener(new OnCompleteListener<Void>() {
                                    @Override
                                    public void onComplete(@NonNull Task<Void> task) {
                                        if (task.isSuccessful()) {
                                            Toast.makeText(getContext(), "Updated Profile",
Toast.LENGTH_SHORT).show();
                                        }
                                    }
                                });
                        }

                        FirebaseFirestore.getInstance().collection("Users")
                                .document(mAuth.getUid()).update(
                                        "FirstName", fname.getText().toString(),
                                "LastName", lname.getText().toString(),
                                "Email", email.getText().toString(),
                                "Dob", dob.getText().toString()
                        );

                        Toast.makeText(getContext(), "Profile Updated",
Toast.LENGTH_SHORT).show();

                        fname.setFocusable(false);
                        fname.setFocusableInTouchMode(false);
                        lname.setFocusable(false);
                        lname.setFocusableInTouchMode(false);
                        email.setFocusable(false);
                        email.setFocusableInTouchMode(false);
                        dob.setFocusable(false);
                        dob.setFocusableInTouchMode(false);
                        gender.setFocusable(false);
                        gender.setFocusableInTouchMode(false);
                        saveProfile.setVisibility(View.INVISIBLE);
                        editProfile.setVisibility(View.VISIBLE);
                    }
                }
            }
```

```
            }
        );

        return view;
    }

    private void getBirthDate() {
        final Calendar calendar = Calendar.getInstance();
        final int birthYear = calendar.get(Calendar.YEAR);
        final int birthMonth = calendar.get(Calendar.MONTH);
        final int birthDay = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePicker = new DatePickerDialog(getContext(),
                new DatePickerDialog.OnDateSetListener() {
                    @Override
                    public void onDateSet(DatePicker datePicker, int year, int month, int day) {
                        dob.setText(day + "/" + (month + 1) + "/" + year);
                    }
                }, birthYear, birthMonth, birthDay);
        datePicker.show();
    }

    private boolean validateUpdate() {
        boolean isValid = true;
        String fnameText = fname.getText().toString();
        String lnameText = lname.getText().toString();
        String emailText = email.getText().toString();
        String dateText = dob.getText().toString();

        if (fnameText.isEmpty()) {
            fnameInput.setError("Field must not be empty");
            isValid = false;
        } else {
            fnameInput.setError(null);
        }

        if (lnameText.isEmpty()) {
            lnameInput.setError("Field must not be empty");
            isValid = false;
        } else {
            lnameInput.setError(null);
        }

        if (emailText.isEmpty() || !Patterns.EMAIL_ADDRESS.matcher(emailText).matches()) {
            emailInput.setError("Enter a valid email address");
            isValid = false;
```

```
        } else {
            emailInput.setError(null);
        }

        if (dateText.isEmpty()) {
            dobInput.setError("Enter a date");
            isValid = false;
        } else {
            dobInput.setError(null);
        }
        return isValid;
    }

}
```

# RegisterFragment

```
package com.ublavins.emotion;

import android.app.DatePickerDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;

import android.util.Patterns;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;

import android.widget.CheckBox;
import android.widget.DatePicker;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.button.MaterialButton;
import com.google.android.material.textfield.TextInputEditText;
```

```
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.tiper.MaterialSpinner;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;


/**
 * A simple {@link Fragment} subclass.
 * Use the {@link RegisterFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class RegisterFragment extends Fragment {

    private AuthCallback callback;
    private TextInputLayout fnameInput, lnameInput, emailInput, passwordInput,
        dateInput;
    private TextInputEditText fname, lname, email, password, date;
    private TextView tosText;
    private MaterialButton signUp;
    private CheckBox tos;
    private static final String[] GENDERS = {"Male", "Female", "Other"};
    private MaterialSpinner gender;
    private FirebaseAuth mAuth;
    private ProgressBar progressBar;

    public RegisterFragment() {
        // Required empty public constructor
    }

    public static RegisterFragment newInstance() {
        RegisterFragment fragment = new RegisterFragment();
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        callback = (AuthCallback)getActivity();
        mAuth = FirebaseAuth.getInstance();
    }
```

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_register, container, false);
        progressBar = view.findViewById(R.id.homeProgress);
        fnameInput = view.findViewById(R.id.fnameInputLayout);
        lnameInput = view.findViewById(R.id.lnameInputLayout);
        emailInput = view.findViewById(R.id.emailInputLayout);
        passwordInput = view.findViewById(R.id.passwordInputLayout);
        dateInput = view.findViewById(R.id.dobInputLayout);
        tos = view.findViewById(R.id.tosCheck);
        tosText = view.findViewById(R.id.tosText);
        fname = view.findViewById(R.id.fnameRegister);
        lname = view.findViewById(R.id.lnameRegister);
        email = view.findViewById(R.id.emailRegister);
        password = view.findViewById(R.id.passwordRegister);
        date = view.findViewById(R.id.dobRegister);
        gender = view.findViewById(R.id.genderRegister);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getContext(),
                android.R.layout.simple_dropdown_item_1line, GENDERS);
        gender.setAdapter(adapter);
        signUp = view.findViewById(R.id.signUpButton);

        tosText.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity(),
R.style.AlertDialogStyle);
                        builder.setTitle("Agree to terms and conditions");
                        builder.setMessage("By clicking agree, I hereby accept the storage of
personal data and possible use of data for research purposes.");
                        builder.setPositiveButton("AGREE", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {
                                tos.setChecked(true);
                            }
                        });
                        builder.setNegativeButton("CANCEL", new DialogInterface.OnClickListener()
{
                            @Override
                            public void onClick(DialogInterface dialogInterface, int i) {

                            }
                        });
```

```
                builder.show();
            }
        }
    );

    date.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                getBirthDate();
            }
        }
    );

    signUp.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                register();
            }
        }
    );

    return view;
}

public void register() {
    if (validateRegister()) {
        progressBar.setVisibility(View.VISIBLE);
        String emailText = email.getText().toString();
        String passText = password.getText().toString();
        final Map<String, Object> user = new HashMap<>();
        user.put("FirstName", fname.getText().toString());
        user.put("LastName", lname.getText().toString());
        user.put("Email", emailText);
        user.put("Dob", date.getText().toString());
        user.put("Gender", gender.getSelectedItem().toString());
        mAuth.createUserWithEmailAndPassword(user.get("Email").toString(), passText)
            .addOnCompleteListener(getActivity(), new OnCompleteListener<AuthResult>()
{
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        FirebaseFirestore.getInstance().collection("Users")
                                .document(FirebaseAuth.getInstance().getCurrentUser().getUid())
```

```
                            .set(user).addOnCompleteListener(new
OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()) {
                                progressBar.invalidate();
                                progressBar.setVisibility(View.INVISIBLE);
                                Intent intent = new Intent(getActivity(), MainActivity.class);
                                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                startActivity(intent);
                                getActivity().finish();
                            } else {
                                progressBar.setVisibility(View.INVISIBLE);
                                makeToast("Error creating account");
                            }

                        }
                    });
                } else {
                    progressBar.setVisibility(View.INVISIBLE);
                    makeToast("Account already exists");
                }
            }
        });
    }
}

private void getBirthDate() {
    final Calendar calendar = Calendar.getInstance();
    final int birthYear = calendar.get(Calendar.YEAR);
    final int birthMonth = calendar.get(Calendar.MONTH);
    final int birthDay = calendar.get(Calendar.DAY_OF_MONTH);

    DatePickerDialog datePicker = new DatePickerDialog(getContext(),
            new DatePickerDialog.OnDateSetListener() {
                @Override
                public void onDateSet(DatePicker datePicker, int year, int month, int day) {
                    date.setText(day + "/" + (month + 1) + "/" + year);
                }
            }, birthYear, birthMonth, birthDay);
    datePicker.show();
}

private boolean validateRegister() {
    boolean isValid = true;
    String fnameText = fname.getText().toString();
```

```
String lnameText = lname.getText().toString();
String emailText = email.getText().toString();
String passText = password.getText().toString();
String dateText = date.getText().toString();
String genderText = "";

try {
    genderText = gender.getSelectedItem().toString();
} catch (NullPointerException ex) {}

ValidateRegistration validate = new ValidateRegistration(
        fnameText, lnameText, emailText, passText, dateText, genderText
);

if (!validate.validateFirstName().getCheck()) {
    fnameInput.setError(validate.validateFirstName().getMessage());
    isValid = validate.validateFirstName().getCheck();
} else {
    fnameInput.setError(null);
}

if (!validate.validateLastName().getCheck()) {
    lnameInput.setError(validate.validateLastName().getMessage());
    isValid = validate.validateLastName().getCheck();
} else {
    lnameInput.setError(null);
}

if (!validate.validateEmail().getCheck()) {
    emailInput.setError(validate.validateEmail().getMessage());
    isValid = validate.validateEmail().getCheck();
} else {
    emailInput.setError(null);
}

if (!validate.validatePassword().getCheck()) {
    passwordInput.setError(validate.validatePassword().getMessage());
    isValid = validate.validatePassword().getCheck();
} else {
    passwordInput.setError(null);
}

if (!validate.validateGender().getCheck()) {
    gender.setError(validate.validateGender().getMessage());
    isValid = validate.validateGender().getCheck();
} else {
```

```
            gender.setError(null);
        }

        if (dateText.isEmpty()) {
            dateInput.setError("Enter a date");
            isValid = false;
        } else {
            dateInput.setError(null);
        }



        if (!tos.isChecked()) {
            isValid = false;
            makeToast("Check terms and conditions");
        }

        return isValid;
    }

    private void makeToast(String msg) {
        Toast.makeText(getContext(), msg, Toast.LENGTH_SHORT).show();
    }
}
```

# ResetPasswordFragment

```
package com.ublavins.emotion;

import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import android.util.Patterns;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.button.MaterialButton;
```

```
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.FirebaseAuth;


/**
 * A simple {@link Fragment} subclass.
 * Use the {@link ResetPasswordFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class ResetPasswordFragment extends Fragment {

    private MaterialButton resetPassButton;
    private TextInputEditText emailReset;
    private TextInputLayout emailResetLayout;
    private FirebaseAuth mAuth;

    public ResetPasswordFragment() {
        // Required empty public constructor
    }

    public static ResetPasswordFragment newInstance() {
        ResetPasswordFragment fragment = new ResetPasswordFragment();
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mAuth = FirebaseAuth.getInstance();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_reset_password, container, false);
        resetPassButton = view.findViewById(R.id.resetPassButton);
        emailReset = view.findViewById(R.id.emailReset);
        emailResetLayout = view.findViewById(R.id.emailInputLayout);

        resetPassButton.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        if (validateEmail()) {
```

```
                    mAuth.sendPasswordResetEmail(emailReset.getText().toString())
                            .addOnCompleteListener(
                            new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {
                                    if (task.isSuccessful()) {
                                        Toast.makeText(getContext(),
                                                "Password reset email sent",
                                                Toast.LENGTH_SHORT).show();
                                        getFragmentManager().popBackStack();
                                    } else {
                                        Toast.makeText(getContext(),
                                                "Email is not registered",
                                                Toast.LENGTH_SHORT).show();
                                    }
                                }
                            }
                    );
                }
            }
        }
    );

    return view;
}

private boolean validateEmail() {
    boolean isValid = true;
    String emailText = emailReset.getText().toString();

    if (emailText.isEmpty()) {
        emailResetLayout.setError("Please enter an email address");
        isValid = false;
    } else if (!Patterns.EMAIL_ADDRESS.matcher(emailText).matches()) {
        emailResetLayout.setError("Please enter a valid email address");
        isValid = false;
    } else {
        emailResetLayout.setError(null);
    }

    return isValid;
}

}
```

# ValidateRegistration

```
package com.ublavins.emotion;

import android.util.Patterns;


public class ValidateRegistration {

    public class ValidationMessage {

        private boolean check;
        private String message;

        public ValidationMessage(boolean bool, String msg) {
            check = bool;
            message = msg;
        }

        public void setCheck(boolean bool) {
            check = bool;
        }

        public void setMessage(String msg) {
            message = msg;
        }

        public boolean getCheck() {
            return check;
        }

        public String getMessage() {
            return message;
        }

    }

    private String firstNameVal;
    private String lastNameVal;
    private String emailVal;
    private String passwordVal;
    private String dobVal;
    private String genderVal;
```

```
public ValidateRegistration(
        String fName,
        String lName,
        String email,
        String pass,
        String dob,
        String gender
) {
    firstNameVal = fName;
    lastNameVal = lName;
    emailVal = email;
    passwordVal = pass;
    dobVal = dob;
    genderVal = gender;
}

public void setFirstNameVal(String fName) {
    firstNameVal = fName;
}

public void setLastNameVal(String lName) {
    lastNameVal = lName;
}

public void setEmailVal(String email) {
    emailVal = email;
}

public void setPasswordVal(String pass) {
    passwordVal = pass;
}

public void setDobVal(String dob) {
    dobVal = dob;
}

public void setGenderVal(String gender) {
    genderVal = gender;
}

public String getFirstNameVal() {
    return firstNameVal;
}

public String getLastNameVal() {
    return lastNameVal;
```

```
}

public String getEmailVal() {
    return emailVal;
}

public String getPasswordVal() {
    return passwordVal;
}

public String getDobVal() {
    return dobVal;
}

public String getGenderVal() {
    return genderVal;
}

public ValidationMessage validateFirstName() {
    ValidationMessage validationMessage = new ValidationMessage(true, "");
    if (firstNameVal.isEmpty()) {
        validationMessage.setCheck(false);
        validationMessage.setMessage("Field must not be empty");
    } else if (!firstNameVal.matches("^[a-zA-Z]+$")) {
        validationMessage.setCheck(false);
        validationMessage.setMessage("First name must contain letters");
    }
    return validationMessage;
}

public ValidationMessage validateLastName() {
    ValidationMessage validationMessage = new ValidationMessage(true, "");
    if (lastNameVal.isEmpty()) {
        validationMessage.setCheck(false);
        validationMessage.setMessage("Field must not be empty");
    } else if (!lastNameVal.matches("^[a-zA-Z]+$")) {
        validationMessage.setCheck(false);
        validationMessage.setMessage("Last name must contain letters");
    }
    return validationMessage;
}

public ValidationMessage validateEmail() {
    ValidationMessage validationMessage = new ValidationMessage(true, "");
    if (emailVal.isEmpty() || !Patterns.EMAIL_ADDRESS.matcher(emailVal).matches()) {
        validationMessage.setCheck(false);
```

```
            validationMessage.setMessage("Enter a valid email");
        }
        return validationMessage;
    }

    public ValidationMessage validatePassword() {
        ValidationMessage validationMessage = new ValidationMessage(true, "");
        if (passwordVal.isEmpty()) {
            validationMessage.setCheck(false);
            validationMessage.setMessage("Field must not be empty");
        } else if (passwordVal.length() < 8) {
            validationMessage.setCheck(false);
            validationMessage.setMessage("Password must have a minimum length of 8
characters");
        }
        return validationMessage;
    }

    public ValidationMessage validateDob() {
        ValidationMessage validationMessage = new ValidationMessage(true, "");
        if (dobVal.isEmpty()) {
            validationMessage.setCheck(false);
            validationMessage.setMessage("Enter a date");
        }
        return validationMessage;
    }

    public ValidationMessage validateGender() {
        ValidationMessage validationMessage = new ValidationMessage(true, "");
        if (genderVal.isEmpty()) {
            validationMessage.setCheck(false);
            validationMessage.setMessage("Select a gender");
        }
        return validationMessage;
    }

}
```

# EmotionMarkerTest

```
package com.ublavins.emotion;

import com.google.android.gms.maps.model.MarkerOptions;
```

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class EmotionMarkerTest {

    private EmotionMarker emotionMarker;

    @Before
    public void setUp() {
        emotionMarker = new EmotionMarker();
    }

    @After
    public void tearDown() {
        emotionMarker = null;
    }

    @Test
    public void emptyMarkerWithoutOptions() {
        MarkerOptions sut = new EmotionMarker().getMarker();
        assertNull(sut.getPosition());
    }

    @Test
    public void setLatLonPosition() {
        String fakeLat = "5.024012";
        String fakeLon = "-1.429342";
        emotionMarker.setLat(fakeLat);
        emotionMarker.setLon(fakeLon);
        MarkerOptions sut = emotionMarker.getMarker();
        assertEquals(fakeLat, String.valueOf(sut.getPosition().latitude));
        assertEquals(fakeLon, String.valueOf(sut.getPosition().longitude));
    }

    @Test
    public void emptyLatLonNullPosition() {
        String fakeLat = "";
        String fakeLon = "";
        emotionMarker.setLat(fakeLat);
        emotionMarker.setLon(fakeLon);
        MarkerOptions sut = emotionMarker.getMarker();
        assertNull(sut.getPosition());
    }
```

```
@Test
public void setEmotionTitle() {
    String fakeLat = "5.024012";
    String fakeLon = "-1.429342";
    String fakeEmotion = "TestEmotion";
    emotionMarker.setLat(fakeLat);
    emotionMarker.setLon(fakeLon);
    emotionMarker.setEmotion(fakeEmotion);
    MarkerOptions sut = emotionMarker.getMarker();
    assertEquals(fakeEmotion, sut.getTitle());
}

@Test
public void emptyEmotionNullTitle() {
    String fakeLat = "5.024012";
    String fakeLon = "-1.429342";
    String fakeEmotion = "";
    emotionMarker.setLat(fakeLat);
    emotionMarker.setLon(fakeLon);
    emotionMarker.setEmotion(fakeEmotion);
    MarkerOptions sut = emotionMarker.getMarker();
    assertNull(sut.getTitle());
}

@Test
public void setThoughtsSnippet() {
    String fakeLat = "5.024012";
    String fakeLon = "-1.429342";
    String fakeThoughts = "This is a test";
    emotionMarker.setLat(fakeLat);
    emotionMarker.setLon(fakeLon);
    emotionMarker.setThoughts(fakeThoughts);
    MarkerOptions sut = emotionMarker.getMarker();
    assertEquals(fakeThoughts, sut.getSnippet());
}

@Test
public void emptyThoughtsNullSnippet() {
    String fakeLat = "5.024012";
    String fakeLon = "-1.429342";
    String fakeThoughts = "";
    emotionMarker.setLat(fakeLat);
    emotionMarker.setLon(fakeLon);
    emotionMarker.setThoughts(fakeThoughts);
    MarkerOptions sut = emotionMarker.getMarker();
```

```
        assertNull(sut.getSnippet());
    }

    @Test
    public void emptyLocationEntryNotMarked() {
        String fakeLat = "";
        String fakeLon = "";
        String fakeEmotion = "Test Emotion";
        String fakeThoughts = "Test Thoughts";
        emotionMarker.setLat(fakeLat);
        emotionMarker.setLon(fakeLon);
        emotionMarker.setEmotion(fakeEmotion);
        emotionMarker.setThoughts(fakeThoughts);
        MarkerOptions sut = emotionMarker.getMarker();
        assertNull(sut.getPosition());
        assertNull(sut.getTitle());
        assertNull(sut.getSnippet());
    }

}
```

# ValidateRegistrationTest

```
package com.ublavins.emotion;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class ValidateRegistrationTest {

    private ValidateRegistration validateRegistration;
    private String mockFirstName;
    private String mockLastName;
    private String mockEmail;
    private String mockPassword;
    private String mockDob;
    private String mockGender;

    @Before
    public void setUp() {
        mockFirstName = "";
```

```
    mockLastName = "";
    mockEmail = "";
    mockPassword = "";
    mockDob = "";
    mockGender = "";
    validateRegistration = new ValidateRegistration(
            mockFirstName, mockLastName, mockEmail,
            mockPassword, mockDob, mockGender);
}

@After
public void tearDown() {
    validateRegistration = null;
}

@Test
public void setGetFirstName() {
    String expected = "test";
    validateRegistration.setFirstNameVal(expected);
    String sut = validateRegistration.getFirstNameVal();
    assertEquals(expected, sut);
}

@Test
public void setGetLastName() {
    String expected = "test";
    validateRegistration.setLastNameVal(expected);
    String sut = validateRegistration.getLastNameVal();
    assertEquals(expected, sut);
}

@Test
public void setGetEmail() {
    String expected = "test";
    validateRegistration.setEmailVal(expected);
    String sut = validateRegistration.getEmailVal();
    assertEquals(expected, sut);
}

@Test
public void setGetPassword() {
    String expected = "test";
    validateRegistration.setPasswordVal(expected);
    String sut = validateRegistration.getPasswordVal();
    assertEquals(expected, sut);
}
```

```
@Test
public void setGetDob() {
    String expected = "test";
    validateRegistration.setDobVal(expected);
    String sut = validateRegistration.getDobVal();
    assertEquals(expected, sut);
}

@Test
public void setGetGender() {
    String expected = "test";
    validateRegistration.setGenderVal(expected);
    String sut = validateRegistration.getGenderVal();
    assertEquals(expected, sut);
}

@Test
public void validateEmptyFirstName() {
    String fakeFirstName = "";
    validateRegistration.setFirstNameVal(fakeFirstName);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateFirstName();
    assertFalse(sut.getCheck());
    assertEquals("Field must not be empty", sut.getMessage());
}

@Test
public void validateInvalidFirstName() {
    String fakeFirstName = "12345";
    validateRegistration.setFirstNameVal(fakeFirstName);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateFirstName();
    assertFalse(sut.getCheck());
    assertEquals("First name must contain letters", sut.getMessage());
}

@Test
public void validateValidFirstName() {
    String fakeFirstName = "Test";
    validateRegistration.setFirstNameVal(fakeFirstName);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateFirstName();
    assertTrue(sut.getCheck());
    assertEquals("", sut.getMessage());
}

@Test
public void validateEmptyLastName() {
```

```
    String fakeLastName = "";
    validateRegistration.setLastNameVal(fakeLastName);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateLastName();
    assertFalse(sut.getCheck());
    assertEquals("Field must not be empty", sut.getMessage());
}

@Test
public void validateInvalidLastName() {
    String fakeLastName = "12345";
    validateRegistration.setLastNameVal(fakeLastName);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateLastName();
    assertFalse(sut.getCheck());
    assertEquals("Last name must contain letters", sut.getMessage());
}

@Test
public void validateValidLastName() {
    String fakeLastName = "Test";
    validateRegistration.setLastNameVal(fakeLastName);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateLastName();
    assertTrue(sut.getCheck());
    assertEquals("", sut.getMessage());
}

@Test
public void validateInvalidEmail() {
    String fakeEmail = "";
    validateRegistration.setEmailVal(fakeEmail);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateEmail();
    assertFalse(sut.getCheck());
    assertEquals("Enter a valid email", sut.getMessage());
}

@Test
public void validateEmptyPassword() {
    String fakePassword = "";
    validateRegistration.setPasswordVal(fakePassword);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validatePassword();
    assertFalse(sut.getCheck());
    assertEquals("Field must not be empty", sut.getMessage());
}

@Test
public void validateInvalidPassword() {
    String fakePassword = "test";
```

```
    validateRegistration.setPasswordVal(fakePassword);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validatePassword();
    assertFalse(sut.getCheck());
    assertEquals("Password must have a minimum length of 8 characters",
            sut.getMessage());
}

@Test
public void validateValidPassword() {
    String fakePassword = "Password";
    validateRegistration.setPasswordVal(fakePassword);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validatePassword();
    assertTrue(sut.getCheck());
    assertEquals("", sut.getMessage());
}

@Test
public void validateEmptyDob() {
    String fakeDob = "";
    validateRegistration.setDobVal(fakeDob);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateDob();
    assertFalse(sut.getCheck());
    assertEquals("Enter a date", sut.getMessage());
}

@Test
public void validateValidDob() {
    String fakeDob = "01/01/01";
    validateRegistration.setDobVal(fakeDob);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateDob();
    assertTrue(sut.getCheck());
    assertEquals("", sut.getMessage());
}

@Test
public void validateEmptyGender() {
    String fakeGender = "";
    validateRegistration.setGenderVal(fakeGender);
    ValidateRegistration.ValidationMessage sut = validateRegistration.validateGender();
    assertFalse(sut.getCheck());
    assertEquals("Select a gender", sut.getMessage());
}

@Test
public void validateValidGender() {
    String fakeGender = "Male";
```

```
        validateRegistration.setGenderVal(fakeGender);
        ValidateRegistration.ValidationMessage sut = validateRegistration.validateGender();
        assertTrue(sut.getCheck());
        assertEquals("", sut.getMessage());
    }

}
```