Allen Lu

Work Samples

Summary: Glo – Android Studio Mobile App

Android Studio application that tracks a user's emotional state history by allowing them to create moods with 5 different states, specify the reason and social situation, upload a jpeg image that is attached to the mood, specify the mood instance location on a map, and shared their most recent mood instance with their friends.

Github Link:
https://github.com/ApluUalberta/GroupProject1/tree/master/Code/app/src/main/java/com/example/myapplication

Code that pulls user's data from a Firestore database:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_mood_history);
    historyActivity = this;
    filterPressed = getResources().getColor(android.R.color.darker_gray);
    db = FirebaseFirestore.getInstance();
    users = db.collection("Users");
    users.addSnapshotListener(new EventListener<QuerySnapshot>() {
        @Override
        public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
            Log.d(TAG, "Something changed");
            if (queryDocumentSnapshots != null) {
                for (DocumentChange doc : queryDocumentSnapshots.getDocumentChanges()) {
                    if (doc.getDocument().get("Username").equals(user.getName())) {
                        user = doc.getDocument().get("Participant", Participant.class);
                        user.setUID(doc.getDocument().getId());
                        moodArrayList = user.getMoodHistory();
                        moodArrayAdapter.clear();
                        moodArrayAdapter.addAll(user.getMoodHistory());
                    }
                }
            }
        }
    });
    Intent intent = getIntent();
    user = (Participant) intent.getSerializableExtra("User");
    moodArrayList = user.getMoodHistory();
    moodArrayAdapter = new CustomList(this,moodArrayList,user,null);

    moodHistory = findViewById(R.id.mood_history);
    moodHistory.setAdapter(moodArrayAdapter);
    filterScroll = findViewById(R.id.FilterScroll);

    greatFilter = findViewById(R.id.GreatFilterButton);
    goodFilter = findViewById(R.id.GoodFilterButton);
    neutralFilter = findViewById(R.id.NeutralFilterButton);
    badFilter = findViewById(R.id.BadFilterButton);
    worstFilter = findViewById(R.id.WorstFilterButton);
    buttonBackground = worstFilter.getBackground();

}
```

The code that updates the user data in the Firestore database after mood history edit, delete, adding, etc:

Allen Lu

Code that allows the user to edit, add, and delete the google map location that a mood instance was in:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_view_map);
    Intent intent = getIntent();
    latLng = new LatLng(intent.getDoubleExtra("Lat",0),intent.getDoubleExtra("Long",0));

    SupportMapFragment mapFrag = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.Viewmap);
    mapFrag.getMapAsync(this);
}


@Override
public void onMapReady(GoogleMap googleMap) {
    getLocationPermission();
    if (mLocationPermissionGranted) {
        googleMap.moveCamera( CameraUpdateFactory.newLatLngZoom(latLng , 16.0f) );

        googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
        googleMap.getUiSettings().setMapToolbarEnabled(false);
        googleMap.getUiSettings().setZoomControlsEnabled(true);
        googleMap.addMarker(new MarkerOptions().position(latLng)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE)));

    }
}

private void getLocationPermission() {
    if (ContextCompat.checkSelfPermission(this.getApplicationContext(), Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PER
        mLocationPermissionGranted = true;
    } else {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, PERMISSIONS_REQUEST_ACCESS_FINE_
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    mLocationPermissionGranted = false;
    if (requestCode == PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            mLocationPermissionGranted = true;
        } else {
            Toast.makeText(this, "App must have permissions for your location if you want to use the map", Toast.LENGTH_LONG).show();
            setResult(RESULT_CANCELED);
            finish();
        }
    }
```

Allen Lu

Summary: Crime Statistics – Embedded SQLite3 Database Program

A command-line interface that allows a user to generate bar/pie charts and maps of Edmonton's crime statistics using embedded SQLite3 queries. Users execute queries that are translated to plain English in order to generate common database information graphics using third-party libraries (Pandas for executing queries, Folium to map query data, and MatPlotLib to graph query data).

Github Link: https://github.com/ApluUalberta/Crime-Statistics-Database-Program/blob/master/src/a4_specific_utils.py

Code that creates new map data upon user activating query that creates map data:

```
65  def create_new_edmonton_map():
66      """
67      Creates a new map object centered on Edmonton
68      """
69      return folium.Map(location=_FOLIUM_EDMONTON_MAP_COORDS, zoom_start=12)
70
71
72  def add_markers_to_map(map, markers, avg_val):
73      """ Plots the given markers on a map
74
75      :param map: The map to add markers to
76      :markers: A list of FolioMarkers that describe info for each maker to place on the map
77      :avg_val: The avg value to use for the markers passed in. Any marker vals less/greater than the average will scaled.
78      """
79
80      for marker in markers:
81          calc_radius = (marker.val / avg_val) * _FOLIUM_AVG_RAD_SIZE + _FOLIUM_ADDITIONAL_RAD_SIZE
82
83          folium.Circle(
84              location=marker.coords,
85              popup=marker.str,
86              radius=calc_radius,
87              color=marker.colour,
88              fill=True,
89              fill_color=marker.colour
90          ).add_to(map)
91
92
93  def write_map_to_file(map, q_num):
94      """
95      Writes the given map to file named appropiately by the question number
96      """
97      map_file_name = generate_filename_for_question_file(q_num, "html")
98      map.save(map_file_name)
99
100     print("Wrote \"{}\" to disk.".format(map_file_name))
101
102
103 def get_num_neighborhoods():
104     return _NUM_NEIGHBORHOODS
105
```

Allen Lu

Code that uses embedded SQLite3 query to generate map data for the top neighbourhoods for a specific crime:

```
def menu_map_of_top_neighborhoods_for_a_given_crime():
    #print("TODO: map_of_top_neighborhoods_for_a_given_crime")

    connection = sqlite3.connect(_DATABASE_PATH)
    cur = connection.cursor()

    lower_limit = utils.input_int_and_validate_with_predicate("Enter start year: ",check_if_int_is_year_format)
    upper_limit = utils.input_int_and_validate_with_predicate("Enter end year: ",check_if_int_is_year_format)
    crime_type = input("Enter crime type: ")
    num_neighborhood = utils.input_int_and_validate_with_predicate("Enter number of neighborhoods: ",check_if_int_is_non_negative_and_handl

    cur.execute("SELECT sum(i.Incidents_Count) as counts, i.Neighbourhood_Name, c.Latitude, c.Longitude \
    FROM coordinates c  \
    LEFT JOIN crime_incidents i on c.Neighbourhood_Name = i.Neighbourhood_Name  \
    WHERE i.Year >= ? AND  \
        i.Year <= ? AND  \
        i.Crime_Type = ?  \
        GROUP BY i.Neighbourhood_Name  \
        ORDER BY counts DESC  \
        LIMIT ?  " , (str(lower_limit), str(upper_limit),crime_type,str(num_neighborhood)) )

    nList = cur.fetchall()
    bot_of_top = nList[num_neighborhood-1][0]
    #print(nList)


    cur.execute("SELECT sum(i.Incidents_Count) as counts, i.Neighbourhood_Name, c.Latitude, c.Longitude \
    FROM coordinates c  \
    LEFT JOIN crime_incidents i on c.Neighbourhood_Name = i.Neighbourhood_Name  \
    WHERE i.Year >= ? AND  \
        i.Year <= ? AND  \
        i.Crime_Type = ? \
        GROUP BY i.Neighbourhood_Name \
        HAVING counts >= ? \
        ORDER BY counts DESC" , (str(lower_limit), str(upper_limit),crime_type, bot_of_top) )

    newList = cur.fetchall()
    markers = []
    avg_val = 0

    edmonton_map = a4_specific_utils.create_new_edmonton_map()

    for n in newList:
        nPopup = "%s <br> %s" % (n[1],n[0])
        markers.append(FolioMarker([n[2], n[3]], nPopup, 'crimson', n[0]))
        avg_val += n[0]

    avg_val /= len(newList)
    a4_specific_utils.add_markers_to_map(edmonton_map, markers, avg_val)
    a4_specific_utils.write_map_to_file(edmonton_map, "Q3")
```