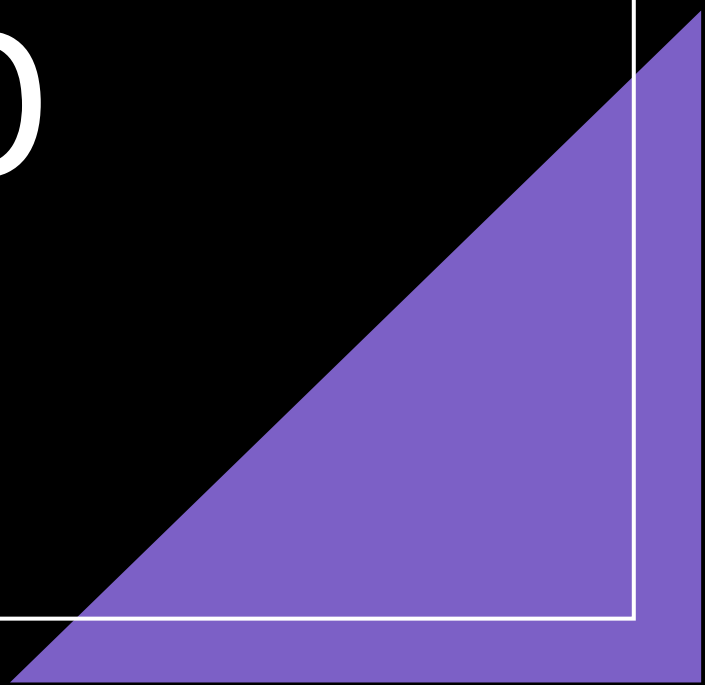


# Rate My Pub

Lukas

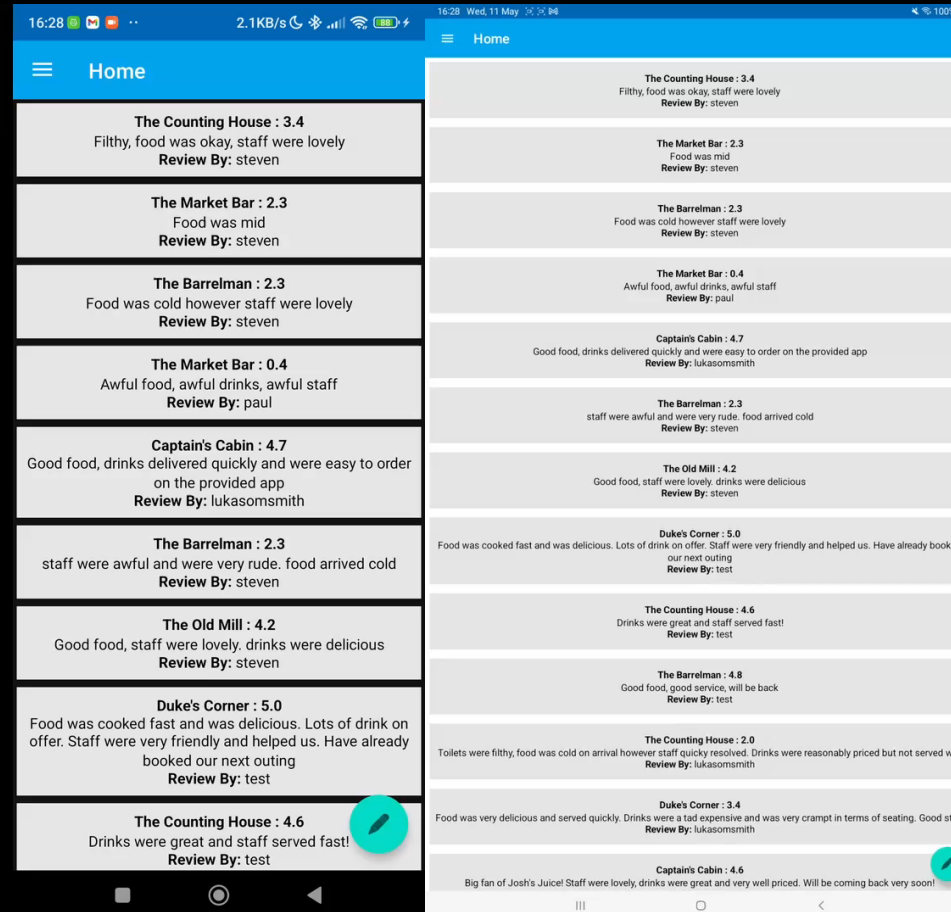


# Demonstration









### **Captain's Cabin : 4.7**

Good food, drinks delivered quickly and were easy to order  
on the provided app

**Review By:** lukasomsmith

### **The Barrelman : 2.3**

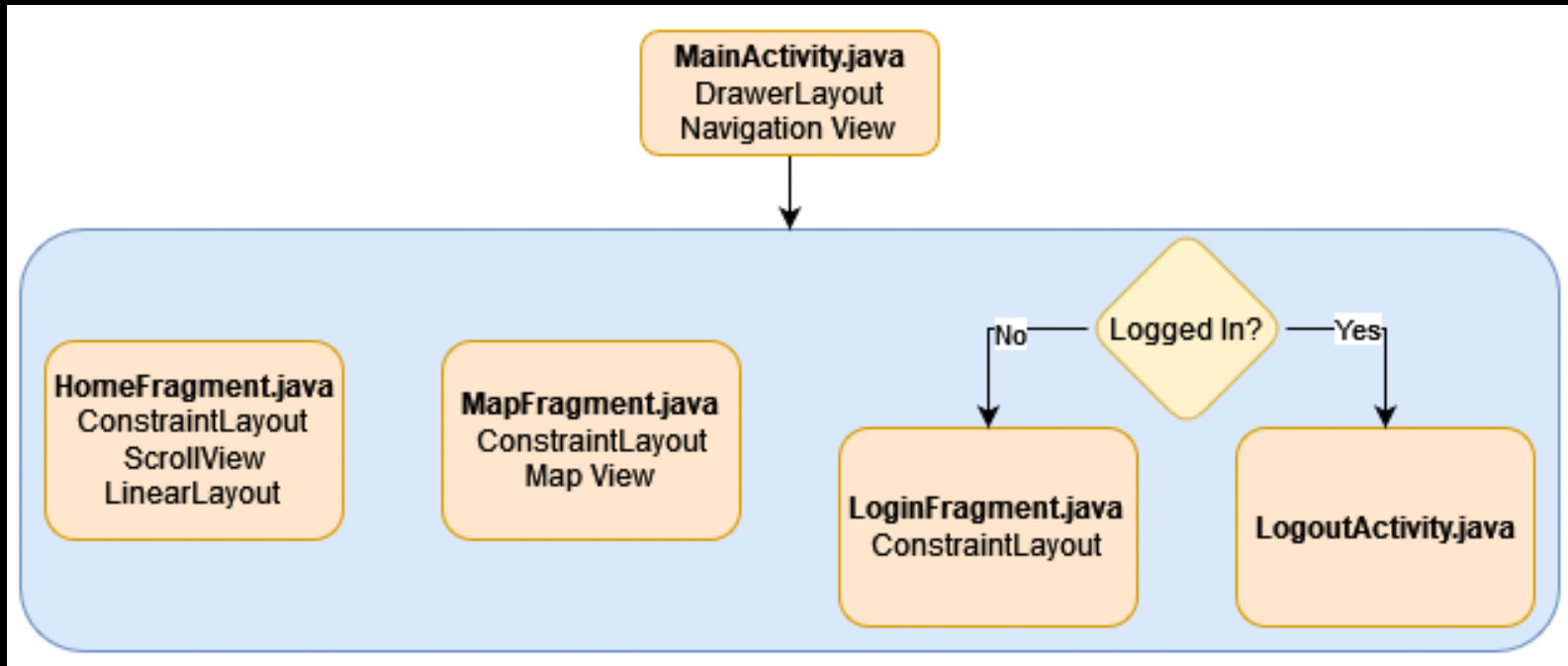
staff were awful and were very rude. food arrived cold

**Review By:** steven

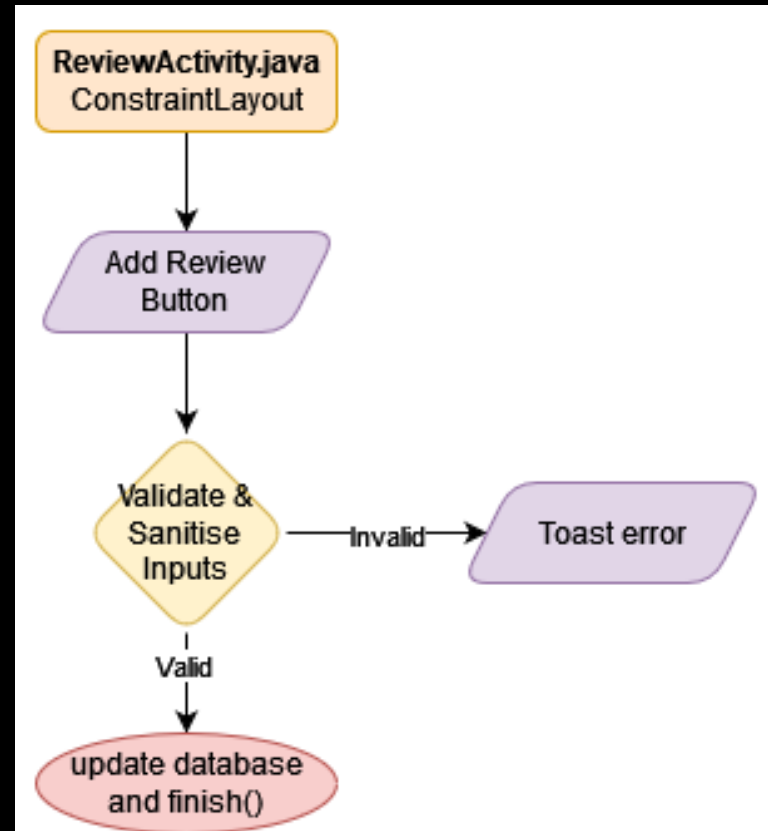
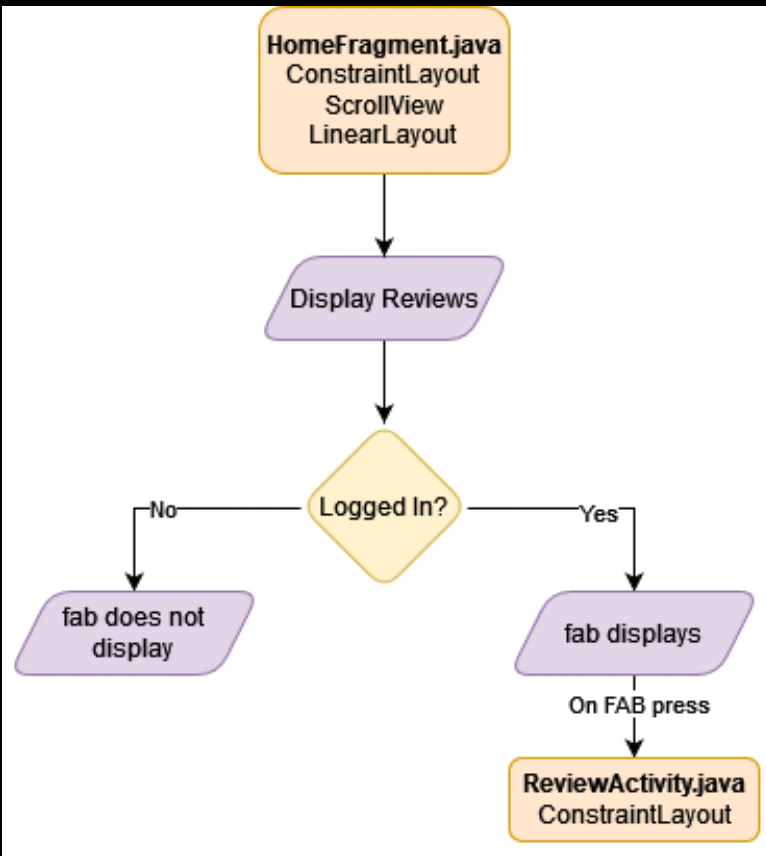
### **The Old Mill : 4.2**

Good food, staff were lovely. drinks were delicious

**Review By:** steven



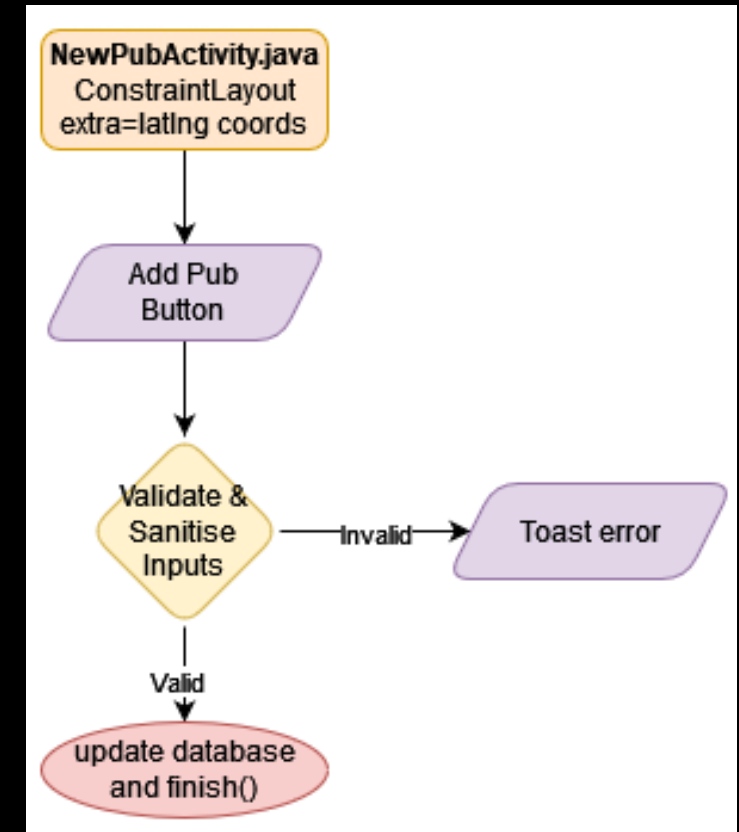
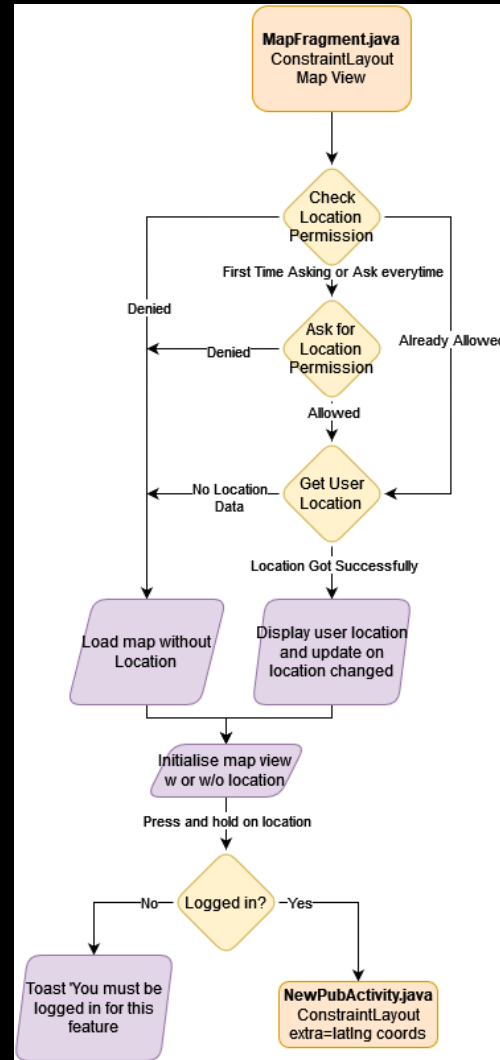
Flow  
Diagram



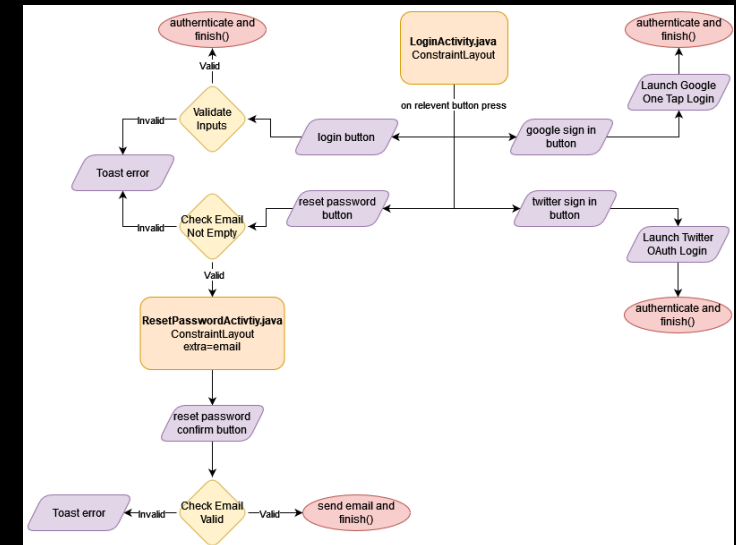
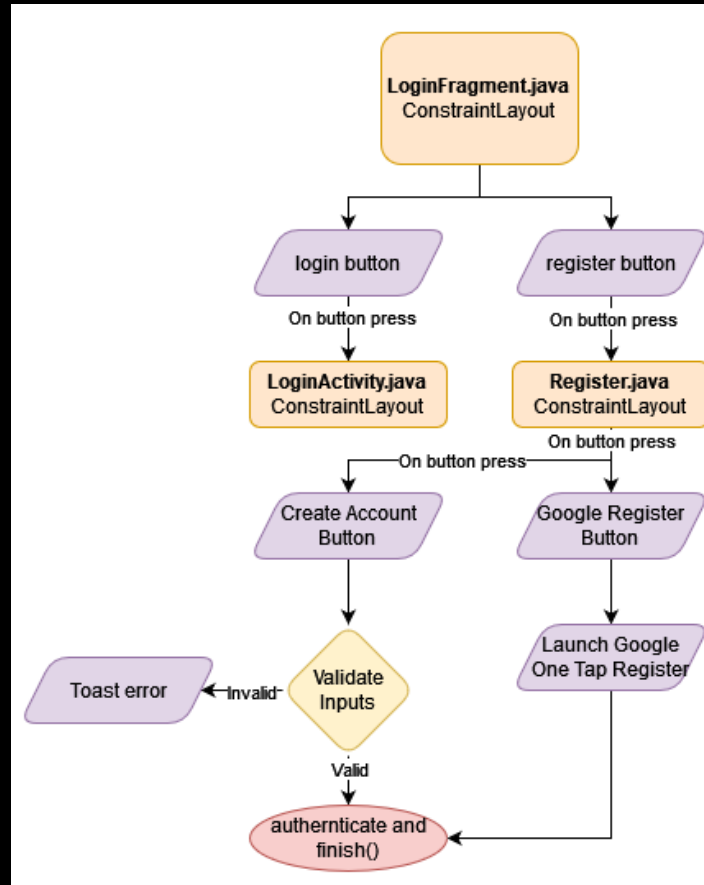
HomeFragment.java

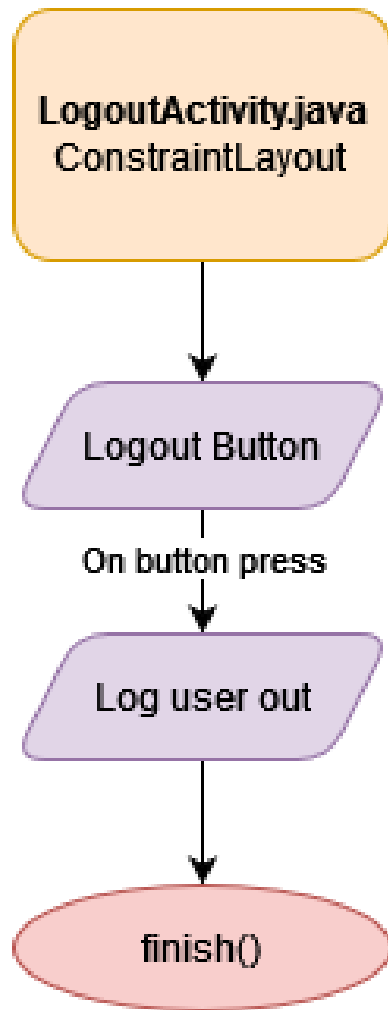


# MapFragment.java



# LoginFragment.java





# LogoutActivity.java

# Code – MainActivity.java

```
@Override
public void onStart() {
    super.onStart();

    // setup nav drawer
    DrawerLayout drawer = binding.drawerLayout;
    NavigationView navigationView = binding.navView;

    // check if user is logged in and update header
    FirebaseUser currentUser = mAuth.getCurrentUser();
    editHeaderText(currentUser);

    // setup navbar
    mAppBarConfiguration = new AppBarConfiguration.Builder(
        R.id.nav_home, R.id.nav_map, R.id.nav_login, R.id.nav_logout)
        .setOpenableLayout(drawer)
        .build();

    // show login/register or logout depending on user status
    navigationView.getMenu().findItem(R.id.nav_login).setVisible(currentUser == null);
    navigationView.getMenu().findItem(R.id.nav_logout).setVisible(currentUser != null);

    // finish nav setup
    NavController navController = Navigation.findNavController( this, R.id.nav_host_fragment_content_main);
    NavigationUI.setupActionBarWithNavController( this, navController, mAppBarConfiguration);
    NavigationUI.setupWithNavController(navigationView, navController);
}
```

```
public void editHeaderText(FirebaseUser currentUser) {
    // show start of email in header if logged in
    NavigationView nav_view = (NavigationView) findViewById(R.id.nav_view);
    View nav_view_header = nav_view.getHeaderView( 0);
    TextView textViewEmail = (TextView) nav_view_header.findViewById(R.id.nav_header_text_main);
    TextView textViewDesc = (TextView) nav_view_header.findViewById(R.id.nav_header_text_desc);

    if (currentUser != null) {
        textViewEmail.setText(String.format("%s, %s",
            "Hi",
            Objects.requireNonNull(currentUser.getEmail()).split( regex: "@" )[0]));
        textViewDesc.setText("Welcome to RateMyPub");
    } else {
        textViewEmail.setText("Hi");
        textViewDesc.setText("Login to access all features");
    }
}
```

```

// create database listener to setup TextViews and change them dynamically
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        // clear the layout and the textView list
        LinearLayout.removeAllViews();
        reviewsList.clear();
        for (DataSnapshot s : dataSnapshot.getChildren()) {
            // for each value in review db, create textView and add to list
            Reviews reviews = s.getValue(Reviews.class);
            assert reviews != null;
            TextView tv = new TextView(getContext());
            LayoutParams layoutParams = new LayoutParams(LayoutParams.MATCH_PARENT,
                LayoutParams.WRAP_CONTENT);
            layoutParams.setMargins(10, 10, 10, 10);
            tv.setLayoutParams(layoutParams);
            tv.setText(HtmlCompat.fromHtml(
                source: "<b>" + reviews.pubName + " : " +
                reviews.rating + "</b>" + "<br />" + reviews.description + "<b>" +
                "<br />Review By: " + "</b>" + reviews.user,
                HtmlCompat.FROM_HTML_MODE_COMPACT));
            tv.setTextColor(Color.BLACK);
            tv.setBackgroundColor(Color.parseColor("#F0F0F0")); // hex color 00A88080
            tv.setGravity(Gravity.CENTER);
            tv.setPadding(10, 10, 10, 10); // in pixels (left, top, right, bottom)
            LinearLayout.addView(tv, index);
            reviewsList.add(tv);
        }
    }

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        //Log.w(TAG, "Failed to read value.", error.toException());
    }
});

```

```

public class Reviews {

    public String description;
    public String pubName;
    public Double rating;
    public String user;

    public Reviews() {
        // Default constructor required for calls to DataSnapshot.getValue(User.class)
    }

    public Reviews(String description, String pubName, Double rating, String user) {
        this.description = description;
        this.pubName = pubName;
        this.rating = rating;
        this.user = user;
    }
}

```

```

if (user != null) {
    fab.setVisibility(View.VISIBLE);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(getActivity(), ReviewActivity.class);
            startActivity(intent);
        }
    });
} else {
    fab.setVisibility(View.GONE);
}

```

# Code - HomeFragment.java

# MapFragment.java

```
public View onCreateView(@NonNull LayoutInflater inflater,
                        ViewGroup container, Bundle savedInstanceState) {

    binding = FragmentMapBinding.inflate(inflater, container, attachToParent: false);
    View root = binding.getRoot();

    // check location permissions and get current location
    requestLocationPermissions();
    getLocation();

    // list to store added markers
    markerList = new ArrayList<>();

    SupportMapFragment mapFragment = SupportMapFragment.newInstance();
    getParentFragmentManager().beginTransaction()
        .add(R.id.map_view, mapFragment)
        .commit();

    mapFragment.getMapAsync( callback: this);

    return root;
}
```

```
// request location permissions
@SuppressLint("MissingPermission")
private void requestLocationPermissions() {
    AtomicBoolean permission = new AtomicBoolean( initialValue: false);
    ActivityResultLauncher<String[]> locationPermissionRequest =
        registerForActivityResult(new ActivityResultContracts
            .RequestMultiplePermissions(), result -> {
            Boolean fineLocationGranted = result.getDefault(
                Manifest.permission.ACCESS_FINE_LOCATION, defaultValue: false);
            Boolean coarseLocationGranted = result.getDefault(
                Manifest.permission.ACCESS_COARSE_LOCATION, defaultValue: false);
            if (fineLocationGranted != null && fineLocationGranted) {
                permission.set(true);
            } else if (coarseLocationGranted != null && coarseLocationGranted) {
                permission.set(true);
            } else {
                permission.set(false);
            }
        });

    locationPermissionRequest.launch(new String[]{
        Manifest.permission.ACCESS_FINE_LOCATION,
        Manifest.permission.ACCESS_COARSE_LOCATION
    });

    if (permission.get()) {
        getLocation();
    }
}
```

```
private void getLocation() {
    if (ActivityCompat.checkSelfPermission(requireActivity(), Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED &&
        ActivityCompat.checkSelfPermission(requireActivity(), Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
        try {
            locationManager = (LocationManager) requireActivity().getSystemService(Context.LOCATION_SERVICE);
            if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
                locationManager.requestLocationUpdates(LocationManager
                    .GPS_PROVIDER, minTimeMs: 5000, minDistanceM: 1, listener: this);
            } else if (locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {
                locationManager.requestLocationUpdates(LocationManager
                    .NETWORK_PROVIDER, minTimeMs: 5000, minDistanceM: 1, listener: this);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    } else {
        requestLocationPermissions();
    }
}
```

```
// setup database listener and update pub markers
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (markerList.isEmpty()) {
            for (DataSnapshot s : dataSnapshot.getChildren()) {
                Coordinates coordinates = s.getValue(Coordinates.class);
                assert coordinates != null;
                LatLng location = new LatLng(coordinates.lat, coordinates.lng);
                markerList.add(googleMap.addMarker(new MarkerOptions()
                    .position(location)
                    .snippet("Added by: " + coordinates.user)
                    .icon(BitmapDescriptorFactory.defaultMarker((float) (coordinates.rating * 24)))
                    .title(coordinates.placeName)));
            }
        } else {
            int counter = 0;
            for (DataSnapshot s : dataSnapshot.getChildren()) {
                Coordinates coordinates = s.getValue(Coordinates.class);
                assert coordinates != null;
                LatLng location = new LatLng(coordinates.lat, coordinates.lng);
                try {
                    markerList.get(counter)
                        .setPosition(location);
                    markerList.get(counter)
                        .setIcon(BitmapDescriptorFactory.defaultMarker((float) (coordinates.rating * 24)));
                    markerList.get(counter)
                        .setTitle(coordinates.placeName);
                    markerList.get(counter)
                        .setSnippet("Added by: " + coordinates.user);
                } catch (Exception e) {
                    markerList.add(googleMap.addMarker(new MarkerOptions()
                        .position(location)
                        .snippet("Added by: " + coordinates.user)
                        .icon(BitmapDescriptorFactory.defaultMarker((float) (coordinates.rating * 24)))

```

```
                        .title(coordinates.placeName)));
                    }
                    counter += 1;
                }
            }
        }

        @Override
        public void onCancelled(DatabaseError error) {
            // Failed to read value
            //Log.w(TAG, "Failed to read value.", error.toException());
        }
    });

    // move camera to centre on dundee
    googleMap.moveCamera(CameraUpdateFactory
        .newLatLngZoom(new LatLng( latitude: 56.462002, longitude: -2.970700), zoom: 12.0f));

    // setup long click listener
    googleMap.setOnMapLongClickListener(this::onMapLongClick);
}

```

```
public class Coordinates {

    public Double lat;
    public Double lng;
    public String placeName;
    public Double rating;
    public String user;

    public Coordinates() {
        // Default constructor required for calls to DataSnapshot.getValue(User.class)
    }

    public Coordinates(Double lat, Double lng, String placeName, Double rating, String user) {
        this.lat = lat;
        this.lng = lng;
        this.placeName = placeName;
        this.rating = rating;
        this.user = user;
    }
}

```

```
public void onMapLongClick(LatLng latLng) {  
    // check if logged in and if logged in launch dialog to create pub  
    FirebaseUser user = mAuth.getCurrentUser();  
    if (user != null) {  
        new AlertDialog.Builder(requireActivity())  
            .setTitle("Add a pub")  
            .setMessage(String.format("Do you want to add a pub at %s, %s", latLng.latitude, latLng.longitude))  
            .setIcon(android.R.drawable.ic_dialog_alert)  
            .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialog, int whichButton) {  
                    Intent intent = new Intent(getActivity(), NewPubActivity.class);  
                    intent.putExtra( name: "latLng", latLng);  
                    startActivity(intent);  
                }  
            })  
            .setNegativeButton(android.R.string.no, listener: null).show();  
    } else {  
        Toast.makeText(requireActivity(), text: "You must be logged in to access this feature.", Toast.LENGTH_SHORT).show();  
    }  
}
```



```

public void loginGoogle(View view) {
    oneTapClient.beginSignIn(signInRequest)
        .addOnSuccessListener( activity: this, new OnSuccessListener<BeginSignInResult>() {
            @Override
            public void onSuccess(BeginSignInResult result) {
                try {
                    startIntentSenderForResult(
                        result.getPendingIntent().getIntentSender(), REQ_ONE_TAP,
                        fillInIntent: null, flagsMask: 0, flagsValues: 0, extraFlags: 0);
                } catch (IntentSender.SendIntentException e) {
                    Toast.makeText( context: LoginActivity.this,
                        text: "Couldn't start One Tap UI: " + e.getLocalizedMessage(),
                        Toast.LENGTH_SHORT).show();
                }
            }
        })
        .addOnFailureListener( activity: this, new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText( context: LoginActivity.this,
                    Objects.requireNonNull(e.getLocalizedMessage()),
                    Toast.LENGTH_SHORT).show();
            }
        });
}

```

```

oneTapClient = Identity.getSignInClient( activity: this);
signInRequest = BeginSignInRequest.builder()
    .setGoogleIdTokenRequestOptions(BeginSignInRequest.GoogleIdTokenRequestOptions.builder()
        .setSupported(true)
        // Your server's client ID, not your Android client ID.
        .setServerClientId("618107406085-98gq3tuqt1pgneq183j20vvk854hn5ok.apps.googleusercontent.com")
        // Only show accounts previously used to sign in.
        .setFilterByAuthorizedAccounts(true)
        .build())
    .build();

```

# Login/Register Activity



# ReviewActivity

```
// setup list of string and setup adapter
stringArrayList = new ArrayList<>();
adapter = new ArrayAdapter<String>( context: this, android.R.layout.simple_spinner_dropdown_item, stringArrayList);

spinner.setAdapter(adapter);

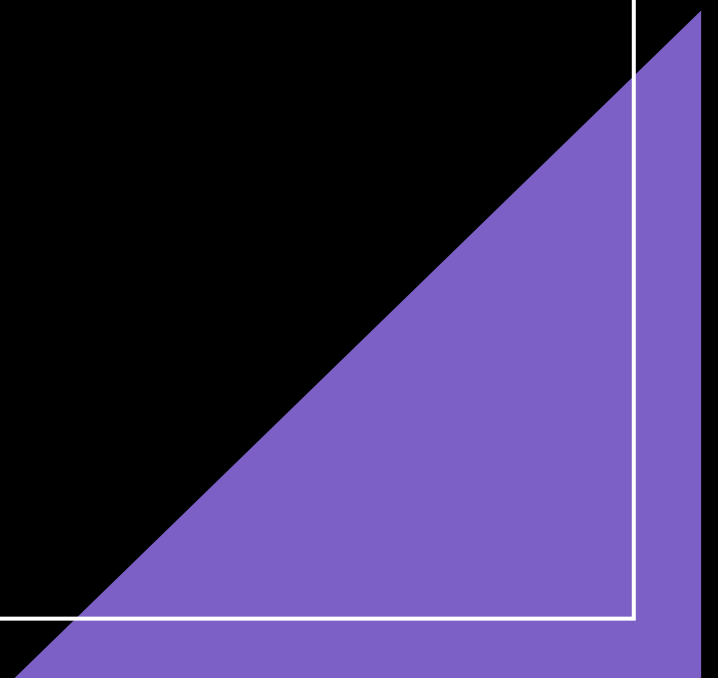
// setup listener for initial db setup and
coordinatesReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        // clear the list on data change
        stringArrayList.clear();
        for (DataSnapshot s : dataSnapshot.getChildren()){
            Coordinates coordinates = s.getValue(Coordinates.class);
            assert coordinates != null;
            stringArrayList.add(coordinates.placeName);
        }
        // update adapter
        adapter.notifyDataSetChanged();
    }
}
```



# Critical Analysis

- Realtime Database was chosen over Cloud Firestore as:
    - Program does not require advanced querying, sorting or transactions
    - A json tree is simple to use and the data did not have to be structured
    - The program sends streams of smaller updates rather than large updates
  - Firebase was chosen for authentication as it:
    - Integrates better with other Firebase services (in this case the database)
    - Supports authentication through several popular identity providers
  - Realtime Database already runs on it's own thread
-

# Critical Analysis Cont.

- Handles orientation changes well
  - Login persist due to Firebase Authentication saving user credentials to local storage
  - Linear layout with scrollview allows all reviews to be displayed
  - Spinner within the review activity reduces user error
- 

# Critical Analysis Cont.

- The app uses two explicit intents
  - Passing the Latitude and Longitude coordinates from the MapFragment to the NewPubActivity
    - Reduces user error by having it already entered for them
  - Passing the email address when resetting password.
    - Gives user last chance to ensure data is correct
    - Reduces chance of misuse



# Critical Analysis Cont.

- Google Play Services required for Google Map
  - Targeted API level 32 (Android 12)
  - Minimum API level 29 (Android 10)
-

# Critical Analysis Cont.

- Battery Drain
  - Location draining however stops requesting location when not the map fragment is not displayed
  - Internet required for downloading maps however this is it's only use
- Potential implementation of geofencing with the users current location
- Currently used to show how close in relation to other pubs
- Minimal storage usage with most data stored in database

# Critical Analysis Cont.

- Validating inputs before entering the database
- Trims inputs and REGEX's passwords to ensure secure
- Firebase security rules are easy to write expressions making it very simple to ensure that the database is secure
- The app only asks for location permissions when accessing the map fragment



(Citations next slide)

# Thanks for listening



# Citations

- Android n.d., Developer Guides | Android Developers, viewed 1 May, 2022a, <<https://developer.android.com/guide>>.
  - Android n.d., Permissions on Android | Android Developers, viewed 3 May, 2022, <<https://developer.android.com/guide/topics/permissions/overview>>.
  - Firebase n.d., Add Firebase to your Android project | Firebase Documentation, viewed 3 May, 2022, <<https://firebase.google.com/docs/android/setup>>.
  - Firebase n.d., Authenticate Using Twitter on Android | Firebase Documentation, viewed 3 May, 2022, <<https://firebase.google.com/docs/auth/android/twitter-login>>.
  - Firebase n.d., Authenticate with Google on Android | Firebase Documentation, *Firebase*, viewed 4 May, 2022, <<https://firebase.google.com/docs/auth/android/google-signin>>.
  - Firebase n.d., Firebase Authentication | Firebase Documentation, viewed 4 May, 2022, <<https://firebase.google.com/docs/auth>>.
  - Firebase n.d., Manage Users in Firebase | Firebase Documentation, viewed 5 May, 2022, <<https://firebase.google.com/docs/auth/android/manage-users>>.
  - Firebase n.d., Read and Write Data on Android | Firebase Documentation, viewed 5 May, 2022, <<https://firebase.google.com/docs/database/android/read-and-write>>.
  - Google n.d., Maps SDK for Android overview, *Google Developers*, viewed 6 May, 2022, <<https://developers.google.com/maps/documentation/android-sdk/overview>>.
  - Anon n.d., Spinners, *Android Developers*, viewed 6 May, 2022, <<https://developer.android.com/guide/topics/ui/controls/spinner>>.
-