CSCI 448 – Lab 11A
Wednesday, April 05, 2023
LAB IS DUE BY **Monday, April 17, 2023 11:59 PM**!!

Now that the map is in place, the next proof of concept idea is to change the map settings and store the user's preferences from session to session.

This lab has some flexibility, creativity, and exploration in it.

Step 0 – Add Datastore Dependency

First, in the Module level `build.gradle` file, add the following dependency:

```
dependencies {
  ...
    implementation 'androidx.datastore:datastore-preferences:1.0.0'
  ...
}
```

Now, get an idea of what settings you'll want to save. Take a look at:

- `MapProperites`: Data type for properties that can be modified on a map (see: https://googlemaps.github.io/android-maps-compose/maps-compose/com.google.maps.android.compose/-map-properties/index.html)

- `MapUISettings`: Data type for UI-related settings on a map (see: https://googlemaps.github.io/android-maps-compose/maps-compose/com.google.maps.android.compose/-map-ui-settings/index.html)

You will need to choose at least one value from each class and store its value in your Datastore.

Step 1 – Create the `DataStoreManager`

Begin by creating a class called `DataStoreManager`. Its constructor should take, and store privately, a `Context`.

Part 1.I – Create the DataStore and Keys

Inside of the companion object, create a private constant that corresponds to the name of your datastore.

```
private const val DATA_STORE_NAME = "your_name_here"
```

Now, create the extension function to access the datastore from the context:

```
private val Context.dataStore: DataStore<Preferences>
    by preferencesDataStore(name = DATA_STORE_NAME)
```

Next, store the private keys for the two preferences you are choosing to save. Be sure to choose the correct preference type based on the type of the map setting. See https://developer.android.com/reference/androidx/datastore/preferences/core/PreferencesKeys for a list of types.

```
// example
private val DATA_KEY = stringPreferencesKey("data_key")
```

Part 1.II – Read the Preference

Following the pattern from the slides, expose the flow publicly to read each of your settings.

```
// example
val dataflow: Flow<String> = context.dataStore.data.map { preferences ->
  preferences[DATA_KEY] ?: ""
}
```

Part 1.III – Write the Preference

Then create the suspend functions to write the new value of each setting.

```
// example
suspend fun setData(newValue: String) {
  context.dataStore.edit { preferences ->
    preferences[DATA_KEY] = newValue
  }
}
```

Step 2 – Toggle the Settings

In `LocationScreen`, first get a reference to the `DataStoreManager`.

```
val dataStoreManager = remember { DataStoreManager(context) }
```

Take note of the use of `remember { }`. This way, when the screen gets recomposed, we will not create a new instance of the `DataStoreManager` – we'll retain the original instance.

Part 2.I – Collect the Flow

Then collect each flow with lifecycle and store it as a state object.

```
// example
val lifecycleOwner = LocalLifecycleOwner.current
val dataState = dataStoreManager
     .dataFlow
     .collectAsStateWithLifecycle(
          initialValue = "",
          lifecycle = lifecycleOwner.lifecycle
     )
```

Part 2.II – Specify the Map Settings

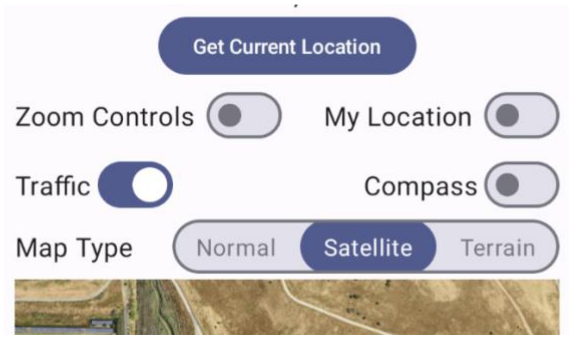
Then create the `MapUiSettings` and `MapProperties` objects and set the appropriate arguments from the state objects nbased on the settings you chose in Step 0.

```
val mapUiSettings = MapUiSettings(
  // TODO - set arguments
)
val mapProperties = MapProperties(
  // TODO - set arguments
)
```

Pass these to our `GoogleMap` composable.

```
GoogleMap(
  modifier = Modifier.fillMaxSize(),
  cameraPositionState = cameraPositionState,
  uiSettings = mapUiSettings,
  properties = mapProperties
) {
  ...
}
```

Part 2.III – Change the Settings



Now, you will need to create the composables to have events change the settings. Place these below the `Get Current Location` button and above the `GoogleMap`.

For setting needs to have a label for what the setting is and then the appropriate input type that corresponds to the preference type. The event listener for the input will need to update the corresponding preference in the DataStore. Changing a setting will result in the map updating at that time.

Step 3 - Submission

Submit a video with the following features for the final sign off:

- Open the app
- Change both settings (the map should react to the changes)
- Go to recents, close the app
- Reopen the app
- Settings should be in their last state

Please name this video file `<your_user_name>_L11`. For instance, my submission would be `jpaone_L11.webm`.

LAB IS DUE BY **Monday, April 17, 2023 11:59 PM**!!