```
@file:OptIn(ExperimentalMaterial3Api::class)

package com.example.myapplicationmmmmmm

import android.util.Log
import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Close
import androidx.compose.material.icons.filled.Done
import androidx.compose.material.icons.filled.Person
import androidx.compose.material.icons.filled.Search
import androidx.compose.material.icons.filled.Settings
import androidx.compose.material3.AssistChip
import androidx.compose.material3.AssistChipDefaults
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.FilterChip
import androidx.compose.material3.FilterChipDefaults
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.InputChip
import androidx.compose.material3.InputChipDefaults
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.SuggestionChip
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.material3.TextFieldDefaults
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
```

```
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.RectangleShape
import androidx.compose.ui.text.input.TextFieldValue
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp


//a data class to define the values within the values
data class ClubValues(
  val clubInfo: String,
  val skills: List<String>
  //skills is a list of strings so that more than one value can be in each one
)
val selectedColor = Color(0xFF973131)
val unselectedColor = Color(0xFFE0A75E)

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun FilterClubs(){
  //ListOfClubs is a value which has all the of ClubValues in it.
  val ListOfClubs by remember{mutableStateOf(listOf(
    ClubValues("Chemistry & Sustainability Club, Monday 12:00-12:40, MIV-VII, S7",
listOf("Chemistry", "Sustainability")),
    ClubValues("Football, Monday 12:40-13:15, MIV-VIII, Sports Hall",
listOf("Teamwork", "Leadership")),
    ClubValues("SPEAC, Tuesday 10:15-10:35, MIV-VIII, 203", listOf("Sustainability")),
    ClubValues("UIV-LV Debating, Tuesday 12:40-13:15, UIV-LV, G5", listOf("Public
Speaking", "Teamwork")),
    ClubValues("Basketball, Wednesday 12:40-13:15, MIV-LV, Sports Hall",
listOf("Teamwork", "Leadership")),
    ClubValues("Politics Society, Wednesday 12:40-13:15, MIV-VIII, G14",
listOf("Politics")),
    ClubValues("German Office Hours, Wednesday 12:40-13:15, All, 108",
listOf("German")),
    ClubValues("Model UN, Thursday 12:00-12:40, V-VIII, G15", listOf("Public Speaking",
"Politics")),
    ClubValues("Chemistry Geek Club, Thursday 12:40-13:15, VII-VIII, S7/S8)",
listOf("Chemistry")),
    ClubValues("Primary Hub, Thursday 16:05-17:15, V & VII, Rosalind Franklin Wing",
listOf("Volunteering", "Leadership")),
    ClubValues("Teaching German to Primary School Children, Friday 12:40-13:15, V-VI,
108", listOf("Volunteering", "Leadership", "German")),
```

```
      ClubValues("Gardening Club, Friday 12:40-13:15, MIV-VIII, S12",
listOf("Sustainability", "Leadership"))
   ))}
   //declaring variables which will be changed when a button is pressed
   var selectedChemistry by remember{mutableStateOf(false)}
   //this variable is whether the Chemistry chip is clicked or not
   var chemOrNot by remember{mutableStateOf("")}
   //this is the variable which goes into the chip. when it is not clicked, it is empty, but
when it is clicked, it will be Chemistry
   var selectedSustainability by remember{mutableStateOf(false)}
   var sustainabilityOrNot  by remember {mutableStateOf("")}
   var selectedTeamwork by remember{mutableStateOf(false)}
   var teamworkOrNot by remember{mutableStateOf("")}
   var selectedSpeaking by remember{ mutableStateOf(false)}
   var speakingOrNot by remember{ mutableStateOf("")}
   var selectedPolitics by remember{ mutableStateOf(false)}
   var politicsOrNot by remember{ mutableStateOf("")}
   var selectedVolunteering by remember{ mutableStateOf(false)}
   var volunteeringOrNot by remember{ mutableStateOf("")}
   var selectedLeadership by remember{ mutableStateOf(false)}
   var leadershipOrNot by remember{ mutableStateOf("")}
   var selectedGerman by remember{ mutableStateOf(false)}
   var germanOrNot by remember{ mutableStateOf("")}

   var searchQuery by remember { mutableStateOf("") }
//this is the value which is in the search bar
   Column(modifier = Modifier
      .verticalScroll(rememberScrollState())
      .background(Color((0xFFF5E7B2)))
      .fillMaxSize()){
      TextField(
         //the search bar. When something is typed into the search bar, searchQuery
becomes that
         value = searchQuery,
         onValueChange = { searchQuery = it },
         label = { Text("Search Clubs") },
         modifier = Modifier
            .fillMaxWidth()
            .border(BorderStroke(1.dp, Color.Black)),
         colors = TextFieldDefaults.textFieldColors(
            containerColor = Color(0xFFF5E7B2),
            textColor = Color.Black,
            cursorColor = Color.Black,
            focusedIndicatorColor = Color.Transparent,
            unfocusedIndicatorColor = Color.Transparent
            ))
      Row(//this row contains the text 'filter', with the icon next to it
```

```
        modifier = Modifier
            .fillMaxWidth()
            .padding(horizontal = 16.dp, vertical = 8.dp),
        horizontalArrangement = Arrangement.End,//puts it on the right
    ) {
        Text(
            text = "Filter",//text title to show that you can filter
            fontSize = 12.sp,
            color = Color.Black
        )
        Spacer(modifier = Modifier.width(4.dp))
        Icon(
            painter = painterResource(id = R.drawable.filter_icon), //adds the filter icon
            contentDescription = null,
            modifier = Modifier.size(16.dp),
            tint = Color.Black
        )
    }

    Spacer(modifier = Modifier.height(2.dp))
    Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement =
Arrangement.SpaceEvenly) {
        FilterChip(
            onClick = {
                selectedChemistry = !selectedChemistry
                //when clicked the chip goes from selected to not selected or vice versa
                chemOrNot = if(chemOrNot.isEmpty()) "Chemistry" else ""
                //the value will change to Chemistry when the chip is clicked, otherwise it is
empty
                    },

            label = {
                Text("Chemistry",
                    fontSize = 17.sp,
                    modifier = Modifier.fillMaxWidth(),
                    textAlign = TextAlign.Center,
                    color = if (selectedChemistry) Color(0xFFF5E7B2) else Color.Black)
            },
            selected = selectedChemistry,
            modifier = Modifier
                .height(50.dp)
                .padding(5.dp)
                .width(150.dp),
            colors = FilterChipDefaults.filterChipColors(
                containerColor = unselectedColor,
                selectedContainerColor = selectedColor),
            leadingIcon = if (selectedChemistry) {
```

```
      {
        Icon(
          imageVector = Icons.Filled.Done,
          contentDescription = "Done icon",
          modifier = Modifier.size(FilterChipDefaults.IconSize)
        //this changes what it looks like if the chip is selected
        )
      }
    }
    else {null}
  //the same for the rest of the chips
  )
  FilterChip(
    onClick = {
      selectedSustainability = !selectedSustainability
      sustainabilityOrNot = if(sustainabilityOrNot.isEmpty()) "Sustainability" else ""
    },

    label = {
      Text("Sustainability",
        fontSize = 17.sp,
        modifier = Modifier.fillMaxWidth(),
        textAlign = TextAlign.Center,
        color = if (selectedSustainability) Color(0xFFF5E7B2) else Color.Black)
    },
    selected = selectedSustainability,
    modifier = Modifier
      .height(50.dp)
      .padding(5.dp)
      .width(150.dp),
    colors = FilterChipDefaults.filterChipColors(
      containerColor = unselectedColor,
      selectedContainerColor = selectedColor),
    leadingIcon = if (selectedSustainability) {
      {
        Icon(
          imageVector = Icons.Filled.Done,
          contentDescription = "Done icon",
          modifier = Modifier.size(FilterChipDefaults.IconSize)
        )
      }
    }
    else {null}
  )
  FilterChip(
    onClick = {
      selectedTeamwork = !selectedTeamwork
```

```
            teamworkOrNot = if(teamworkOrNot.isEmpty()) "Teamwork" else ""
          },

          label = {
            Text("Teamwork",
              fontSize = 17.sp,
              modifier = Modifier.fillMaxWidth(),
              textAlign = TextAlign.Center,
              color = if (selectedTeamwork) Color(0xFFF5E7B2) else Color.Black)
          },
          selected = selectedTeamwork,
          modifier = Modifier
            .height(50.dp)
            .padding(5.dp)
            .width(150.dp),
          colors = FilterChipDefaults.filterChipColors(
            containerColor = unselectedColor,
            selectedContainerColor = selectedColor),
          leadingIcon = if (selectedTeamwork) {
            {
              Icon(
                imageVector = Icons.Filled.Done,
                contentDescription = "Done icon",
                modifier = Modifier.size(FilterChipDefaults.IconSize)
              )
            }
          }
          else {null}
        )}
      Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement =
Arrangement.SpaceEvenly) {
        FilterChip(
          onClick = {
            selectedSpeaking = !selectedSpeaking
            speakingOrNot = if(speakingOrNot.isEmpty()) "Public Speaking" else ""
          },

          label = {
            Text("Public Speaking",
              fontSize = 17.sp,
              modifier = Modifier.fillMaxWidth(),
              textAlign = TextAlign.Center,
              color = if (selectedSpeaking) Color(0xFFF5E7B2) else Color.Black)
          },
          selected = selectedSpeaking,
          modifier = Modifier
            .height(50.dp)
```

```
      .padding(5.dp)
      .width(150.dp),
    colors = FilterChipDefaults.filterChipColors(
      containerColor = unselectedColor,
      selectedContainerColor = selectedColor),
    leadingIcon = if (selectedSpeaking) {
      {
        Icon(
          imageVector = Icons.Filled.Done,
          contentDescription = "Done icon",
          modifier = Modifier.size(FilterChipDefaults.IconSize)
        )
      }
    }
    else {null}
  )
  FilterChip(
    onClick = {
      selectedPolitics = !selectedPolitics
      politicsOrNot = if(politicsOrNot.isEmpty()) "Politics" else ""
    },

    label = {
      Text("Politics",
        fontSize = 17.sp,
        modifier = Modifier.fillMaxWidth(),
        textAlign = TextAlign.Center,
        color = if (selectedPolitics) Color(0xFFF5E7B2) else Color.Black)
    },
    selected = selectedPolitics,
    modifier = Modifier
      .height(50.dp)
      .padding(5.dp)
      .width(150.dp),
    colors = FilterChipDefaults.filterChipColors(
      containerColor = unselectedColor,
      selectedContainerColor = selectedColor),
    leadingIcon = if (selectedPolitics) {
      {
        Icon(
          imageVector = Icons.Filled.Done,
          contentDescription = "Done icon",
          modifier = Modifier.size(FilterChipDefaults.IconSize)
        )
      }
    }
    else {null}
```

```
        )
        FilterChip(
          onClick = {
            selectedVolunteering = !selectedVolunteering
            volunteeringOrNot = if(volunteeringOrNot.isEmpty()) "Volunteering" else ""
          },

          label = {
            Text("Volunteering",
              fontSize = 17.sp,
              modifier = Modifier.fillMaxWidth(),
              textAlign = TextAlign.Center,
              color = if (selectedVolunteering) Color(0xFFF5E7B2) else Color.Black)
          },
          selected = selectedVolunteering,
          modifier = Modifier
            .height(50.dp)
            .padding(5.dp)
            .width(150.dp),
          colors = FilterChipDefaults.filterChipColors(
            containerColor = unselectedColor,
            selectedContainerColor = selectedColor),
          leadingIcon = if (selectedVolunteering) {
            {
              Icon(
                imageVector = Icons.Filled.Done,
                contentDescription = "Done icon",
                modifier = Modifier.size(FilterChipDefaults.IconSize)
              )
            }
          }
          else {null}
        )}
      Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement =
Arrangement.SpaceEvenly) {
        FilterChip(
          onClick = {
            selectedLeadership = !selectedLeadership
            leadershipOrNot = if(leadershipOrNot.isEmpty()) "Leadership" else ""
          },

          label = {
            Text("Leadership",
              fontSize = 17.sp,
              modifier = Modifier.fillMaxWidth(),
              textAlign = TextAlign.Center,
              color = if (selectedLeadership) Color(0xFFF5E7B2) else Color.Black)
```

```
    },
    selected = selectedLeadership,
    modifier = Modifier
      .height(50.dp)
      .padding(5.dp)
      .width(150.dp),
    colors = FilterChipDefaults.filterChipColors(
      containerColor = unselectedColor,
      selectedContainerColor = selectedColor),
    leadingIcon = if (selectedLeadership) {
      {
        Icon(
          imageVector = Icons.Filled.Done,
          contentDescription = "Done icon",
          modifier = Modifier.size(FilterChipDefaults.IconSize)
        )
      }
    }
    else {null}
  )
  FilterChip(
    onClick = {
      selectedGerman = !selectedGerman
      germanOrNot = if(germanOrNot.isEmpty()) "German" else ""
    },

    label = {
      Text("German",
        fontSize = 17.sp,
        modifier = Modifier.fillMaxWidth(),
        textAlign = TextAlign.Center,
        color = if (selectedGerman) Color(0xFFF5E7B2) else Color.Black)
    },
    selected = selectedGerman,
    modifier = Modifier
      .height(50.dp)
      .padding(5.dp)
      .width(150.dp),
    colors = FilterChipDefaults.filterChipColors(
      containerColor = unselectedColor,
      selectedContainerColor = selectedColor),
    leadingIcon = if (selectedGerman) {
      {
        Icon(
          imageVector = Icons.Filled.Done,
          contentDescription = "Done icon",
          modifier = Modifier.size(FilterChipDefaults.IconSize)
```

```
                )
            }
        }
        else {null}
    )}
    Text("Clubs",
    fontSize = 20.sp,
    modifier = Modifier.fillMaxWidth(),
    textAlign = TextAlign.Center)
    //filtered clubs is a value which is filtered to include only the values containing the values of the chips in their value 'skills'
    //it also only shows the values which have searchQuery in their names

    val filteredClubs = ListOfClubs.filter{ club ->
      (chemOrNot.isEmpty() || club.skills.contains(chemOrNot)) &&
          (sustainabilityOrNot.isEmpty() || club.skills.contains(sustainabilityOrNot)) &&
          (teamworkOrNot.isEmpty() || club.skills.contains(teamworkOrNot)) &&
          (politicsOrNot.isEmpty() || club.skills.contains(politicsOrNot)) &&
          (speakingOrNot.isEmpty() || club.skills.contains(speakingOrNot)) &&
          (leadershipOrNot.isEmpty() || club.skills.contains(leadershipOrNot)) &&
          (volunteeringOrNot.isEmpty() || club.skills.contains(volunteeringOrNot)) &&
          (germanOrNot.isEmpty() || club.skills.contains(germanOrNot)) &&
          (searchQuery.isEmpty() || club.clubInfo.contains(searchQuery, ignoreCase = true))
    }
    filteredClubs.forEach { club ->
      Box(
        modifier = Modifier
          .fillMaxWidth()
          .padding(horizontal = 16.dp, vertical = 8.dp)
          .border(
            BorderStroke(2.dp, Color.Black),
            shape = RoundedCornerShape(8.dp) //adds rounded corners to the border
          )
          .padding(16.dp) // Adds padding inside the border
      ) {
        Text(
          club.clubInfo
        )
      }
    }

      //it will show filteredClubs
    }
  }
```