package com.shruti.lofo.ui.Found;  
  
import static android.app.Activity.*RESULT\_OK*;  
import static android.content.ContentValues.*TAG*;  
  
import android.app.DatePickerDialog;  
import android.app.Dialog;  
import android.content.Intent;  
  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.firestore.FirebaseFirestore;  
import com.google.firebase.firestore.QueryDocumentSnapshot;  
import com.google.firebase.storage.FirebaseStorage;  
import com.google.firebase.storage.StorageReference;  
  
import android.net.Uri;  
import android.os.Bundle;  
import android.provider.MediaStore;  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageButton;  
import android.widget.ImageView;  
import android.widget.Spinner;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.fragment.app.DialogFragment;  
  
import com.shruti.lofo.Utility;  
  
import com.google.firebase.firestore.DocumentReference;  
import com.shruti.lofo.R;  
import com.shruti.lofo.OnImageUploadCallback;  
  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.Locale;  
  
  
public class FoundItemsFragment extends DialogFragment {  
 private ImageButton datePickerButton;  
  
 private TextView dateEdit;  
 private Spinner categorySpinner;  
 ImageView image;  
 Button upload;  
 Uri imageUri;  
  
 EditText description;  
  
 private EditText location ;  
 String date= null;  
  
 final int REQ\_CODE=1000;  
 String imageUrl;  
 private int mYear, mMonth, mDay, mHour, mMinute;  
  
 @NonNull  
 @Override  
 public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {  
 return super.onCreateDialog(savedInstanceState);  
 }  
  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.*fragment\_found\_items*, container, false);  
 }  
  
 @Override  
 public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 super.onViewCreated(view, savedInstanceState);  
  
 description = view.findViewById(R.id.*description*);  
 datePickerButton = view.findViewById(R.id.*datePickerButton*);  
 datePickerButton.setOnClickListener(v -> showDatePicker());  
 dateEdit= view.findViewById(R.id.*selectedDateEditText*);  
 location= view.findViewById(R.id.*location*);  
  
  
 categorySpinner = view.findViewById(R.id.*categorySpinner*);  
  
  
 ArrayAdapter<CharSequence> adapter = ArrayAdapter.*createFromResource*(requireContext(),  
 R.array.*categories\_array*, android.R.layout.*simple\_spinner\_item*);  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 categorySpinner.setAdapter(adapter);  
  
 final String[] selectedCategory = new String[1];  
 categorySpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
  
 selectedCategory[0] = categorySpinner.getItemAtPosition(position).toString();  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> parent) {  
 // Handle the case when nothing is selected  
  
 }  
 });  
  
 upload = view.findViewById(R.id.*uploadImageButton*);  
  
 upload.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent iGallery = new Intent(Intent.*ACTION\_PICK*);  
 iGallery.setData(MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*);  
 startActivityForResult(iGallery,REQ\_CODE);  
 }  
 });  
  
 Button submitButton = view.findViewById(R.id.*submit\_button*);  
  
 submitButton.setOnClickListener(v -> {  
  
 EditText item = view.findViewById(R.id.*item\_name\_edittext*);  
 String itemName = item.getText().toString();  
  
  
  
 // validation  
 if (itemName.isEmpty()) {  
 Utility.*showToast*(getContext(), "Name cannot be empty");  
 return;  
 }  
  
 // Check if category is selected  
 if (selectedCategory[0] == null) {  
 Utility.*showToast*(getContext(), "Please select a category");  
 return;  
 }  
  
  
  
 if (date == null) {  
 showDatePicker();  
 return;  
 }  
  
 // Check that location is not empty  
 String loc = location.getText().toString();  
 if (loc.isEmpty()) {  
 Utility.*showToast*(getContext(), "Please provide location");  
 return;  
 }  
  
 // Check that description is not empty  
 String desc = description.getText().toString();  
 if (desc.isEmpty()) {  
 Utility.*showToast*(getContext(), "Please add description");  
 return;  
 }  
  
 if(imageUri==null){  
 Utility.*showToast*(getContext(),"Please upload the image of the thing you found");  
 return;  
 }  
  
  
 FoundItems FoundItem = new FoundItems();  
 FoundItem.setItemName(itemName);  
 FoundItem.setCategory(selectedCategory[0]);  
 FoundItem.setDateFound(date);  
 FoundItem.setImageURI(imageUri.toString());  
 FoundItem.setLocation(location.getText().toString());  
 FoundItem.setDescription(description.getText().toString());  
  
  
 FirebaseAuth mAuth = FirebaseAuth.*getInstance*();  
 FirebaseUser currentUser = mAuth.getCurrentUser();  
 FirebaseFirestore db = FirebaseFirestore.*getInstance*();  
  
 if (currentUser != null) {  
 String userEmail = currentUser.getEmail();  
 String userID = currentUser.getUid();  
  
 // Query the user collection for the current user's details based on their email  
 db.collection("users")  
 .whereEqualTo("email", userEmail)  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : task.getResult()) {  
 // Retrieve the user's name and phone number from the document  
 String userName = document.getString("name");  
 String userPhone = document.getString("phone");  
 FoundItem.setfinderName(userName);  
 FoundItem.setPhnum(userPhone);  
 FoundItem.setEmail(userEmail);  
 FoundItem.setfinderId(userID);  
 }  
 } else {  
 Log.*d*(*TAG*, "Error getting documents: ", task.getException());  
 }  
 });  
 }  
 saveItemToFirebase(FoundItem);  
  
 });  
  
 }  
 private String generateImageName() {  
 String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss", Locale.*getDefault*()).format(new Date());  
 return "image\_" + timeStamp + ".jpg";  
 }  
 private void saveImageToFirebaseStorage(Uri imageUri, OnImageUploadCallback callback) {  
  
 String imageName = generateImageName();  
 StorageReference storageReference = FirebaseStorage.*getInstance*().getReference().child("foundImages/" + imageName);  
  
 storageReference.putFile(imageUri)  
 .addOnSuccessListener(taskSnapshot -> {  
 storageReference.getDownloadUrl().addOnSuccessListener(uri -> {  
 String imageUrl = uri.toString();  
 callback.onSuccess(imageUrl);  
 });  
 })  
 .addOnFailureListener(e -> {  
 // Handle any errors that may occur during the upload  
 callback.onFailure();  
 });  
 }  
  
  
 private void saveItemToFirebase(FoundItems item) {  
 try {  
 saveImageToFirebaseStorage(imageUri, new OnImageUploadCallback() {  
 @Override  
 public void onSuccess(String imageUrl) {  
 item.setImageURI(imageUrl);  
 DocumentReference documentReference = Utility.*getCollectionReferrenceForFound*().document();  
 documentReference.set(item).addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 Utility.*showToast*(getContext(), "Item added successfully");  
 dismiss();  
 } else {  
 Utility.*showToast*(getContext(), "Failed to add item");  
 dismiss();  
 }  
 });  
 }  
  
 @Override  
 public void onFailure() {  
 Utility.*showToast*(getContext(), "An error occurred while uploading the image");  
 }  
 });  
 } catch (Exception e) {  
 e.printStackTrace();  
 Utility.*showToast*(getContext(), "An error occurred while saving data");  
 }  
 }  
  
  
 @Override  
 public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
  
 if(resultCode == *RESULT\_OK*){  
 if(requestCode == REQ\_CODE){  
 // for gallery  
 imageUri = data.getData();  
 upload.setText("Image added");  
  
 }  
 }  
 }  
  
  
 private void showDatePicker() {  
 final Calendar c = Calendar.*getInstance*();  
 mYear = c.get(Calendar.*YEAR*);  
 mMonth = c.get(Calendar.*MONTH*);  
 mDay = c.get(Calendar.*DAY\_OF\_MONTH*);  
  
 DatePickerDialog datePickerDialog = new DatePickerDialog(requireContext(),  
 (view, year, monthOfYear, dayOfMonth) -> {  
 mYear = year;  
 mMonth = monthOfYear;  
 mDay = dayOfMonth;  
 updateDateButton();  
 }, mYear, mMonth, mDay);  
 datePickerDialog.show();  
 updateDateButton();  
 }  
  
 private String updateDateButton() {  
 String date = mDay + "/" + (mMonth + 1) + "/" + mYear;  
 dateEdit.setText(date);  
 this.date=date;  
 return date;  
 }  
  
}