Android Studio

1. LoginActivity.java

```
package com.example.agrifarmer.activities.registration;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.os.Handler;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationSet;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;

import com.airbnb.lottie.LottieAnimationView;
import com.example.agrifarmer.R;
import com.example.agrifarmer.activities.MainActivity;
import com.example.agrifarmer.getterSetterClasses.Profile;
import com.example.agrifarmer.localDatabases.MyProfileDbHandler;
import com.google.android.material.checkbox.MaterialCheckBox;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.Objects;

public class LoginActivity extends AppCompatActivity {

    private ScrollView scrollView;
    private ImageView backBtn, profileIcon;
    private TextView title, subtitle, forgotPassword, signup, login;
    private TextInputLayout email, password;
```

```
    private MaterialCheckBox rememberMe;
    private LinearLayout newUserSignup;

    private CountDownTimer backBtnTimer, signUpTimer;
    private Dialog dialog;
    private ProgressDialog progressDialog;

    private FirebaseAuth auth;

    MyProfileDbHandler profileDbHandler = new MyProfileDbHandler(LoginActivity.this);

    Animation animLeft1, animLeft2, animRight1, animRight2, animTop1, animTop2, animBottom1,
animBottom2;
    Animation ranimLeft1, ranimLeft2, ranimRight1, ranimRight2, ranimTop1, ranimTop2,
ranimBottom1, ranimBottom2;

    AnimationSet setAnimLeft = new AnimationSet(true);
    AnimationSet setAnimRight = new AnimationSet(true);
    AnimationSet setAnimTop = new AnimationSet(true);
    AnimationSet setAnimBottom = new AnimationSet(true);
    AnimationSet rsetAnimLeft = new AnimationSet(true);
    AnimationSet rsetAnimRight = new AnimationSet(true);
    AnimationSet rsetAnimTop = new AnimationSet(true);
    AnimationSet rsetAnimBottom = new AnimationSet(true);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setUIViews();
        setAndLoadAnimations();

        auth = FirebaseAuth.getInstance();

        signUpTimer = new CountDownTimer(1300, 100) {
            @Override
            public void onTick(long l) {}
            @Override
            public void onFinish() {
                backBtn.setVisibility(View.INVISIBLE);
                profileIcon.setVisibility(View.INVISIBLE);
                title.setVisibility(View.INVISIBLE);
                subtitle.setVisibility(View.INVISIBLE);
                email.setVisibility(View.INVISIBLE);
                password.setVisibility(View.INVISIBLE);
                rememberMe.setVisibility(View.INVISIBLE);
                forgotPassword.setVisibility(View.INVISIBLE);
                login.setVisibility(View.INVISIBLE);
                newUserSignup.setVisibility(View.INVISIBLE);

                Intent intent = new Intent(LoginActivity.this, Register1Activity.class);
```

```
                intent.setFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
                startActivity(intent);
                finish();
            }
        };

        backBtnTimer = new CountDownTimer(1300, 100) {
            @Override
            public void onTick(long l) {}
            @Override
            public void onFinish() {
                backBtn.setVisibility(View.INVISIBLE);
                profileIcon.setVisibility(View.INVISIBLE);
                title.setVisibility(View.INVISIBLE);
                subtitle.setVisibility(View.INVISIBLE);
                email.setVisibility(View.INVISIBLE);
                password.setVisibility(View.INVISIBLE);
                rememberMe.setVisibility(View.INVISIBLE);
                forgotPassword.setVisibility(View.INVISIBLE);
                login.setVisibility(View.INVISIBLE);
                newUserSignup.setVisibility(View.INVISIBLE);

                finish();
            }
        };

        backBtn.startAnimation(setAnimLeft);
        profileIcon.startAnimation(setAnimTop);
        title.startAnimation(setAnimLeft);
        subtitle.startAnimation(setAnimLeft);
        email.startAnimation(setAnimRight);
        password.startAnimation(setAnimLeft);
        rememberMe.startAnimation(setAnimLeft);
        forgotPassword.startAnimation(setAnimRight);
        login.startAnimation(setAnimBottom);
        newUserSignup.startAnimation(setAnimBottom);

        backBtn.setOnClickListener(view -> onBackPressed());

        signup.setOnClickListener(view -> {
            backBtn.startAnimation(rsetAnimLeft);
            profileIcon.startAnimation(rsetAnimTop);
            title.startAnimation(rsetAnimLeft);
            subtitle.startAnimation(rsetAnimLeft);
            email.startAnimation(rsetAnimRight);
            password.startAnimation(rsetAnimLeft);
            rememberMe.startAnimation(rsetAnimLeft);
            forgotPassword.startAnimation(rsetAnimRight);
            login.startAnimation(rsetAnimBottom);
            newUserSignup.startAnimation(rsetAnimBottom);

            signUpTimer.start();
        });

        email.getEditText().addTextChangedListener(new TextWatcher() {
```

```
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
                String emailPattern = "[a-zA-Z0-9._-]+@[a-z]+\\.+[a-z]+";
                String txtEmail = email.getEditText().getText().toString().trim();
                if (txtEmail.equals(""))
                    email.setError("Empty Credential!");
                else if (!txtEmail.matches(emailPattern))
                    email.setError("Invalid Email!");
                else
                    email.setError(null);
            }
        });

        password.getEditText().addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
                String passwordPattern =
"^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&*+=()])(?=\\S+$).{8,20}$";
                String txtPassword = password.getEditText().getText().toString().trim();
                if (txtPassword.equals(""))
                    password.setError("Empty Credential!");
                else if (!txtPassword.matches(passwordPattern))
                    // Add a dialog here to specify all conditions for password
                    password.setError("Doesn't meet all conditions!");
                else
                    password.setError(null);
            }
        });

        login.setOnClickListener(view -> {
            progressDialog = new ProgressDialog(LoginActivity.this);
            progressDialog.setCancelable(false);
            String txtEmail = email.getEditText().getText().toString().trim();
            String txtPassword = password.getEditText().getText().toString().trim();

            if (txtEmail.equals("") || txtPassword.equals("")) {
                if (txtEmail.equals("")) {
                    email.setError("Empty Credential!");
                }
                if (txtPassword.equals("")) {
                    password.setError("Empty Credential!");
                }
            }else {
                progressDialog.setMessage("Authenticating...");
                progressDialog.show();
```

```
                dialog = new Dialog(LoginActivity.this);
                dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
                dialog.setContentView(R.layout.message_dialog);
                dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
                dialog.setCancelable(false);

                LottieAnimationView animationView =
dialog.findViewById(R.id.message_lottie_animation);
                TextView message = dialog.findViewById(R.id.message_textview);
                Button positiveBtn = dialog.findViewById(R.id.message_positive_btn);
                Button negativeBtn = dialog.findViewById(R.id.message_negative_btn);
                negativeBtn.setVisibility(View.GONE);

                if (email.getError() == null && password.getError() == null) {
                    auth.signInWithEmailAndPassword(txtEmail, txtPassword)
                            .addOnCompleteListener(this, task -> {
                                progressDialog.dismiss();

                                if (task.isSuccessful()) {
                                    DatabaseReference reference =
FirebaseDatabase.getInstance().getReference().child("UserInfo").child(auth.getUid());
                                    reference.addValueEventListener(new ValueEventListener()
{
                                        @Override
                                        public void onDataChange(@NonNull DataSnapshot
snapshot) {
                                            Profile profile =
snapshot.getValue(Profile.class);
                                            new Handler().postDelayed(() ->
profileDbHandler.addUser(profile), 2000);
                                        }

                                        @Override
                                        public void onCancelled(@NonNull DatabaseError error)
{
                                            Toast.makeText(LoginActivity.this,
error.getMessage(), Toast.LENGTH_SHORT).show();
                                        }
                                    });
                                    animationView.setAnimation(R.raw.success_animation);
                                    message.setText(R.string.user_successful_login);
                                    positiveBtn.setText(R.string.ok);
                                    positiveBtn.setOnClickListener(view1 -> {
                                        dialog.dismiss();
                                        startActivity(new Intent(LoginActivity.this,
MainActivity.class));
                                        finish();
                                    });
                                }else {
                                    String error =
Objects.requireNonNull(task.getException()).toString();
                                    String[] separated = error.split(":");
```

```
                                    animationView.setAnimation(R.raw.failed_animation);
                                    message.setText(separated[1]);
                                    positiveBtn.setText(R.string.try_again);
                                    positiveBtn.setOnClickListener(view1 ->
dialog.dismiss());
                                }
                                dialog.show();
                            });
                }else {
                    progressDialog.dismiss();

                    animationView.setAnimation(R.raw.failed_animation);
                    message.setText(R.string.credential_fail);
                    positiveBtn.setText(R.string.try_again);
                    positiveBtn.setOnClickListener(view1 -> dialog.dismiss());
                }
            }
        });
    }

    private void setUIViews() {
        scrollView = findViewById(R.id.login_scrollview);
        backBtn = findViewById(R.id.login_back_btn);
        profileIcon = findViewById(R.id.login_profile_icon);
        title = findViewById(R.id.login_welcome_textview);
        subtitle = findViewById(R.id.login_subtitle_textview);
        forgotPassword = findViewById(R.id.login_forgot_password_textview);
        signup = findViewById(R.id.login_signup_btn);
        email = findViewById(R.id.login_email_textInputLayout);
        password = findViewById(R.id.login_password_textInputLayout);
        rememberMe = findViewById(R.id.login_remember_me_checkBox);
        login = findViewById(R.id.login_btn);
        newUserSignup = findViewById(R.id.login_new_user_signup_textview);
    }

    private void setAndLoadAnimations() {
        animLeft1 = AnimationUtils.loadAnimation(this, R.anim.anim_left_first);
        animLeft2 = AnimationUtils.loadAnimation(this, R.anim.anim_left_second);
        animRight1 = AnimationUtils.loadAnimation(this, R.anim.anim_right_first);
        animRight2 = AnimationUtils.loadAnimation(this, R.anim.anim_right_second);
        animTop1 = AnimationUtils.loadAnimation(this, R.anim.anim_top_first);
        animTop2 = AnimationUtils.loadAnimation(this, R.anim.anim_top_second);
        animBottom1 = AnimationUtils.loadAnimation(this, R.anim.anim_bottom_first);
        animBottom2 = AnimationUtils.loadAnimation(this, R.anim.anim_bottom_second);
        ranimLeft1 = AnimationUtils.loadAnimation(this, R.anim.ranim_left_first);
        ranimLeft2 = AnimationUtils.loadAnimation(this, R.anim.ranim_left_second);
        ranimRight1 = AnimationUtils.loadAnimation(this, R.anim.ranim_right_first);
        ranimRight2 = AnimationUtils.loadAnimation(this, R.anim.ranim_right_second);
        ranimTop1 = AnimationUtils.loadAnimation(this, R.anim.ranim_top_first);
        ranimTop2 = AnimationUtils.loadAnimation(this, R.anim.ranim_top_second);
        ranimBottom1 = AnimationUtils.loadAnimation(this, R.anim.ranim_bottom_first);
        ranimBottom2 = AnimationUtils.loadAnimation(this, R.anim.ranim_bottom_second);

        setAnimLeft.addAnimation(animLeft1);
        setAnimLeft.addAnimation(animLeft2);
```

```
        setAnimRight.addAnimation(animRight1);
        setAnimRight.addAnimation(animRight2);
        setAnimTop.addAnimation(animTop1);
        setAnimTop.addAnimation(animTop2);
        setAnimBottom.addAnimation(animBottom1);
        setAnimBottom.addAnimation(animBottom2);
        rsetAnimLeft.addAnimation(ranimLeft1);
        rsetAnimLeft.addAnimation(ranimLeft2);
        rsetAnimRight.addAnimation(ranimRight1);
        rsetAnimRight.addAnimation(ranimRight2);
        rsetAnimTop.addAnimation(ranimTop1);
        rsetAnimTop.addAnimation(ranimTop2);
        rsetAnimBottom.addAnimation(ranimBottom1);
        rsetAnimBottom.addAnimation(ranimBottom2);
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        backBtn.startAnimation(rsetAnimLeft);
        profileIcon.startAnimation(rsetAnimTop);
        title.startAnimation(rsetAnimLeft);
        subtitle.startAnimation(rsetAnimLeft);
        email.startAnimation(rsetAnimRight);
        password.startAnimation(rsetAnimLeft);
        rememberMe.startAnimation(rsetAnimLeft);
        forgotPassword.startAnimation(rsetAnimRight);
        login.startAnimation(rsetAnimBottom);
        newUserSignup.startAnimation(rsetAnimBottom);

        backBtnTimer.start();
    }
}
```

2. Register1Activity.java

```
package com.example.agrifarmer.activities.registration;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationSet;
import android.view.animation.AnimationUtils;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.ImageView;
import android.widget.LinearLayout;
```

```
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.agrifarmer.R;
import com.google.android.material.textfield.TextInputLayout;

public class Register1Activity extends AppCompatActivity {

    private ScrollView scrollView;
    private ImageView backBtn;
    private TextView title, next, login;
    private TextInputLayout fullname, email, phone, location;
    private AutoCompleteTextView autoCompleteLocations;
    private LinearLayout alreadyRegistered;

    private CountDownTimer backBtnTimer, loginBtnTimer;

    Animation animLeft1, animLeft2, animRight1, animRight2, animTop1, animTop2, animBottom1,
animBottom2;
    Animation ranimLeft1, ranimLeft2, ranimRight1, ranimRight2, ranimTop1, ranimTop2,
ranimBottom1, ranimBottom2;

    AnimationSet setAnimLeft = new AnimationSet(true);
    AnimationSet setAnimRight = new AnimationSet(true);
    AnimationSet setAnimTop = new AnimationSet(true);
    AnimationSet setAnimBottom = new AnimationSet(true);
    AnimationSet rsetAnimLeft = new AnimationSet(true);
    AnimationSet rsetAnimRight = new AnimationSet(true);
    AnimationSet rsetAnimTop = new AnimationSet(true);
    AnimationSet rsetAnimBottom = new AnimationSet(true);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register1);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setUIViews();
        setAndLoadAnimations();

        String[] locations = getResources().getStringArray(R.array.locations);
        ArrayAdapter<String> locationAdapter = new ArrayAdapter<>(this,
R.layout.dropdown_item, locations);
        autoCompleteLocations.setAdapter(locationAdapter);

        backBtnTimer = new CountDownTimer(1300, 100) {
            @Override
            public void onTick(long l) {}
            @Override
            public void onFinish() {
                backBtn.setVisibility(View.INVISIBLE);
                title.setVisibility(View.INVISIBLE);
```

```
                fullname.setVisibility(View.INVISIBLE);
                email.setVisibility(View.INVISIBLE);
                phone.setVisibility(View.INVISIBLE);
                location.setVisibility(View.INVISIBLE);
                next.setVisibility(View.INVISIBLE);
                alreadyRegistered.setVisibility(View.INVISIBLE);

                finish();
            }
        };

        loginBtnTimer = new CountDownTimer(1300, 100) {
            @Override
            public void onTick(long l) {}
            @Override
            public void onFinish() {
                backBtn.setVisibility(View.INVISIBLE);
                title.setVisibility(View.INVISIBLE);
                fullname.setVisibility(View.INVISIBLE);
                email.setVisibility(View.INVISIBLE);
                phone.setVisibility(View.INVISIBLE);
                location.setVisibility(View.INVISIBLE);
                next.setVisibility(View.INVISIBLE);
                alreadyRegistered.setVisibility(View.INVISIBLE);

                Intent intent = new Intent(Register1Activity.this, LoginActivity.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
                startActivity(intent);
                finish();
            }
        };

        backBtn.startAnimation(setAnimLeft);
        title.startAnimation(setAnimLeft);
        fullname.startAnimation(setAnimRight);
        email.startAnimation(setAnimLeft);
        phone.startAnimation(setAnimRight);
        location.startAnimation(setAnimLeft);
        next.startAnimation(setAnimRight);
        alreadyRegistered.startAnimation(setAnimBottom);

        backBtn.setOnClickListener(View -> onBackPressed());

        login.setOnClickListener(View -> {
            backBtn.startAnimation(rsetAnimLeft);
            title.startAnimation(rsetAnimLeft);
            fullname.startAnimation(rsetAnimRight);
            email.startAnimation(rsetAnimLeft);
            phone.startAnimation(rsetAnimRight);
            location.startAnimation(rsetAnimLeft);
            next.startAnimation(rsetAnimRight);
            alreadyRegistered.startAnimation(rsetAnimBottom);

            loginBtnTimer.start();
        });
```

```
        fullname.getEditText().addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
                String txtFullName = fullname.getEditText().getText().toString().trim();
                if (txtFullName.equals(""))
                    fullname.setError("Empty Credential!");
                else if (txtFullName.length() < 4)
                    fullname.setError("Credential too short!");
                else if (txtFullName.length() > 20)
                    fullname.setError("Credential too long!");
                else
                    fullname.setError(null);
            }
        });

        email.getEditText().addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
                String emailPattern = "[a-zA-Z0-9._-]+@[a-z]+\\.+[a-z]+";
                String txtEmail = email.getEditText().getText().toString().trim();
                if (txtEmail.equals(""))
                    email.setError("Empty Credential!");
                else if (!txtEmail.matches(emailPattern))
                    email.setError("Invalid Email!");
                else
                    email.setError(null);
            }
        });

        phone.getEditText().addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
                String phonePattern = "\\d{10}";
                String txtPhone = phone.getEditText().getText().toString().trim();
                if (txtPhone.equals(""))
                    phone.setError("Empty Credential!");
                else if (!txtPhone.matches(phonePattern))
                    phone.setError("Invalid Phone Number!");
                else
```

```
                    phone.setError(null);
                }
            });

            autoCompleteLocations.setOnItemClickListener((adapterView, view, i, l) ->
location.setError(null));

            next.setOnClickListener(view -> {
                String txtFullName = fullname.getEditText().getText().toString().trim();
                String txtEmail = email.getEditText().getText().toString().trim();
                String txtPhone = phone.getEditText().getText().toString().trim();
                String txtLocation = autoCompleteLocations.getText().toString().trim();

                if (txtFullName.equals("") || txtEmail.equals("") || txtPhone.equals("") ||
txtLocation.equals("")) {
                    if (txtFullName.equals(""))
                        fullname.setError("Empty Credential!");
                    if (txtEmail.equals(""))
                        email.setError("Empty Credential!");
                    if (txtPhone.equals(""))
                        phone.setError("Empty Credential!");
                    if (txtLocation.equals(""))
                        location.setError("Please Select Location!");
                }else {
                    if (fullname.getError() == null && email.getError() == null &&
phone.getError() == null && location.getError() == null) {
                        CountDownTimer timer = new CountDownTimer(1300, 100) {
                            @Override
                            public void onTick(long l) {}
                            @Override
                            public void onFinish() {
                                backBtn.setVisibility(View.INVISIBLE);
                                title.setVisibility(View.INVISIBLE);
                                fullname.setVisibility(View.INVISIBLE);
                                email.setVisibility(View.INVISIBLE);
                                phone.setVisibility(View.INVISIBLE);
                                location.setVisibility(View.INVISIBLE);
                                next.setVisibility(View.INVISIBLE);
                                alreadyRegistered.setVisibility(View.INVISIBLE);

                                Intent intent = new Intent(Register1Activity.this,
Register2Activity.class);
                                intent.setFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
                                intent.putExtra("FullName", txtFullName);
                                intent.putExtra("Email", txtEmail);
                                intent.putExtra("Phone", txtPhone);
                                intent.putExtra("Location", txtLocation);

                                startActivity(intent);
                            }
                        };

                        backBtn.startAnimation(rsetAnimLeft);
                        title.startAnimation(rsetAnimLeft);
                        fullname.startAnimation(rsetAnimRight);
```

```
                    email.startAnimation(rsetAnimLeft);
                    phone.startAnimation(rsetAnimRight);
                    location.startAnimation(rsetAnimLeft);
                    next.startAnimation(rsetAnimRight);
                    alreadyRegistered.startAnimation(rsetAnimBottom);

                    timer.start();
                }else {
                    //Add dialog here and after that remove below toast
                    Toast.makeText(this, "Failed to Register! Try Again.",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private void setUIViews() {
        scrollView = findViewById(R.id.register1_scrollview);
        backBtn = findViewById(R.id.register1_back_btn);
        title = findViewById(R.id.register1_create_account_textview);
        next = findViewById(R.id.register1_next_btn);
        login = findViewById(R.id.register1_login_btn);
        fullname = findViewById(R.id.register1_fullname_textInputLayout);
        email = findViewById(R.id.register1_email_textInputLayout);
        phone = findViewById(R.id.register1_phone_textInputLayout);
        location = findViewById(R.id.register1_locations_textInputLayout);
        autoCompleteLocations =
findViewById(R.id.register1_locations_autocomplete_textview);
        alreadyRegistered = findViewById(R.id.register1_already_registered_login_textview);
    }

    private void setAndLoadAnimations() {
        animLeft1 = AnimationUtils.loadAnimation(this, R.anim.anim_left_first);
        animLeft2 = AnimationUtils.loadAnimation(this, R.anim.anim_left_second);
        animRight1 = AnimationUtils.loadAnimation(this, R.anim.anim_right_first);
        animRight2 = AnimationUtils.loadAnimation(this, R.anim.anim_right_second);
        animTop1 = AnimationUtils.loadAnimation(this, R.anim.anim_top_first);
        animTop2 = AnimationUtils.loadAnimation(this, R.anim.anim_top_second);
        animBottom1 = AnimationUtils.loadAnimation(this, R.anim.anim_bottom_first);
        animBottom2 = AnimationUtils.loadAnimation(this, R.anim.anim_bottom_second);
        ranimLeft1 = AnimationUtils.loadAnimation(this, R.anim.ranim_left_first);
        ranimLeft2 = AnimationUtils.loadAnimation(this, R.anim.ranim_left_second);
        ranimRight1 = AnimationUtils.loadAnimation(this, R.anim.ranim_right_first);
        ranimRight2 = AnimationUtils.loadAnimation(this, R.anim.ranim_right_second);
        ranimTop1 = AnimationUtils.loadAnimation(this, R.anim.ranim_top_first);
        ranimTop2 = AnimationUtils.loadAnimation(this, R.anim.ranim_top_second);
        ranimBottom1 = AnimationUtils.loadAnimation(this, R.anim.ranim_bottom_first);
        ranimBottom2 = AnimationUtils.loadAnimation(this, R.anim.ranim_bottom_second);

        setAnimLeft.addAnimation(animLeft1);
        setAnimLeft.addAnimation(animLeft2);
        setAnimRight.addAnimation(animRight1);
        setAnimRight.addAnimation(animRight2);
        setAnimTop.addAnimation(animTop1);
        setAnimTop.addAnimation(animTop2);
```

```
        setAnimBottom.addAnimation(animBottom1);
        setAnimBottom.addAnimation(animBottom2);
        rsetAnimLeft.addAnimation(ranimLeft1);
        rsetAnimLeft.addAnimation(ranimLeft2);
        rsetAnimRight.addAnimation(ranimRight1);
        rsetAnimRight.addAnimation(ranimRight2);
        rsetAnimTop.addAnimation(ranimTop1);
        rsetAnimTop.addAnimation(ranimTop2);
        rsetAnimBottom.addAnimation(ranimBottom1);
        rsetAnimBottom.addAnimation(ranimBottom2);
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        backBtn.startAnimation(rsetAnimLeft);
        title.startAnimation(rsetAnimLeft);
        fullname.startAnimation(rsetAnimRight);
        email.startAnimation(rsetAnimLeft);
        phone.startAnimation(rsetAnimRight);
        location.startAnimation(rsetAnimLeft);
        next.startAnimation(rsetAnimRight);
        alreadyRegistered.startAnimation(rsetAnimBottom);

        backBtnTimer.start();
    }

    @Override
    protected void onResume() {
        super.onResume();
        backBtn.setVisibility(View.VISIBLE);
        title.setVisibility(View.VISIBLE);
        fullname.setVisibility(View.VISIBLE);
        email.setVisibility(View.VISIBLE);
        phone.setVisibility(View.VISIBLE);
        location.setVisibility(View.VISIBLE);
        next.setVisibility(View.VISIBLE);
        alreadyRegistered.setVisibility(View.VISIBLE);
    }
}
```

3. Register2Activity.java

```
package com.example.agrifarmer.activities.registration;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.text.Editable;
```

```
import android.text.TextWatcher;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationSet;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;

import com.airbnb.lottie.LottieAnimationView;
import com.example.agrifarmer.R;
import com.example.agrifarmer.activities.MainActivity;
import com.example.agrifarmer.getterSetterClasses.Profile;
import com.example.agrifarmer.localDatabases.MyProfileDbHandler;
import com.google.android.material.checkbox.MaterialCheckBox;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.FirebaseDatabase;

import java.util.HashMap;
import java.util.Objects;

public class Register2Activity extends AppCompatActivity {

    private ScrollView scrollView;
    private ImageView backBtn;
    private TextView title, register;
    private TextInputLayout enterPass, confirmPass, referralCode;
    private MaterialCheckBox haveReferralCode;

    private CountDownTimer backBtnTimer;
    private ProgressDialog progressDialog;
    private Dialog dialog;

    private String txtFullName, txtEmail, txtPhone, txtLocation;

    private FirebaseAuth auth;

    private MyProfileDbHandler profileDbHandler = new
MyProfileDbHandler(Register2Activity.this);

    Animation animLeft1, animLeft2, animRight1, animRight2, animTop1, animTop2, animBottom1,
animBottom2;
    Animation ranimLeft1, ranimLeft2, ranimRight1, ranimRight2, ranimTop1, ranimTop2,
ranimBottom1, ranimBottom2;

    AnimationSet setAnimLeft = new AnimationSet(true);
    AnimationSet setAnimRight = new AnimationSet(true);
    AnimationSet setAnimTop = new AnimationSet(true);
    AnimationSet setAnimBottom = new AnimationSet(true);
    AnimationSet rsetAnimLeft = new AnimationSet(true);
```

```
    AnimationSet rsetAnimRight = new AnimationSet(true);
    AnimationSet rsetAnimTop = new AnimationSet(true);
    AnimationSet rsetAnimBottom = new AnimationSet(true);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register2);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setUIViews();
        setAndLoadAnimations();

        auth = FirebaseAuth.getInstance();

        Intent intent = getIntent();
        txtFullName = intent.getStringExtra("FullName");
        txtEmail = intent.getStringExtra("Email");
        txtPhone = intent.getStringExtra("Phone");
        txtLocation = intent.getStringExtra("Location");

        backBtnTimer = new CountDownTimer(1300, 100) {
            @Override
            public void onTick(long l) {}
            @Override
            public void onFinish() {
                backBtn.setVisibility(View.INVISIBLE);
                title.setVisibility(View.INVISIBLE);
                enterPass.setVisibility(View.INVISIBLE);
                confirmPass.setVisibility(View.INVISIBLE);
                haveReferralCode.setVisibility(View.INVISIBLE);
                register.setVisibility(View.INVISIBLE);

                finish();
            }
        };

        backBtn.startAnimation(setAnimLeft);
        title.startAnimation(setAnimLeft);
        enterPass.startAnimation(setAnimRight);
        confirmPass.startAnimation(setAnimLeft);
        haveReferralCode.startAnimation(setAnimLeft);
        register.startAnimation(setAnimBottom);

        enterPass.getEditText().addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
```

```
                String passwordPattern =
"^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&*+=()])(?=\\S+$).{8,20}$";
                String txtEnterPass = enterPass.getEditText().getText().toString().trim();
                if (txtEnterPass.equals(""))
                    enterPass.setError("Empty Credential!");
                else if (!txtEnterPass.matches(passwordPattern))
                    enterPass.setError("Doesn't meet all conditions!");
                else
                    enterPass.setError(null);
            }
        });

        confirmPass.getEditText().addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
                String passwordPattern =
"^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&*+=()])(?=\\S+$).{8,20}$";
                String txtEnterP = enterPass.getEditText().getText().toString().trim();
                String txtConfirmP = confirmPass.getEditText().getText().toString().trim();
                if (!txtConfirmP.equals(txtEnterP))
                    confirmPass.setError("Password doesn't match!");
                else if (!txtConfirmP.matches(passwordPattern))
                    confirmPass.setError("Doesn't meet all conditions!");
                else
                    confirmPass.setError(null);
            }
        });

        backBtn.setOnClickListener(view -> onBackPressed());

        haveReferralCode.setOnClickListener(view -> {
            if (haveReferralCode.isChecked())
                referralCode.setVisibility(View.VISIBLE);
            else
                referralCode.setVisibility(View.INVISIBLE);
        });

        register.setOnClickListener(view -> {
            progressDialog = new ProgressDialog(Register2Activity.this);
            progressDialog.setCancelable(false);

            String enterP = enterPass.getEditText().getText().toString().trim();
            String confirmP = confirmPass.getEditText().getText().toString().trim();

            if (enterP.equals("") || confirmP.equals("")) {
                if (enterP.equals(""))
                    enterPass.setError("Empty Credential!");
                if (confirmP.equals(""))
                    confirmPass.setError("EmptyCredential!");
            }else{
```

```
                progressDialog.setMessage("Registering...");
                progressDialog.show();

                dialog = new Dialog(Register2Activity.this);
                dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
                dialog.setContentView(R.layout.message_dialog);
                dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
                dialog.setCancelable(false);

                LottieAnimationView animationView =
dialog.findViewById(R.id.message_lottie_animation);
                TextView message = dialog.findViewById(R.id.message_textview);
                Button positiveBtn = dialog.findViewById(R.id.message_positive_btn);
                Button negativeBtn = dialog.findViewById(R.id.message_negative_btn);
                negativeBtn.setVisibility(View.GONE);

                if (enterPass.getError() == null && confirmPass.getError() == null) {
                    auth.createUserWithEmailAndPassword(txtEmail, confirmP)
                            .addOnCompleteListener(this, task -> {
                                progressDialog.dismiss();

                                if (task.isSuccessful()) {
                                    HashMap<String, Object> map = new HashMap<>();
                                    map.put("fullname", txtFullName);
                                    map.put("email", txtEmail);
                                    map.put("phone", txtPhone);
                                    map.put("location", txtLocation);
                                    Toast.makeText(this, auth.getUid(),
Toast.LENGTH_SHORT).show();

FirebaseDatabase.getInstance().getReference().child("UserInfo").child(auth.getUid()).updateC
hildren(map);

                                    Profile profile = new Profile();
                                    profile.setEmail(txtEmail);
                                    profile.setFullname(txtFullName);
                                    profile.setLocation(txtLocation);
                                    profile.setPhone(txtPhone);
                                    profileDbHandler.addUser(profile);

                                    animationView.setAnimation(R.raw.success_animation);
                                    message.setText(R.string.user_successful_login);
                                    positiveBtn.setText(R.string.ok);
                                    positiveBtn.setOnClickListener(view1 -> {
                                        dialog.dismiss();
                                        startActivity(new Intent(Register2Activity.this,
MainActivity.class));
                                        finish();
                                    });
                                    dialog.show();
                                }else {
                                    String error =
Objects.requireNonNull(task.getException()).toString();
                                    String[] separated = error.split(":");
```

```
                                    animationView.setAnimation(R.raw.failed_animation);
                                    message.setText(separated[1]);
                                    positiveBtn.setText(R.string.try_again);
                                    positiveBtn.setOnClickListener(view1 ->
dialog.dismiss());
                                    dialog.show();
                                }
                            });
                }else {
                    progressDialog.dismiss();

                    animationView.setAnimation(R.raw.failed_animation);
                    message.setText(R.string.credential_fail);
                    positiveBtn.setText(R.string.try_again);
                    positiveBtn.setOnClickListener(view1 -> dialog.dismiss());
                }
            }
        });
    }

    private void setUIViews() {
        scrollView = findViewById(R.id.register2_scrollview);
        backBtn = findViewById(R.id.register2_back_btn);
        title = findViewById(R.id.register2_create_password_textview);
        register = findViewById(R.id.register2_register_btn);
        enterPass = findViewById(R.id.register2_enter_password_textInputLayout);
        confirmPass = findViewById(R.id.register2_confirm_password_textInputLayout);
        referralCode = findViewById(R.id.register2_referral_code_textInputLayout);
        haveReferralCode = findViewById(R.id.register2_referral_code_checkbox);
    }

    private void setAndLoadAnimations() {
        animLeft1 = AnimationUtils.loadAnimation(this, R.anim.anim_left_first);
        animLeft2 = AnimationUtils.loadAnimation(this, R.anim.anim_left_second);
        animRight1 = AnimationUtils.loadAnimation(this, R.anim.anim_right_first);
        animRight2 = AnimationUtils.loadAnimation(this, R.anim.anim_right_second);
        animTop1 = AnimationUtils.loadAnimation(this, R.anim.anim_top_first);
        animTop2 = AnimationUtils.loadAnimation(this, R.anim.anim_top_second);
        animBottom1 = AnimationUtils.loadAnimation(this, R.anim.anim_bottom_first);
        animBottom2 = AnimationUtils.loadAnimation(this, R.anim.anim_bottom_second);
        ranimLeft1 = AnimationUtils.loadAnimation(this, R.anim.ranim_left_first);
        ranimLeft2 = AnimationUtils.loadAnimation(this, R.anim.ranim_left_second);
        ranimRight1 = AnimationUtils.loadAnimation(this, R.anim.ranim_right_first);
        ranimRight2 = AnimationUtils.loadAnimation(this, R.anim.ranim_right_second);
        ranimTop1 = AnimationUtils.loadAnimation(this, R.anim.ranim_top_first);
        ranimTop2 = AnimationUtils.loadAnimation(this, R.anim.ranim_top_second);
        ranimBottom1 = AnimationUtils.loadAnimation(this, R.anim.ranim_bottom_first);
        ranimBottom2 = AnimationUtils.loadAnimation(this, R.anim.ranim_bottom_second);

        setAnimLeft.addAnimation(animLeft1);
        setAnimLeft.addAnimation(animLeft2);
        setAnimRight.addAnimation(animRight1);
        setAnimRight.addAnimation(animRight2);
        setAnimTop.addAnimation(animTop1);
```

```
        setAnimTop.addAnimation(animTop2);
        setAnimBottom.addAnimation(animBottom1);
        setAnimBottom.addAnimation(animBottom2);
        rsetAnimLeft.addAnimation(ranimLeft1);
        rsetAnimLeft.addAnimation(ranimLeft2);
        rsetAnimRight.addAnimation(ranimRight1);
        rsetAnimRight.addAnimation(ranimRight2);
        rsetAnimTop.addAnimation(ranimTop1);
        rsetAnimTop.addAnimation(ranimTop2);
        rsetAnimBottom.addAnimation(ranimBottom1);
        rsetAnimBottom.addAnimation(ranimBottom2);
    }
}
```

4. PredictCropsActivity.java

```
package com.example.agrifarmer.activities;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Dialog;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.airbnb.lottie.LottieAnimationView;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.RequestFuture;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.example.agrifarmer.R;
import com.example.agrifarmer.getterSetterClasses.CropRecord;
import com.example.agrifarmer.localDatabases.CropRecordDbHandler;
import com.example.agrifarmer.localDatabases.MyProfileDbHandler;
import com.google.android.material.datepicker.MaterialDatePicker;
import com.google.android.material.datepicker.MaterialPickerOnPositiveButtonClickListener;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;

import org.json.JSONArray;
```

```
import org.json.JSONException;
import org.json.JSONObject;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;

public class PredictCropsActivity extends AppCompatActivity {

    private TextInputLayout startDate, duration, soilPH;
    private TextInputEditText editStartDate, editSoilPH;
    private AutoCompleteTextView autoDuration;
    private TextView startFetch, startPrediction;
    private LottieAnimationView fetchAnimation;

    Dialog dialog;

    ArrayList<String> crops = new ArrayList<>();

    private String urlPredictCrop = "https://agri-farmer-api.herokuapp.com/predictCrop";

    private MyProfileDbHandler profileDbHandler = new
MyProfileDbHandler(PredictCropsActivity.this);
    private CropRecordDbHandler recordDbHandler = new
CropRecordDbHandler(PredictCropsActivity.this);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_predict_crops);

        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setUIViews();

        HashMap<String, String> months = new HashMap<>();
        months.put("Jan", "01");
        months.put("Feb", "02");
        months.put("Mar", "03");
        months.put("Apr", "04");
        months.put("May", "05");
        months.put("Jun", "06");
        months.put("Jul", "07");
        months.put("Aug", "08");
        months.put("Sep", "09");
```

```
        months.put("Oct", "10");
        months.put("Nov", "11");
        months.put("Dec", "12");

        String[] duration_in_months =
getResources().getStringArray(R.array.duration_in_months);
        ArrayAdapter<String> durationAdapter = new ArrayAdapter<>(this,
R.layout.dropdown_item, duration_in_months);
        autoDuration.setAdapter(durationAdapter);

        MaterialDatePicker.Builder<Long> dateBuilder =
MaterialDatePicker.Builder.datePicker();
        dateBuilder.setTitleText("Select Start Date");

        final MaterialDatePicker materialDatePicker = dateBuilder.build();

        editStartDate.setOnClickListener(view ->
materialDatePicker.show(getSupportFragmentManager(), "MATERIAL_DATE_PICKER"));

        materialDatePicker.addOnPositiveButtonClickListener(selection ->
editStartDate.setText(materialDatePicker.getHeaderText()));

        /*
        startDate.getEditText().addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
                DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd-MM-yyyy");
                LocalDateTime now = LocalDateTime.now();
                String today = dtf.format(now);

                SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
                try {
                    Date d1 = format.parse(today);
                    Date d2 = format.parse(startDate.getEditText().getText().toString());
                    if (d1.compareTo(d2) > 0)
                        startDate.setError("Please select future date!!");
                    else
                        startDate.setError(null);
                } catch (ParseException e) {
                    e.printStackTrace();
                }
            }
        });
        */

        autoDuration.setOnItemClickListener((adapterView, view, i, l) ->
duration.setError(null));

        startFetch.setOnClickListener(View -> {
            String startingDate = startDate.getEditText().getText().toString();
```

```
            String strDuration = duration.getEditText().getText().toString();

            if (startingDate.equals("") || strDuration.equals("")) {
                if (startingDate.equals(""))
                    startDate.setError("Empty Credential!");
                if (strDuration.equals(""))
                    duration.setError("Empty Credential!");
            }else {
                if (startDate.getError() == null && duration.getError() == null) {
                    startFetch.setVisibility(android.view.View.GONE);
                    fetchAnimation.setVisibility(android.view.View.VISIBLE);
                    fetchAnimation.playAnimation();
                }else {
                    Toast.makeText(this, "Please enter valid credentials!!",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        soilPH.getEditText().addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2)
{}
            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {}
            @Override
            public void afterTextChanged(Editable editable) {
                String ph = soilPH.getEditText().getText().toString();
                if (ph.equals("")) {
                    soilPH.setError("Empty Credential!");
                }else {
                    float fph = Float.parseFloat(ph);
                    if (fph < 4.0f || fph > 12.0)
                        soilPH.setError("Please enter valid value!");
                    else
                        soilPH.setError(null);
                }
            }
        });

        startPrediction.setOnClickListener(View -> {
            if (startFetch.getVisibility() == android.view.View.GONE){
                String[] date = startDate.getEditText().getText().toString().split(" ");
                String finalDate = date[0] + "-" + months.get(date[1]);
                String finalDuration = autoDuration.getText().toString();
                String finalPH = soilPH.getEditText().getText().toString();
                String finalLocation =
profileDbHandler.getRegisteredUser().getLocation().toLowerCase(Locale.ROOT);

                Toast.makeText(PredictCropsActivity.this, finalPH + finalDate + finalDuration
+ finalLocation, Toast.LENGTH_LONG).show();

                dialog = new Dialog(PredictCropsActivity.this);
                dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
                dialog.setContentView(R.layout.message_dialog);
```

```
                dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
                dialog.setCancelable(false);

                LottieAnimationView animationView =
dialog.findViewById(R.id.message_lottie_animation);
                TextView message = dialog.findViewById(R.id.message_textview);
                Button positiveBtn = dialog.findViewById(R.id.message_positive_btn);
                Button negativeBtn = dialog.findViewById(R.id.message_negative_btn);

                animationView.setAnimation(R.raw.corn_growing_animation);
                message.setText(R.string.processing);
                positiveBtn.setVisibility(android.view.View.GONE);
                negativeBtn.setVisibility(android.view.View.GONE);

                CountDownTimer timer = new CountDownTimer(5000, 1000) {
                    @Override
                    public void onTick(long l) {}
                    @Override
                    public void onFinish() {
                        animationView.setAnimation(R.raw.farmer_animation);
                        message.setText(R.string.process_complete);
                        positiveBtn.setText(R.string.show_results);
                        positiveBtn.setVisibility(android.view.View.VISIBLE);
                        positiveBtn.setOnClickListener(view -> {
                            dialog.dismiss();
                            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd
HH:mm:ss");
                            LocalDateTime now = LocalDateTime.now();
                            CropRecord cropRecord = new CropRecord();
                            cropRecord.setCrop1(crops.get(0));
                            cropRecord.setCrop2(crops.get(1));
                            cropRecord.setCrop3(crops.get(2));

cropRecord.setLocation(profileDbHandler.getRegisteredUser().getLocation());
                            cropRecord.setDuration(finalDuration);
                            cropRecord.setDatetime(String.valueOf(dtf.format(now)));
                            recordDbHandler.addRecord(cropRecord);

                            Intent intent = new Intent(PredictCropsActivity.this,
ShowResultActivity.class);
                            intent.putExtra("location",
profileDbHandler.getRegisteredUser().getLocation());
                            intent.putExtra("crop1", crops.get(0));
                            intent.putExtra("crop2", crops.get(1));
                            intent.putExtra("crop3", crops.get(2));
                            intent.putExtra("duration", finalDuration);
                            startActivity(intent);
                            finish();
                        });
                    }
                };

                StringRequest request = new StringRequest(Request.Method.POST,
urlPredictCrop,
```

```
                        response -> {
                            try {
                                Toast.makeText(this, "Inside Try",
Toast.LENGTH_SHORT).show();
                                JSONObject jsonObject = new JSONObject(response);
                                dialog.show();
                                JSONArray array = jsonObject.getJSONArray("Suitable Crop");
                                for (int i = 0; i < array.length(); i++) {
                                    crops.add(String.valueOf(array.get(i)));
                                }
                                timer.start();
                            }catch(JSONException e){
                                e.printStackTrace();
                                Toast.makeText(this, "Catch Error",
Toast.LENGTH_SHORT).show();
                            }
                        },
                        error -> {
                            Toast.makeText(this, "error", Toast.LENGTH_SHORT).show();
                        }){
                    @Override
                    protected Map<String, String> getParams() {
                        Map<String, String> params = new HashMap<>();
                        params.put("ph", finalPH);
                        params.put("startDate", finalDate);
                        params.put("duration", finalDuration);
                        params.put("location",
profileDbHandler.getRegisteredUser().getLocation().toLowerCase());
                        return params;
                    }
                };
                RequestQueue queue = Volley.newRequestQueue(PredictCropsActivity.this);
                queue.add(request);
            }else {
                Toast.makeText(this, "Please get weather data first!",
Toast.LENGTH_SHORT).show();
            }

            //Get all the values (startDate, duration, pH) and make a post call to api
            //Until we get output show crop growing animation as loading screen
            //Once output is fetched then go to results activity and show the result
            //Also save this record in local database
            //Show in recent activity on main page
            //Also show in crop history activity.
        });
    }

    private void setUIViews() {
        startDate = findViewById(R.id.predict_start_date_textInputLayout);
        duration = findViewById(R.id.predict_duration_textInputLayout);
        soilPH = findViewById(R.id.predict_soilPH_textInputLayout);
        editStartDate = findViewById(R.id.predict_start_date_textInputEditText);
        editSoilPH = findViewById(R.id.predict_soilPH_textInputEditText);
        autoDuration = findViewById(R.id.predict_duration_autoCompleteTextView);
        startFetch = findViewById(R.id.predict_fetch_weather_data_btn);
```

```
        startPrediction = findViewById(R.id.predict_start_prediction_btn);
        fetchAnimation = findViewById(R.id.predict_weather_fetch_complete_animation);
    }
}
```

Weather API

1. app.py

```
from flask import Flask, request, jsonify
from urllib.request import urlopen
from apscheduler.schedulers.background import BackgroundScheduler
from collections import Counter
import json
import pickle
import pandas as pd
import numpy as np

model = pickle.load(open('model.pkl', 'rb'))
scaler = pickle.load(open('scaler.pkl', 'rb'))


def get_weather_conditions_from_api():
    kolhapur_url =
"https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/kolhapu
r/today?unitGroup=metric&key=PNSZ5F63HRRRB3P67KPBWMLDM&contentType=json"
    sangli_url =
"https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/sangli/
today?unitGroup=metric&key=PNSZ5F63HRRRB3P67KPBWMLDM&contentType=json"
    nashik_url =
"https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/nashik/
today?unitGroup=metric&key=PNSZ5F63HRRRB3P67KPBWMLDM&contentType=json"

    kolhapur_response = urlopen(kolhapur_url)
    sangli_response = urlopen(sangli_url)
    nashik_response = urlopen(nashik_url)

    kolhapur_data = json.loads(kolhapur_response.read())
    sangli_data = json.loads(sangli_response.read())
    nashik_data = json.loads(nashik_response.read())

    kolhapur_json_obj = json.dumps(kolhapur_data, indent=4)
    sangli_json_obj = json.dumps(sangli_data, indent=4)
    nashik_json_obj = json.dumps(nashik_data, indent=4)

    with open("Kolhapur_weather_data.json", "w") as outfile:
        outfile.write(kolhapur_json_obj)

    with open("Sangli_weather_data.json", "w") as outfile:
        outfile.write(sangli_json_obj)
```

```
    with open("Nashik_weather_data.json", "w") as outfile:
        outfile.write(nashik_json_obj)


scheduler = BackgroundScheduler(daemon=True)
scheduler.add_job(get_weather_conditions_from_api, 'interval', hours=24)
scheduler.start()

app = Flask(__name__)


@app.route('/currentWeather', methods=['POST'])
def get_current_weather_conditions():
    location = request.form.get('location')
    hr = int(request.form.get('hour'))

    file = None

    if location == 'kolhapur':
        file = open('Kolhapur_weather_data.json')
    elif location == 'sangli':
        file = open('Sangli_weather_data.json')
    elif location == 'nashik':
        file = open('Nashik_weather_data.json')

    data = json.load(file)

    result = dict()
    result['latitude'] = data['latitude']
    result['longitude'] = data['longitude']
    result['address'] = data['resolvedAddress']
    result['description'] = data['description']
    result['temp'] = data['days'][0]['hours'][hr]['temp']
    result['humidity'] = data['days'][0]['hours'][hr]['humidity']
    result['precip'] = data['days'][0]['hours'][hr]['precip']
    result['windspeed'] = data['days'][0]['hours'][hr]['windspeed']
    result['conditions'] = data['days'][0]['hours'][hr]['conditions']

    return jsonify(result)


@app.route('/predictCrop', methods=['POST'])
def predict_crop():
    ph = request.form.get('ph')
    start_date = request.form.get('startDate')
    duration = int(request.form.get('duration'))
    location = request.form.get('location')

    predictions = []

    df = pd.read_csv(location + '.csv')
    start_index = int(df.loc[df['date'] == start_date]['index'])
    for i in range(start_index, (start_index + (duration * 30))):
        index = i % 365
```

```
        input_query = np.array([[df.iloc[index]['temp'], df.iloc[index]['humidity'], ph,
df.iloc[index]['precip']]])
        input_query = scaler.transform(input_query)
        predictions.append(model.predict(input_query)[0])

    counts = Counter(predictions)
    counts = sorted(counts.items(), key=lambda kv: (kv[1], kv[0]), reverse=True)
    suitable_crops = [count[0] for count in counts]

    return jsonify({'Suitable Crop': suitable_crops[:3]})


if __name__ == '__main__':
    app.run(debug=True)
```

2. Kolhapur_Weather_Data.json


```
{
    "queryCost": 1,
    "latitude": 16.7044,
    "longitude": 74.2414,
    "resolvedAddress": "Kolhapur, MH, India",
    "address": "kolhapur",
    "timezone": "Asia/Kolkata",
    "tzoffset": 5.5,
    "description": "Similar temperatures continuing with no rain expected.",
    "days": [
        {
            "datetime": "2022-05-08",
            "datetimeEpoch": 1651948200,
            "tempmax": 39.1,
            "tempmin": 22.3,
            "temp": 28.7,
            "feelslikemax": 36.8,
            "feelslikemin": 22.3,
            "feelslike": 28.3,
            "dew": 15.7,
            "humidity": 53.7,
            "precip": 0.0,
            "precipprob": 0.0,
            "precipcover": 0.0,
            "preciptype": null,
            "snow": 0.0,
            "snowdepth": 0.0,
            "windgust": 30.6,
            "windspeed": 24.5,
            "winddir": 268.3,
            "pressure": 1005.7,
            "cloudcover": 43.7,
            "visibility": 24.1,
            "solarradiation": 237.4,
            "solarenergy": 20.5,
            "uvindex": 10.0,
            "severerisk": 15.0,
```

```
            "sunrise": "06:04:49",
            "sunriseEpoch": 1651970089,
            "sunset": "18:54:32",
            "sunsetEpoch": 1652016272,
            "moonphase": 0.22,
            "conditions": "Partially cloudy",
            "description": "Becoming cloudy in the afternoon.",
            "icon": "partly-cloudy-day",
            "stations": [
                "remote"
            ],
            "source": "comb",
            "hours": [
                {
                    "datetime": "00:00:00",
                    "datetimeEpoch": 1651948200,
                    "temp": 22.7,
                    "feelslike": 22.7,
                    "humidity": 86.87,
                    "dew": 20.4,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 15.8,
                    "windspeed": 9.0,
                    "winddir": 274.1,
                    "pressure": 1008.0,
                    "visibility": 24.1,
                    "cloudcover": 5.2,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Clear",
                    "icon": "clear-night",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "01:00:00",
                    "datetimeEpoch": 1651951800,
                    "temp": 22.4,
                    "feelslike": 22.4,
                    "humidity": 86.84,
                    "dew": 20.1,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 10.1,
```

```
                    "windspeed": 6.8,
                    "winddir": 280.2,
                    "pressure": 1008.0,
                    "visibility": 24.1,
                    "cloudcover": 5.2,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Clear",
                    "icon": "clear-night",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "02:00:00",
                    "datetimeEpoch": 1651955400,
                    "temp": 22.3,
                    "feelslike": 22.3,
                    "humidity": 85.23,
                    "dew": 19.7,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 7.2,
                    "windspeed": 6.1,
                    "winddir": 273.4,
                    "pressure": 1007.0,
                    "visibility": 24.1,
                    "cloudcover": 19.6,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Clear",
                    "icon": "clear-night",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "03:00:00",
                    "datetimeEpoch": 1651959000,
                    "temp": 22.4,
                    "feelslike": 22.4,
                    "humidity": 79.6,
                    "dew": 18.7,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
```

```
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 7.9,
                    "windspeed": 6.8,
                    "winddir": 277.5,
                    "pressure": 1006.0,
                    "visibility": 24.1,
                    "cloudcover": 86.9,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-night",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "04:00:00",
                    "datetimeEpoch": 1651962600,
                    "temp": 23.6,
                    "feelslike": 23.6,
                    "humidity": 69.08,
                    "dew": 17.6,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 11.5,
                    "windspeed": 7.9,
                    "winddir": 292.7,
                    "pressure": 1006.0,
                    "visibility": 24.1,
                    "cloudcover": 89.6,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-night",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "05:00:00",
                    "datetimeEpoch": 1651966200,
                    "temp": 23.3,
                    "feelslike": 23.3,
                    "humidity": 66.87,
                    "dew": 16.8,
```

```
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 7.2,
                    "windspeed": 6.5,
                    "winddir": 269.0,
                    "pressure": 1006.0,
                    "visibility": 24.1,
                    "cloudcover": 48.4,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-night",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "06:00:00",
                    "datetimeEpoch": 1651969800,
                    "temp": 22.8,
                    "feelslike": 22.8,
                    "humidity": 66.34,
                    "dew": 16.2,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 6.5,
                    "windspeed": 6.5,
                    "winddir": 242.9,
                    "pressure": 1006.0,
                    "visibility": 24.1,
                    "cloudcover": 9.3,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Clear",
                    "icon": "clear-night",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "07:00:00",
                    "datetimeEpoch": 1651973400,
                    "temp": 22.5,
```

```
                    "feelslike": 22.5,
                    "humidity": 67.56,
                    "dew": 16.2,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 5.0,
                    "windspeed": 5.0,
                    "winddir": 238.5,
                    "pressure": 1007.0,
                    "visibility": 24.1,
                    "cloudcover": 5.3,
                    "solarradiation": 5.0,
                    "solarenergy": 0.0,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Clear",
                    "icon": "clear-day",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "08:00:00",
                    "datetimeEpoch": 1651977000,
                    "temp": 25.9,
                    "feelslike": 25.9,
                    "humidity": 52.01,
                    "dew": 15.3,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 6.1,
                    "windspeed": 5.8,
                    "winddir": 253.8,
                    "pressure": 1008.0,
                    "visibility": 24.1,
                    "cloudcover": 5.0,
                    "solarradiation": 138.0,
                    "solarenergy": 0.5,
                    "uvindex": 1.0,
                    "severerisk": 5.0,
                    "conditions": "Clear",
                    "icon": "clear-day",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
```

```
                    "datetime": "09:00:00",
                    "datetimeEpoch": 1651980600,
                    "temp": 30.1,
                    "feelslike": 29.1,
                    "humidity": 34.16,
                    "dew": 12.6,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 6.5,
                    "windspeed": 6.5,
                    "winddir": 277.1,
                    "pressure": 1008.0,
                    "visibility": 24.1,
                    "cloudcover": 4.6,
                    "solarradiation": 376.0,
                    "solarenergy": 1.4,
                    "uvindex": 4.0,
                    "severerisk": 3.0,
                    "conditions": "Clear",
                    "icon": "clear-day",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "10:00:00",
                    "datetimeEpoch": 1651984200,
                    "temp": 33.7,
                    "feelslike": 32.0,
                    "humidity": 24.09,
                    "dew": 10.4,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 11.5,
                    "windspeed": 5.4,
                    "winddir": 309.6,
                    "pressure": 1008.0,
                    "visibility": 24.1,
                    "cloudcover": 9.2,
                    "solarradiation": 609.0,
                    "solarenergy": 2.2,
                    "uvindex": 6.0,
                    "severerisk": 10.0,
                    "conditions": "Clear",
                    "icon": "clear-day",
                    "stations": [
                        "remote"
                    ],
```

```
                    "source": "obs"
                },
                {
                    "datetime": "11:00:00",
                    "datetimeEpoch": 1651987800,
                    "temp": 36.4,
                    "feelslike": 34.4,
                    "humidity": 18.5,
                    "dew": 8.7,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 18.4,
                    "windspeed": 6.8,
                    "winddir": 330.2,
                    "pressure": 1007.0,
                    "visibility": 24.1,
                    "cloudcover": 4.7,
                    "solarradiation": 803.0,
                    "solarenergy": 2.9,
                    "uvindex": 8.0,
                    "severerisk": 10.0,
                    "conditions": "Clear",
                    "icon": "clear-day",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "12:00:00",
                    "datetimeEpoch": 1651991400,
                    "temp": 38.3,
                    "feelslike": 36.1,
                    "humidity": 15.59,
                    "dew": 7.7,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 16.9,
                    "windspeed": 8.6,
                    "winddir": 2.0,
                    "pressure": 1006.0,
                    "visibility": 24.1,
                    "cloudcover": 5.4,
                    "solarradiation": 944.0,
                    "solarenergy": 3.4,
                    "uvindex": 9.0,
                    "severerisk": 10.0,
                    "conditions": "Clear",
                    "icon": "clear-day",
```

```
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "13:00:00",
                    "datetimeEpoch": 1651995000,
                    "temp": 39.1,
                    "feelslike": 36.8,
                    "humidity": 14.52,
                    "dew": 7.3,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 14.4,
                    "windspeed": 9.7,
                    "winddir": 345.9,
                    "pressure": 1005.0,
                    "visibility": 24.1,
                    "cloudcover": 94.0,
                    "solarradiation": 1019.0,
                    "solarenergy": 3.7,
                    "uvindex": 10.0,
                    "severerisk": 10.0,
                    "conditions": "Overcast",
                    "icon": "cloudy",
                    "stations": [
                        "remote"
                    ],
                    "source": "obs"
                },
                {
                    "datetime": "14:00:00",
                    "datetimeEpoch": 1651998600,
                    "temp": 38.3,
                    "feelslike": 36.0,
                    "humidity": 15.27,
                    "dew": 7.4,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 18.7,
                    "windspeed": 12.6,
                    "winddir": 314.2,
                    "pressure": 1003.0,
                    "visibility": 24.1,
                    "cloudcover": 98.9,
                    "solarradiation": 525.0,
                    "solarenergy": 1.9,
                    "uvindex": 5.0,
```

```
                    "severerisk": 10.0,
                    "conditions": "Overcast",
                    "icon": "cloudy",
                    "stations": [],
                    "source": "fcst",
                    "sunrise": "06:04:49",
                    "sunriseEpoch": 1651970089,
                    "sunset": "18:54:32",
                    "sunsetEpoch": 1652016272,
                    "moonphase": 0.22
                },
                {
                    "datetime": "15:00:00",
                    "datetimeEpoch": 1652002200,
                    "temp": 36.5,
                    "feelslike": 34.6,
                    "humidity": 19.16,
                    "dew": 9.3,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 17.6,
                    "windspeed": 18.0,
                    "winddir": 306.9,
                    "pressure": 1003.0,
                    "visibility": 24.1,
                    "cloudcover": 100.0,
                    "solarradiation": 188.0,
                    "solarenergy": 0.7,
                    "uvindex": 2.0,
                    "severerisk": 10.0,
                    "conditions": "Overcast",
                    "icon": "cloudy",
                    "stations": null,
                    "source": "fcst"
                },
                {
                    "datetime": "16:00:00",
                    "datetimeEpoch": 1652005800,
                    "temp": 33.8,
                    "feelslike": 33.1,
                    "humidity": 31.15,
                    "dew": 14.4,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 24.8,
                    "windspeed": 21.2,
                    "winddir": 281.9,
                    "pressure": 1002.0,
                    "visibility": 24.1,
```

```
                    "cloudcover": 34.9,
                    "solarradiation": 93.0,
                    "solarenergy": 0.3,
                    "uvindex": 1.0,
                    "severerisk": 10.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-day",
                    "stations": null,
                    "source": "fcst"
                },
                {
                    "datetime": "17:00:00",
                    "datetimeEpoch": 1652009400,
                    "temp": 35.0,
                    "feelslike": 35.0,
                    "humidity": 31.68,
                    "dew": 15.7,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 23.0,
                    "windspeed": 24.5,
                    "winddir": 277.5,
                    "pressure": 1002.0,
                    "visibility": 24.1,
                    "cloudcover": 16.9,
                    "solarradiation": 541.0,
                    "solarenergy": 1.9,
                    "uvindex": 5.0,
                    "severerisk": 10.0,
                    "conditions": "Clear",
                    "icon": "clear-day",
                    "stations": null,
                    "source": "fcst"
                },
                {
                    "datetime": "18:00:00",
                    "datetimeEpoch": 1652013000,
                    "temp": 31.8,
                    "feelslike": 33.1,
                    "humidity": 46.11,
                    "dew": 18.8,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 24.5,
                    "windspeed": 22.7,
                    "winddir": 263.9,
                    "pressure": 1003.0,
                    "visibility": 24.1,
                    "cloudcover": 71.6,
```

```
                    "solarradiation": 368.0,
                    "solarenergy": 1.3,
                    "uvindex": 4.0,
                    "severerisk": 10.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-day",
                    "stations": null,
                    "source": "fcst"
                },
                {
                    "datetime": "19:00:00",
                    "datetimeEpoch": 1652016600,
                    "temp": 28.9,
                    "feelslike": 30.3,
                    "humidity": 56.53,
                    "dew": 19.4,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 28.1,
                    "windspeed": 18.7,
                    "winddir": 262.8,
                    "pressure": 1004.0,
                    "visibility": 24.1,
                    "cloudcover": 89.8,
                    "solarradiation": 84.0,
                    "solarenergy": 0.3,
                    "uvindex": 0.0,
                    "severerisk": 3.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-night",
                    "stations": null,
                    "source": "fcst"
                },
                {
                    "datetime": "20:00:00",
                    "datetimeEpoch": 1652020200,
                    "temp": 27.0,
                    "feelslike": 28.5,
                    "humidity": 66.37,
                    "dew": 20.2,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 29.5,
                    "windspeed": 16.9,
                    "winddir": 263.3,
                    "pressure": 1005.0,
                    "visibility": 24.1,
                    "cloudcover": 91.6,
                    "solarradiation": 4.0,
```

```
                    "solarenergy": 0.0,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Overcast",
                    "icon": "cloudy",
                    "stations": null,
                    "source": "fcst"
                },
                {
                    "datetime": "21:00:00",
                    "datetimeEpoch": 1652023800,
                    "temp": 25.5,
                    "feelslike": 25.5,
                    "humidity": 76.19,
                    "dew": 21.0,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 30.6,
                    "windspeed": 14.4,
                    "winddir": 269.1,
                    "pressure": 1006.0,
                    "visibility": 24.1,
                    "cloudcover": 75.4,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 5.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-night",
                    "stations": null,
                    "source": "fcst"
                },
                {
                    "datetime": "22:00:00",
                    "datetimeEpoch": 1652027400,
                    "temp": 24.3,
                    "feelslike": 24.3,
                    "humidity": 84.91,
                    "dew": 21.6,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 26.3,
                    "windspeed": 13.3,
                    "winddir": 265.5,
                    "pressure": 1006.0,
                    "visibility": 24.1,
                    "cloudcover": 24.7,
                    "solarradiation": 0.0,
                    "solarenergy": null,
```

```
                    "uvindex": 0.0,
                    "severerisk": 15.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-night",
                    "stations": null,
                    "source": "fcst"
                },
                {
                    "datetime": "23:00:00",
                    "datetimeEpoch": 1652031000,
                    "temp": 23.2,
                    "feelslike": 23.2,
                    "humidity": 92.97,
                    "dew": 22.0,
                    "precip": 0.0,
                    "precipprob": 0.0,
                    "snow": 0.0,
                    "snowdepth": 0.0,
                    "preciptype": null,
                    "windgust": 26.3,
                    "windspeed": 12.2,
                    "winddir": 267.4,
                    "pressure": 1007.0,
                    "visibility": 24.1,
                    "cloudcover": 52.6,
                    "solarradiation": 0.0,
                    "solarenergy": null,
                    "uvindex": 0.0,
                    "severerisk": 15.0,
                    "conditions": "Partially cloudy",
                    "icon": "partly-cloudy-night",
                    "stations": null,
                    "source": "fcst"
                }
            ]
        }
    ],
    "alerts": [],
    "currentConditions": {
        "datetime": "14:00:00",
        "datetimeEpoch": 1651998600,
        "temp": 38.3,
        "feelslike": 36.0,
        "humidity": 15.27,
        "dew": 7.4,
        "precip": 0.0,
        "precipprob": 0.0,
        "snow": 0.0,
        "snowdepth": 0.0,
        "preciptype": null,
        "windgust": 18.7,
        "windspeed": 12.6,
        "winddir": 314.2,
        "pressure": 1003.0,
        "visibility": 24.1,
```

```
        "cloudcover": 98.9,
        "solarradiation": 525.0,
        "solarenergy": 1.9,
        "uvindex": 5.0,
        "severerisk": 10.0,
        "conditions": "Overcast",
        "icon": "cloudy",
        "stations": [],
        "source": "fcst",
        "sunrise": "06:04:49",
        "sunriseEpoch": 1651970089,
        "sunset": "18:54:32",
        "sunsetEpoch": 1652016272,
        "moonphase": 0.22
    }
}
```