ДОДАТОК А
(Довідковий)

ПОВНИЙ СКЕТЧ ДЛЯ ПЛАТИ ESP 32

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <Adafruit_BME680.h>
const char* ssid = "MERCUSYS_5G_27";
const char* password = "dk32645dk";
const char* serverAddress = "http://cleanairapi.com";
const int serverPort = 80;
const String endpoint = "/api/air_quality_data";
Adafruit_BME680 bme;
void setup() {
  Serial.begin(115200);

  // Підключення до Wi-Fi мережі
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Ініціалізація сенсора BME680
  if (!bme.begin()) {
    Serial.println("Could not find BME680 sensor!");
    while (1);
  }
  Serial.println("BME680 sensor found");

  // Установка інтервалу вимірювань
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
}
void loop() {
  // Вимірювання показників якості повітря
  float temperature = bme.temperature;
```

```
  float humidity = bme.humidity;
  float pressure = bme.pressure;
  float gasResistance = bme.gas_resistance / 1000.0; // Конвертування в кОм
  float airQualityIndex = bme.readGas();

  // Створення JSON об'єкта
  DynamicJsonDocument jsonDocument(200);
  jsonDocument["temperature"] = temperature;
  jsonDocument["humidity"] = humidity;
  jsonDocument["pressure"] = pressure;
  jsonDocument["gasResistance"] = gasResistance;
  jsonDocument["airQualityIndex"] = airQualityIndex;

  // Перетворення JSON об'єкта в рядок
  String jsonString;
  serializeJson(jsonDocument, jsonString);

  // Відправка POST запиту на сервер
  HTTPClient http;
  http.begin(serverAddress, serverPort, endpoint);
  http.addHeader("Content-Type", "application/json");
  int httpResponseCode = http.POST(jsonString);

  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("Error sending POST request! HTTP Error code: ");
    Serial.println(httpResponseCode);
  }

  http.end();

  delay(60000); // Затримка в 1 хвилину перед наступним вимірюванням
}
```

ДОДАТОК Б
(Довідковий)

ПОВНИЙ КОД ДЛЯ IOS ДОДАТКА

Розділ Model:

```
import Foundation
import MapKit
import Observation
import SwiftUI
import XCAAQI

enum LocationStatus: Equatable {
   case requestLocation
   case locationAuthorized(String)
   case error(String)
   case requestAQIConditions
   case standby
}

@Observable
class ViewModel: NSObject {
   let locationManager = CLLocationManager()
   let aqiClient = AirQualityClient(apiKey: "AI-
zaSyA610Rl_yWfGJctGLEiZSZdCl43zDaWh6c")
   let coordinatesFinder = CoordinatesFinder()

   var currentLocation: CLLocationCoordinate2D?
   var locationStatus = LocationStatus.requestLocation
   var position: MapCameraPosition = .automatic
   var annotations: [AQIResponse] = []
   var selection: AQIResponse?
   var presentationD = PresentationDetent.height(176)
   var lat: Double = 0
   var long: Double = 0

   var radiusArray: [(Double, Int)]

   init(radiusArray: [(Double, Int)] = [(4000, 6), (8000, 12)]) {
      self.radiusArray = radiusArray
      super.init()
      locationManager.delegate = self
      locationManager.requestWhenInUseAuthorization()
```

```
        }
        @MainActor
        func handleCoordinateChange(_ coordinate: CLLocationCoordinate2D)
async {
            do {
                self.locationStatus = .requestAQIConditions
                self.position = .region(.init(center: coordinate, latitudinalMeters: 0,
longitudinalMeters: 16000))
                let coordinates = getCoordinates(coordinate)
                self.annotations = try await aqiClient.getCurrentConditions(coordi-
nates: coordinates.map {($0.latitude, $0.longitude)})
                self.locationStatus = .standby
            } catch {
                self.locationStatus = .error(error.localizedDescription)
            }
        }
        func getCoordinates(_ coordinate: CLLocationCoordinate2D) -> [CLLoca-
tionCoordinate2D] {
            var results: [CLLocationCoordinate2D] = [coordinate]
            radiusArray.forEach {
                results += coordinatesFinder.findCoordinates(coordinate, r: $0.0, n:
$0.1)
            }
            return results
        }

        // Закоментований метод для отримання даних з власного сервера
        /*
        @MainActor
        func fetchAQIDataFromServer() async {
            do {
                let serverURL = URL(string: "http://cleanairapi.com/api/air_qual-
ity_data")!
                let (data, _) = try await URLSession.shared.data(from: serverURL)
                let decoder = JSONDecoder()
                let aqiResponses = try decoder.decode([AQIResponse].self, from:
data)
                self.annotations = aqiResponses
                self.locationStatus = .standby
```

```
            } catch {
                self.locationStatus = .error(error.localizedDescription)
            }
        }
        */
    }

    extension ViewModel: CLLocationManagerDelegate {
        func locationManagerDidChangeAuthorization(_ manager: CLLocation-
Manager) {
            switch manager.authorizationStatus {
            case .authorizedWhenInUse:
                manager.requestLocation()
            default:
                self.locationStatus = .locationAuthorized("Unauthorized location ac-
cess")
            }
        }
        func locationManager(_ manager: CLLocationManager, didFailWithError
error: any Error) {
            self.locationStatus = .error(error.localizedDescription)
        }
        func locationManager(_ manager: CLLocationManager, didUpdateLoca-
tions locations: [CLLocation]) {
            guard let coordinate = locations.first?.coordinate else { return }
            if currentLocation == nil {
                lat = coordinate.latitude
                long = coordinate.longitude
                Task { await self.handleCoordinateChange(coordinate) }
            }
            currentLocation = coordinate
        }
    }
```

Розділ ViewModel:

```
import Foundation
import CoreLocation

struct CoordinatesFinder {

    let R = 6371000.0 // радіус Землі в метрах
    let pi = 3.141592653589793 // константа пі

    // Функція для конвертації градусів у радіани
    func deg2rad(_ deg: Double) -> Double {
        return deg * pi / 180
    }

    // Функція для конвертації радіан у градуси
    func rad2deg(_ rad: Double) -> Double {
        return rad * 180 / pi
    }

    // Функція для знаходження координат на колі навколо заданої точки
    func findCoordinates(_ coordinate: CLLocationCoordinate2D, r: Double,
n: Int) -> [CLLocationCoordinate2D] {
        // Конвертація вхідних даних у радіани
        let phi1 = deg2rad(coordinate.latitude)
        let lambda1 = deg2rad(coordinate.longitude)
        let d = r / R // кутова відстань

        // Ініціалізація пустого масиву для зберігання координат
        var coordinates: [CLLocationCoordinate2D] = []

        // Цикл по різних азимутах
        for i in 0..<n {
            // Конвертація азимуту в радіани
            let theta = deg2rad(Double(i) * 360 / Double(n))

            // Застосування формули для знаходження точки призначення
            let phi2 = asin(sin(phi1) *  cos(d) + cos(phi1) * sin(d) * cos(theta))
            let lambda2 = lambda1 + atan2(sin(theta) * sin(d) * cos(phi1), cos(d)
```

```
- sin(phi1) * sin(phi2))

                // Конвертація вихідних даних у градуси
                let lat2 = rad2deg(phi2)
                let lon2 = rad2deg(lambda2)

                // Додавання координат до масиву
                coordinates.append(.init(latitude: lat2, longitude: lon2))
            }

            // Повернення масиву координат
            return coordinates
        }

    }
```

Розділ View:

```
    import MapKit
    import SwiftUI
    import XCAAQI

    struct ContentView: View {
        @State var vm = ViewModel()
        var body: some View {

            Map(position: $vm.position, selection: $vm.selection) {
                ForEach(vm.annotations) { aqi in
                    Annotation(aqi.aqiDisplay, coordinate: aqi.coordinate) {
                        CircleViewAqi(aqi: aqi, isSelected: aqi == vm.selection)
                    }
                    .tag(aqi)
                    .annotationTitles(.hidden)
                }
            }
            .mapStyle(.hybrid(elevation: .flat, pointsOfInterest: .all, showsTraffic:
false))
            //.mapControls {
              // MapUserLocationButton()
```

```
            //  MapCompass()
           // }
            .sheet(isPresented: .constant(true)) {
               ScrollView {
                  VStack{
                     if let selection = vm.selection {
                        selectedAQIView(aqi: selection)
                     } else {
                        if vm.locationStatus != .requestLocation && vm.locationSta-
tus != .requestAQIConditions {
                           locationFromView
                        }
                        if vm.locationStatus == .requestAQIConditions {
                           ProgressView("Requesting AQI conditions ... just wait ...")
                        }
                        if vm.locationStatus == .requestLocation {
                           ProgressView("Requesting current locations ... just wait ...")
                        }
                        if case let .locationAutorized(text) = vm.locationStatus {
                           Text(text)
                        }
                        if case let .error(text) = vm.locationStatus {
                           Text(text)
                        }
                     }
                  }
               }
               .padding()
               .safeAreaPadding(.top)
               .presentationDetents([.height(24), .height(176)], selection:
$vm.presentationD)
               .presentationBackground(.ultraThinMaterial)
               .presentationBackgroundInteraction(.enabled(upThrough:
.height(176)))
               .interactiveDismissDisabled()
            }
            .onChange(of: vm.selection) { oldValue, newValue in
               if oldValue == nil && newValue != nil {
                  vm.presentationD = .height(176)
```

```
            }
        }
        .navigationTitle("Air Quality")
        .navigationBarTitleDisplayMode(.inline)
        .toolbarBackground(.ultraThinMaterial, for: .navigationBar)
    }
    func selectedAQIView(aqi: AQIResponse) -> some View {
        HStack(spacing: 16) {
            CircleViewAqi(aqi: aqi, size: CGSize(width: 80, height: 80))
            VStack(alignment: .leading)  {
                Text("Coordinate: \(aqi.coordinate.latitude), \(aqi.coordinate.longi-
tude)")
                Text(aqi.category)
                Text("Dominant Pollutant: \(aqi.dominantPollutant)")
                Text(aqi.displayName)
            }
        }
        .padding(.top)
        .padding(.horizontal)
        .frame(maxWidth: .infinity)
    }
    @ViewBuilder
    var locationFromView: some View {
        Text("Get AQI current (around a coordinate)")
            .font(.headline)
            .padding(.bottom, 8)

        HStack {
            Text("Latit")
            TextField("Enter latitude", value: $vm.lat, format: .number)
            Text("Long")
            TextField("Enter longitude", value: $vm.long, format: .number)
        }
        .keyboardType(.decimalPad)
        .textFieldStyle(.roundedBorder)
        .padding(.bottom, 8)

        HStack{
            Button("Use current location") {
```

```
                vm.lat = vm.currentLocation?.latitude ?? 0
                vm.long = vm.currentLocation?.longitude ?? 0
                Task {
                    await vm.handleCoordinateChange(.init(latitude: vm.lat, longitude: vm.long))

                }
            }
            .frame(width: 170, height: 35)
                .background(.green)
                .foregroundColor(.black)
                .clipShape(Rectangle())
            Button("Refresh AQI") {
                Task {
                    await vm.handleCoordinateChange(.init(latitude: vm.lat, longitude: vm.long))
                }
            }
            .frame(width: 120, height: 35)
                .background(.green)
                .foregroundColor(.black)
                .clipShape(Rectangle())
        }
    }
}

#Preview {
    NavigationStack {
        ContentView(vm: .init(radiusArray: []))
    }
}
```

Розділ CircleViewAqi:

```
import SwiftUI
import XCAAQI
struct CircleViewAqi: View {
    let aqi: AQIResponse
    var isSelected: Bool = false
```

```
        var size: CGSize = .init(width: 44, height: 44)
    var body: some View {
        Circle()
            .stroke(Color(red: aqi.color.red, green: aqi.color.green, blue: aqi.color.blue), lineWidth: isSelected ? 4 : 3)
            .frame(width: size.width, height: size.height)
            .overlay {
                Text(aqi.aqiDisplay)
                    .foregroundStyle(.white)
                    .fontWeight(.bold)
            }
            .scaleEffect(isSelected ? CGSize(width: 1.5, height: 1.5) : CGSize(width: 1, height: 1))
    }
    #Preview {
        CircleViewAqi(aqi: .init(aqiDisplay: "23", color: .init(red: 0.2, green: 0.5, blue: 0.5)), isSelected: true)
    }
```

ДОДАТОК В
(Довідковий)

КОПІЇ ПРЕЗЕНТАЦІЇ

Рисунок В.1 – Титульний слайд презентації

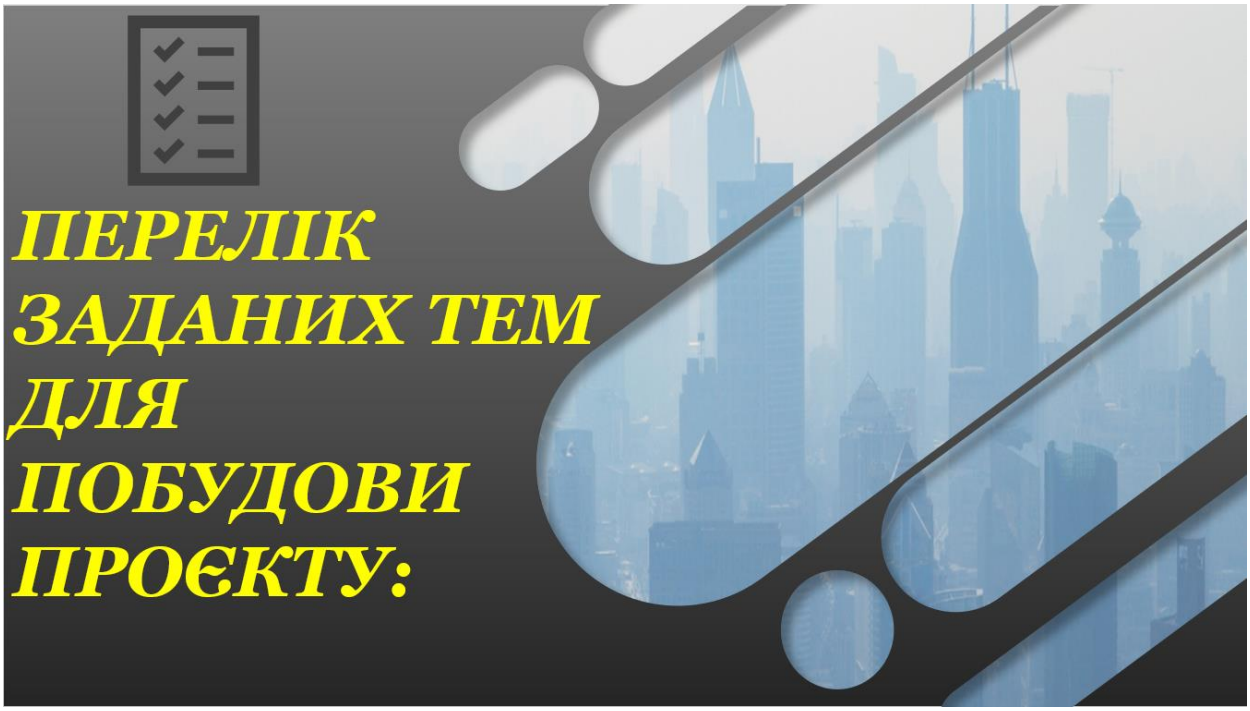

Рисунок В.2 – Перелік заданих тем

Рисунок В.3 – Конкретно задані теми

Рисунок В.4 – Вступ

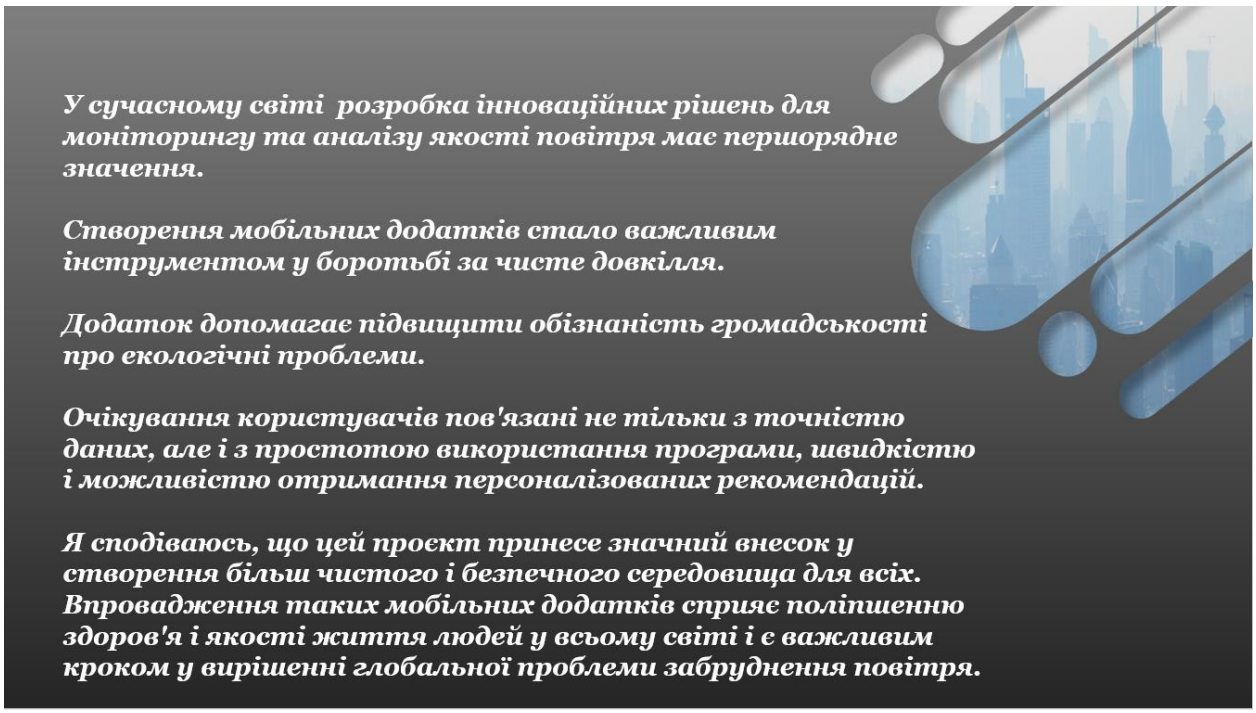

Рисунок В.5 – Детальна інформація вступу

Рисунок В.6 – Актуальність

Рисунок В.7 – Детальна інформація актуальності

Рисунок В.8 – Структура проєкту

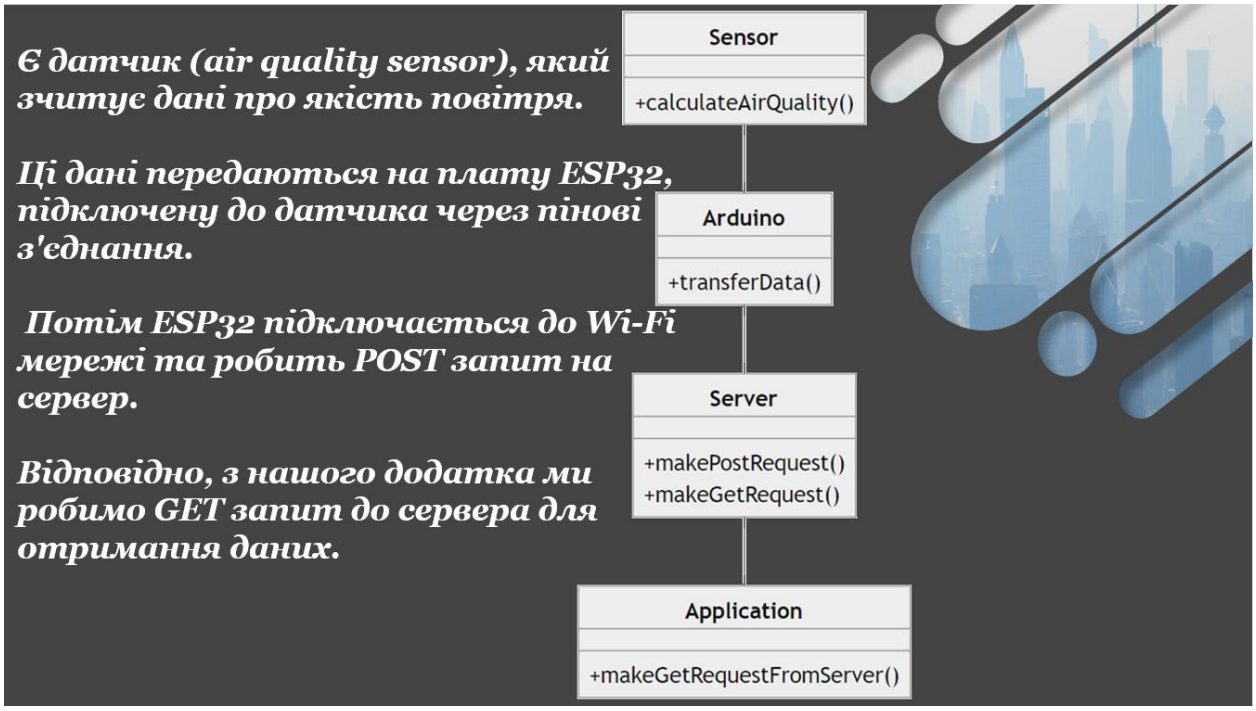

Рисунок В.9 – Детальна інформація структури проєкту

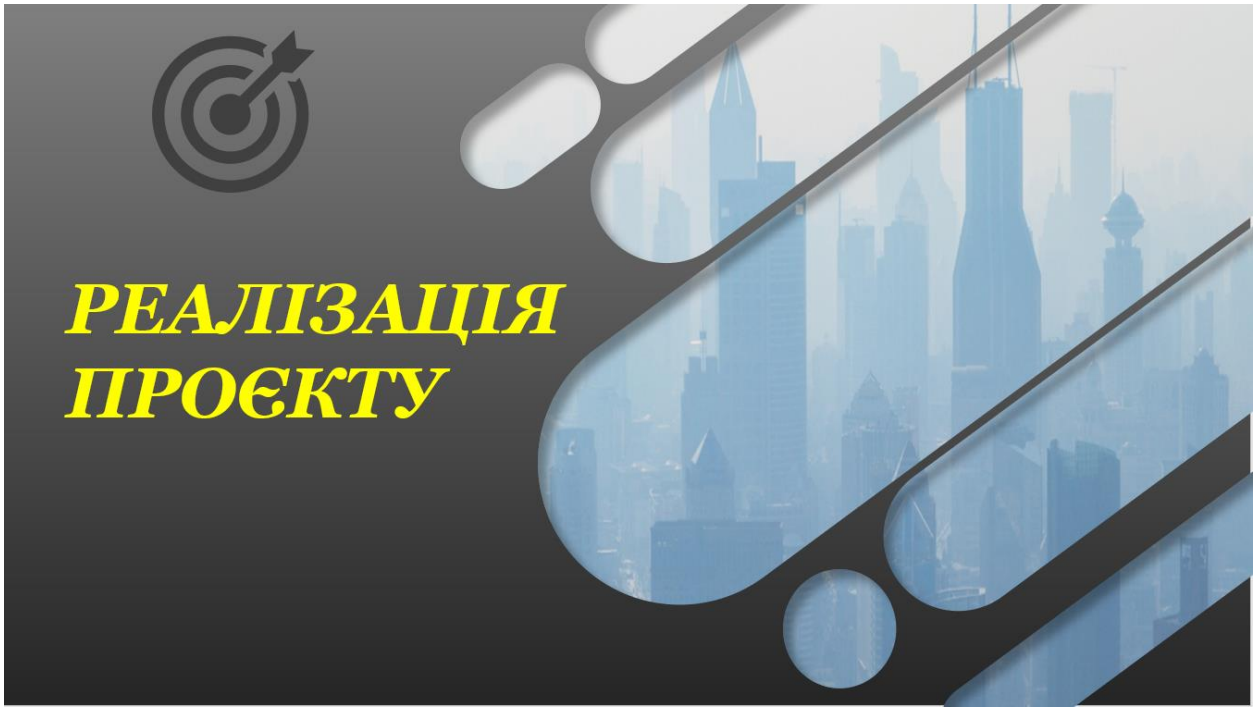

Рисунок В.10 – Реалізація проєкту

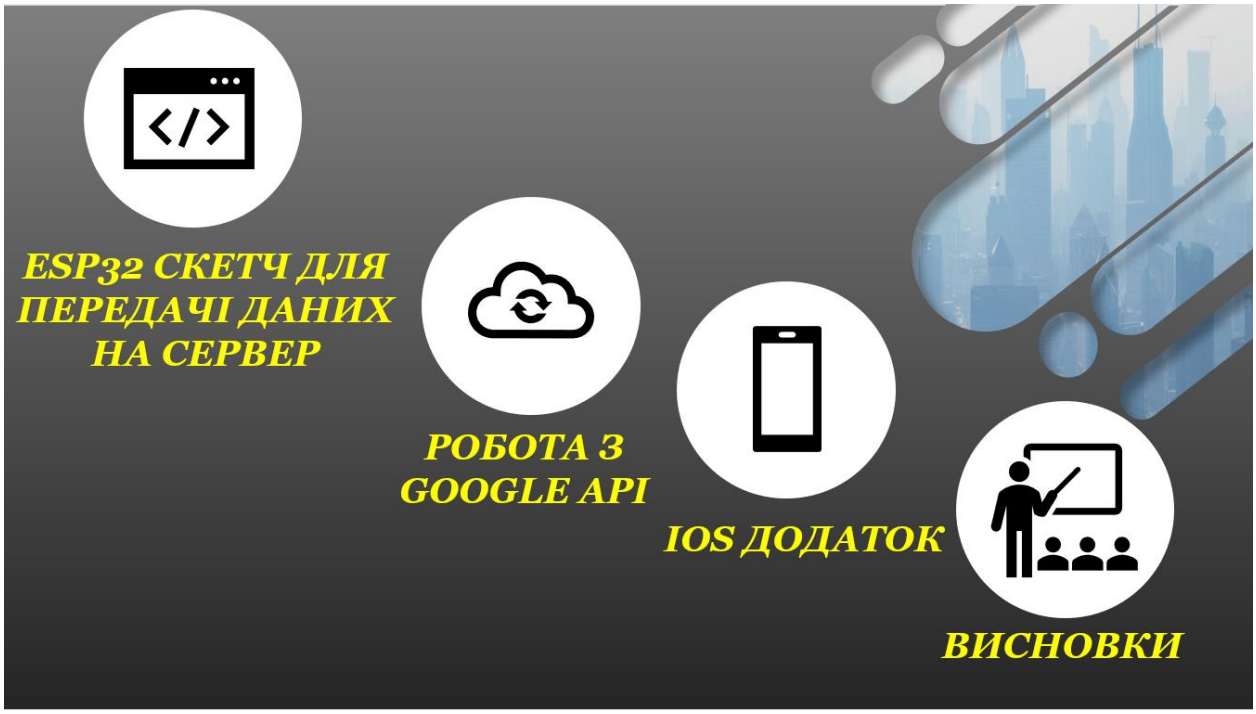

Рисунок В.11 – Конкретний план реалізації проєкту

Рисунок В.12 – Підключення схеми та передача даних на сервер

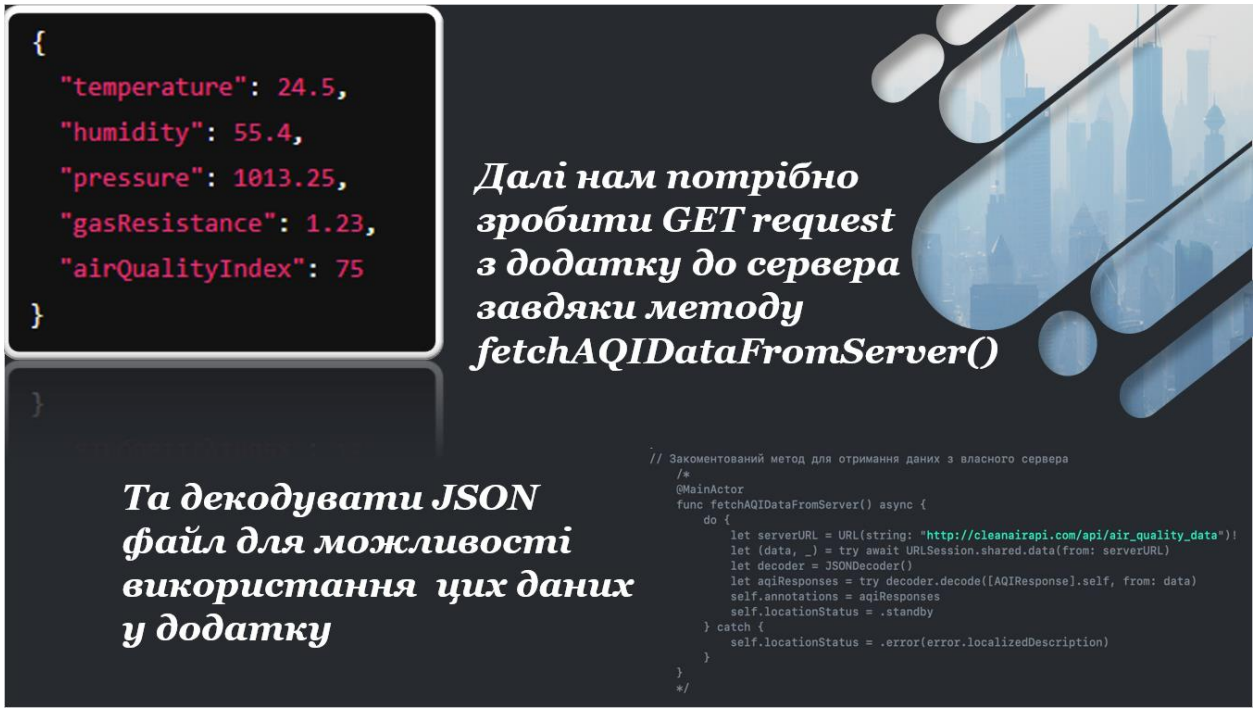

Рисунок В.13 – Використання переданих даних у додатку

Рисунок В.14 – Робота з Google API

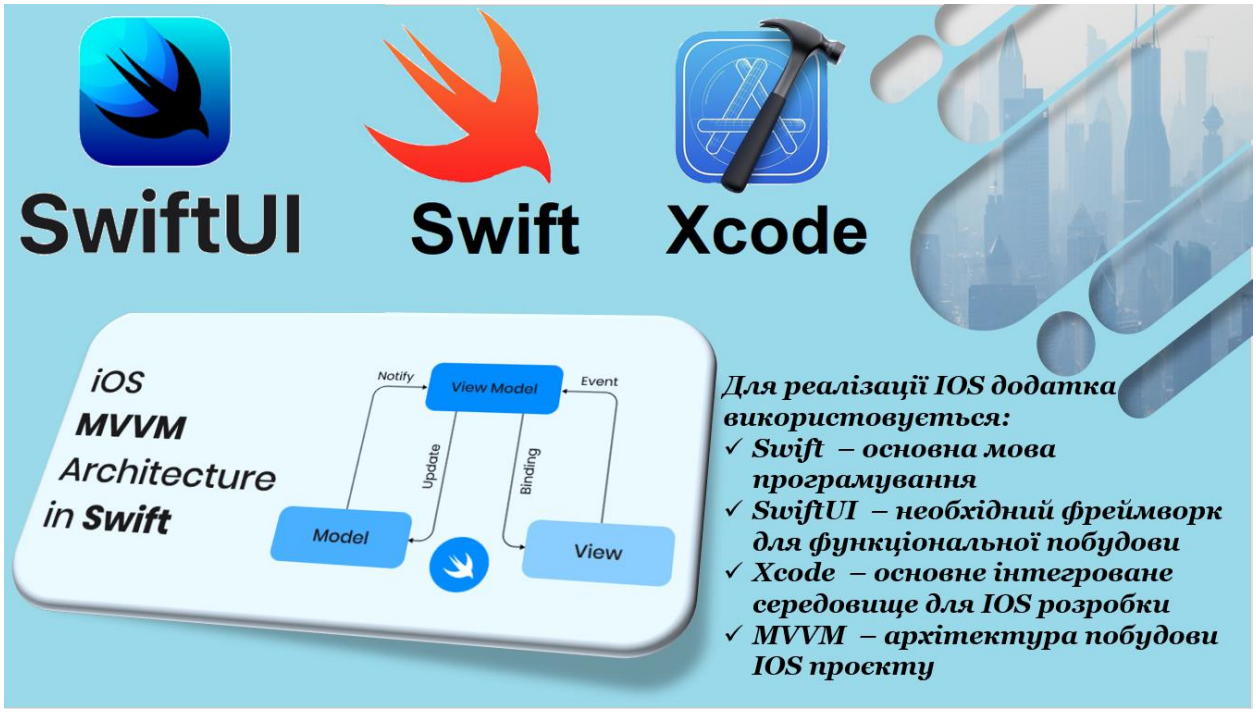

Рисунок В.15 – Архітектура IOS додатка

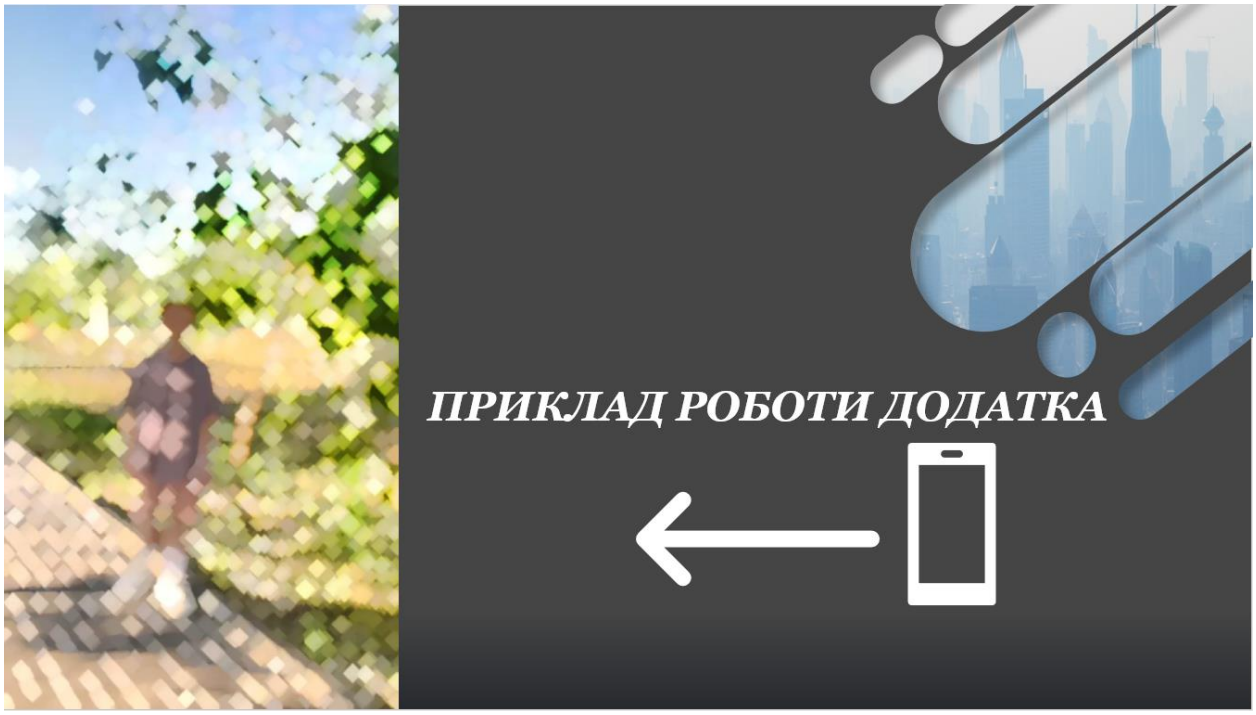

Рисунок В.16 – Приклад роботи IOS додатка

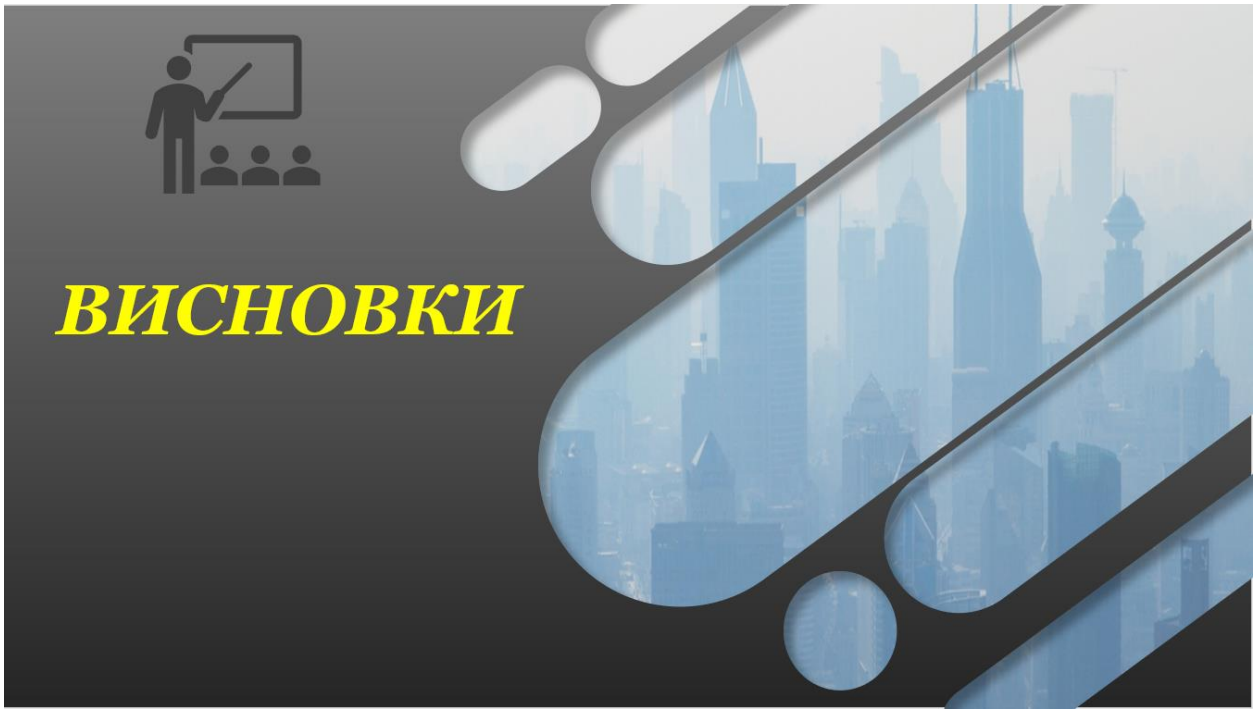

Рисунок В.17 – Висновки

Рисунок В.18 – Детальна інформація щодо висновків

Рисунок В.19 – Заключний слайд презентації

ДОДАТОК Г
(Довідковий)

ВІДОМОСТІ АТЕСТАЦІЙНОГО ПРОЄКТУ

№ п/п	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1		Пояснювальна записка	92 с.

Зм	Арк.	№ докум.	Підп.	Дата	ГЮІК.ХХХХХХ.000Д2		
Розроб.		Косенко			Розробка додатку для IOS. Надання даних про якість повітря.	Літера	Аркуш / Аркушів
Перевір.		Ганшин				У	1 / 1
Н. контр.		Бітченко				ХНУРЕ Кафедра РТІКС	
Затв.		Зарудний					