Final Software Development Project:

Build Multiplatform “Location Tracker" App with .NET MAUI

Hoai Nam Tran

(MSCS-533-A01) Software Engineering and Multiplatform App Development

Dr. Gary Perry

University of the Cumberlands

1. Introduction

Mobile applications that leverage geolocation data have become increasingly important in modern software systems. These applications are widely used in navigation, logistics, fitness tracking, and behavioral analytics. The purpose of this project is to design and develop a cross-platform mobile application that tracks user location and visualizes movement patterns using a heatmap.

This application is developed using C# and the .NET MAUI framework. It collects user location data over time, stores it locally using SQLite, and displays the data on a map using a heatmap-style visualization. The system provides a practical demonstration of integrating GPS services, local databases, and map-based visualization in a real-world application.

2. GitHub Repository

https://github.com/NamTran9694/LocationHeatmapApp

3. Objectives:

The main objectives of this project are:

- To develop a mobile application using .NET MAUI
- To capture real-time user location data
- To store location data efficiently using SQLite
- To visualize user movement patterns on a map
- To simulate a heatmap using graphical overlays

4. Technologies Used

This project utilizes the following technologies:

- C#: Primary programming language used for application development
- .NET MAUI: Framework for building cross-platform mobile applications

- SQLite: Lightweight database for storing location data locally
- .NET MAUI Maps: Used for displaying maps and rendering location data
- Android Emulator: Used to simulate GPS movement during testing
- Visual Studio Code / .NET CLI: Development environment

These tools were selected because they provide strong support for cross-platform development, efficient data handling, and seamless integration with mobile device features.

5. System Architecture

The application follows a modular architecture consisting of the following components:

5.1.UI Layer

- Built using XAML
- Contains buttons (Start, Stop, Refresh, Clear)
- Includes a slider to control visualization radius
- Displays map and status information

5.2. Service Layer

- Handles GPS location tracking
- Uses Geolocation API to retrieve user coordinates

5.3.Data Layer

- Uses SQLite database
- Stores latitude, longitude, and timestamp

5.4.Visualization Layer

- Uses map overlays (circles) to simulate heatmap
- Adjusts intensity based on data density

6. System Functionality

The application provides the following features:

Start Tracking:

- Begins collecting user location every few seconds
- Stores data in SQLite database

Stop Tracking:

- Stops location collection process

Refresh:

- Retrieves stored data
- Generates heatmap visualization

Clear:

- Deletes all stored location data
- Clears map overlays

Radius Control:

- Allows user to adjust how far each data point spreads
- Influences heatmap intensity and visualization

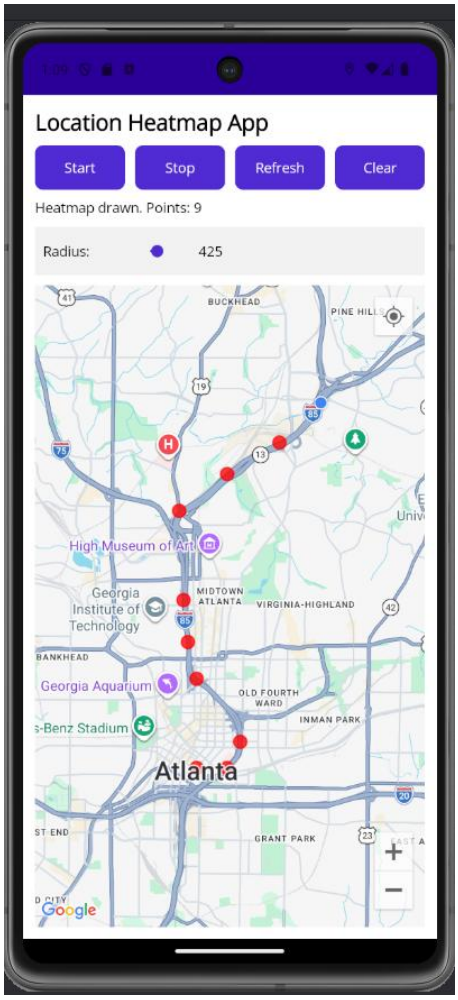
7. Testing and Validation

The application was tested using the Android Emulator.

Testing Process:

- GPS coordinates were manually simulated using emulator controls
- Multiple locations were injected to simulate movement
- The application successfully recorded and stored location data
- Heatmap visualization was generated using stored data

Results:



- Location tracking worked consistently
- Data was accurately stored in SQLite
- Heatmap correctly reflected user movement and clustering
- UI responded properly to user interactions

8. Challenges and Solutions

Challenge 1: Lack of Built-in Heatmap Support

One of the primary challenges encountered during development was that the .NET MAUI Maps component does not provide native support for heatmap visualization. This limitation made it difficult to directly represent location density using standard tools.

Solution: To address this, a custom heatmap approach was implemented using overlapping semi-transparent circles. Location data points were grouped and rendered as circular overlays on the map. Areas with higher data density resulted in more overlapping circles, creating a visual effect similar to a heatmap. This approach provided a flexible and effective workaround despite the lack of built-in functionality.

Challenge 2: User Interface (UI) Layout Issues

During development, several UI challenges were encountered, particularly with element alignment and responsiveness across different screen sizes. For example, the radius control slider and labels overlapped or appeared compressed on smaller screens.

Solution: The layout was redesigned using a combination of Grid and StackLayout components to ensure proper spacing and alignment. Fixed widths and responsive layout structures were applied to prevent UI elements from overlapping. Additional adjustments such as padding, spacing, and alignment properties improved overall usability and visual clarity.

Challenge 3: GPS Simulation and Testing

Testing real-time location tracking posed a challenge because physical movement is not always practical during development. Additionally, consistent and repeatable testing scenarios are difficult to achieve with real-world movement.

Solution: The Android Emulator's location control feature was used to simulate GPS movement. By manually setting different coordinates, it was possible to mimic user movement across multiple locations. This allowed for controlled testing of data collection, storage, and heatmap visualization without requiring physical travel.

9. Manifest File

GitHub Link:

https://github.com/NamTran9694/LocationHeatmapApp/blob/main/Platforms/Android/AndroidManifest.xml

Android Manifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

  <!-- Permissions must be directly under <manifest>, NOT inside <application> --
>
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

  <application
      android:allowBackup="true"
      android:icon="@mipmap/appicon"
      android:roundIcon="@mipmap/appicon_round">

    <!-- API key MUST be inside <application> -->
    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyCyiqbyWNROEDSpkrrjs7ZCvU2H2jpfs1Q" />

  </application>
</manifest>
```

10. Source Code

Logic: MainPage.xaml.cs

https://github.com/NamTran9694/LocationHeatmapApp/blob/main/MainPage.xaml.cs

Data Handling: MauiProgram.cs

https://github.com/NamTran9694/LocationHeatmapApp/blob/main/MauiProgram.cs

GPS Tracking: LocationDatabase.cs

https://github.com/NamTran9694/LocationHeatmapApp/blob/main/Data/LocationDatabase.cs

Heatmap Rendering: LocationPoint.cs

https://github.com/NamTran9694/LocationHeatmapApp/blob/main/Models/LocationPoint.cs