Flight Tracker App

This is an Android application that allows users to select multiple city stops for a journey, calculate the total distance and travel time, and view detailed information about each stop, including visa requirements. The app supports both metric (km) and imperial (miles) distance units and features lazy loading for optimal performance.

Features

- **City Selection:** Choose cities as travel stops from a predefined list.
- **Distance Calculation:** Calculates total distance and time between stops.
- **Unit Toggle:** Switch between metric (km) and imperial (miles) units.
- **Visa Information:** Displays visa requirements for each stop.
- **Journey Navigation:** Navigate through stops with "Next" and "Previous" buttons.
- **Progress Tracking:** View journey progress with a progress bar and detailed text.
- **Lazy Loading:** Efficiently displays city stops for smooth scrolling.
- **Reset Option:** Reset the journey to start over.

Tech Stack

- **Kotlin**: Programming language.
- **RecyclerView**: Displaying stops with lazy loading.
- **Spinner**: City selection.
- **ProgressBar**: Visual representation of journey progress.
- **Layouts**: XML-based UI design (to be converted to Jetpack Compose).
- **Jetpack Compose (in progress)**: Migrating UI from XML to Compose.

Installation

1. **Clone the Repository**
2. **Open in Android Studio**
- Open Android Studio and select `Open an existing project`.
- Navigate to the cloned directory and select the project folder.
3. **Build & Run**
- Connect an Android device or start an emulator.
- Click on `Run` or use the shortcut `Shift + F10`.

Usage

1. **Select City Stops:**
    - Use the spinner to select a city.
    - Click on "Add Stop" to include it in the journey.
2. **Calculate Route:**
    - Once at least two stops are selected, click on "Calculate Route".
    - Distance and estimated travel time are displayed for each leg.
3. **Toggle Distance Units:**
    - Click on "Switch to Miles" or "Switch to KM" to toggle units.
4. **Navigate Journey:**
    - Use "Next Stop" and "Previous Stop" buttons to navigate.
5. **Reset Journey:**
    - Click on "Reset Journey" to start a new route selection.

File Structure

```
src
└── main
    ├── java
    │   └── com.example.flightapp
    │       ├── MainActivity.kt      # Main logic for the app
    │       └── StopsAdapter.kt      # RecyclerView adapter for stops
    └── res
        ├── layout
        │   ├── activity_main.xml    # Main screen layout
        │   └── stop_item.xml        # Layout for each stop item
        └── raw
            └── stops.txt            # List of cities and details
```

Data Source

- The list of stops is loaded from `res/raw/stops.txt` in the following format:

```
CityName, Latitude, Longitude, VisaRequired (true/false), VisaDetails
```

Example:

```
Paris, 48.8566, 2.3522, true, Schengen Visa Required
New York, 40.7128, -74.0060, false, No Visa Required
```

File Structure

- MainActivity.kt: Main logic for managing stops, calculating distances, and handling UI interactions.
- StopsAdapter.kt: RecyclerView adapter to display stops and their details.
- activity_main.xml: Layout file for the main screen.
- stop_item.xml: Layout file for individual stop items in the RecyclerView.
- res/raw/stops.csv: Contains predefined stops with city names, coordinates, and visa information.

How It Works

MainActivity.kt

- **Data Model**: Stop class stores city name, latitude, longitude, visa requirement, and visa details.
- **UI Components**: Uses Spinner for city selection, RecyclerView for displaying stops, Buttons for navigation, and ProgressBar for journey tracking.
- **Data Loading**: Reads city data from a CSV file in the res/raw folder.
- **Route Calculation**: Calculates distances using the Haversine formula.
- **Unit Conversion**: Supports both kilometers and miles.
- **Navigation**: Next and Previous buttons allow users to navigate through stops, updating the progress bar accordingly.

StopsAdapter.kt

- **RecyclerView Adapter**: Manages the display of stops with city name, coordinates, distance from the previous stop, and visa requirements.
- **Highlighting**: Highlights the current stop during navigation.
- **Lazy Loading**: Efficiently displays stops using lazy loading for better performance.

Flight Tracker App

This is a Flight Tracker Android app built using **Jetpack Compose**. It allows users to plan flight routes by selecting multiple stops, view distances and travel times between stops, and track journey progress.

Features

- **City Selection:** Choose from a list of cities as stops for your journey.
- **Route Calculation:** Calculate the total distance and travel time for the selected route.
- **Distance Units:** Switch between metric (km) and imperial (miles) units.
- **Journey Tracking:** Navigate between stops to see distance and time from the previous city.
- **Progress Display:** Visual representation of journey progress.
- **Dynamic Loading:** Efficient rendering of stops using lazy loading for smooth scrolling.

Tech Stack

- **Kotlin** - For Android development.
- **Jetpack Compose** - For building the UI in a declarative manner.
- **Material 3** - For consistent design and theming.

Installation

1. Clone the repository.
2. Open the project in **Android Studio**.
3. Sync the Gradle dependencies.
4. Run the app on an emulator or physical device.

Usage

1. Select a city from the dropdown and add it to the journey.
2. Add multiple stops to create your flight route.
3. Calculate the route to see total distance and travel time.
4. Navigate between stops to track the journey.
5. Switch between kilometers and miles as needed.
6. Reset the journey to start over.

File Structure

```
com.example.flightappco
|
├── MainActivity.kt             # Main entry point, Jetpack Compose UI
└── res/raw/stops.txt            # Contains city data
```

Dependencies

The app uses the following dependencies:

```
// Jetpack Compose
implementation "androidx.compose.ui:ui:"
implementation "androidx.compose.material3:material3:"
implementation "androidx.activity:activity-compose:"
```

Data Format

The app reads stop data from a text file (`res/raw/stops.txt`) in the following format:

```
CityName, Latitude, Longitude, VisaRequired, VisaDetails
```

Example:

```
New York, 40.7128, -74.0060, true, Visa required for non-US citizens
Paris, 48.8566, 2.3522, false, No visa required
```