BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC THỦY LỢI

BÀI TẬP LỚN

PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

ĐỀ TÀI: ỨNG DỤNG DỰ BÁO THỜI TIẾT

Giáo viên hướng dẫn: ThS. Kiều Tuấn Dũng

Sinh viên thực hiện:

STT	Mã sinh viên	Họ và tên	Lớp
1	2251061789	Lê Đức Hùng	64CNTT1
2	2251061788	Đỗ Trần Hùng	64CNTT1
3	2251061748	Thái Duy Đức	64CNTT1

Hà Nội, năm 2025

BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC THỦY LỢI

BÀI TẬP LỚN

PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

ĐỀ TÀI: ỨNG DỤNG DỰ BÁO THỜI TIẾT

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bằng Số	Bằng Chữ
1	2251061789	Lê Đức Hùng	12/05/2004		
2	2251061788	Đỗ Trần Hùng	09/12/2004		
3	2251061748	Thái Duy Đức	12/04/2004		

CÁN BỘ CHẤM THI

Hà Nội, năm 2025

LỜI NÓI ĐẦU

Trong thời đại công nghệ số phát triển mạnh mẽ như hiện nay, các ứng dụng di động ngày càng đóng vai trò thiết yếu trong đời sống con người, hỗ trợ hiệu quả trong việc học tập, làm việc, giải trí và quản lý thông tin cá nhân. Trong đó, ứng dụng dự báo thời tiết là một trong những tiện ích quan trọng, giúp người dùng chủ động hơn trong việc sắp xếp lịch trình và lên kế hoạch cho các hoạt động ngoài trời, đặc biệt trong bối cảnh biến đổi khí hậu ngày càng khó lường.

Việt Nam là quốc gia có điều kiện thời tiết phức tạp và thay đổi liên tục giữa các vùng miền. Do đó, việc xây dựng một ứng dụng dự báo thời tiết chính xác, trực quan, dễ sử dụng là nhu cầu thiết thực, đặc biệt khi nhu cầu theo dõi thời tiết ngày càng tăng cao. Trên nền tảng đó, nhóm sinh viên lựa chọn thực hiện đề tài *"Dự báo thời tiết"* nhằm xây dựng một ứng dụng chạy trên hệ điều hành Android – nền tảng di động phổ biến nhất hiện nay.

Ứng dụng được thiết kế với giao diện hiện đại, hỗ trợ ngôn ngữ tiếng Việt, cung cấp thông tin thời tiết theo thời gian thực tại vị trí hiện tại hoặc các địa điểm người dùng quan tâm. Bên cạnh các thông tin cơ bản như nhiệt độ, độ ẩm, tình trạng mưa/nắng, hướng gió,… ứng dụng còn tích hợp chức năng dự báo 5 ngày tiếp theo, gợi ý hoạt động phù hợp theo điều kiện thời tiết, và lưu trữ danh sách thành phố yêu thích của người dùng.

Trong quá trình xây dựng, nhóm đã áp dụng nhiều công nghệ và thư viện hiện đại như Retrofit (kết nối API), Room (lưu dữ liệu cục bộ), LiveData, ViewModel, và kiến trúc MVVM, giúp tổ chức mã nguồn khoa học, tăng tính mở rộng và dễ bảo trì. Qua đó, nhóm không chỉ củng cố kiến thức về lập trình Android mà còn học hỏi thêm nhiều kỹ năng như phân tích hệ thống, làm việc nhóm, xử lý dữ liệu thời gian thực và cải thiện trải nghiệm người dùng.

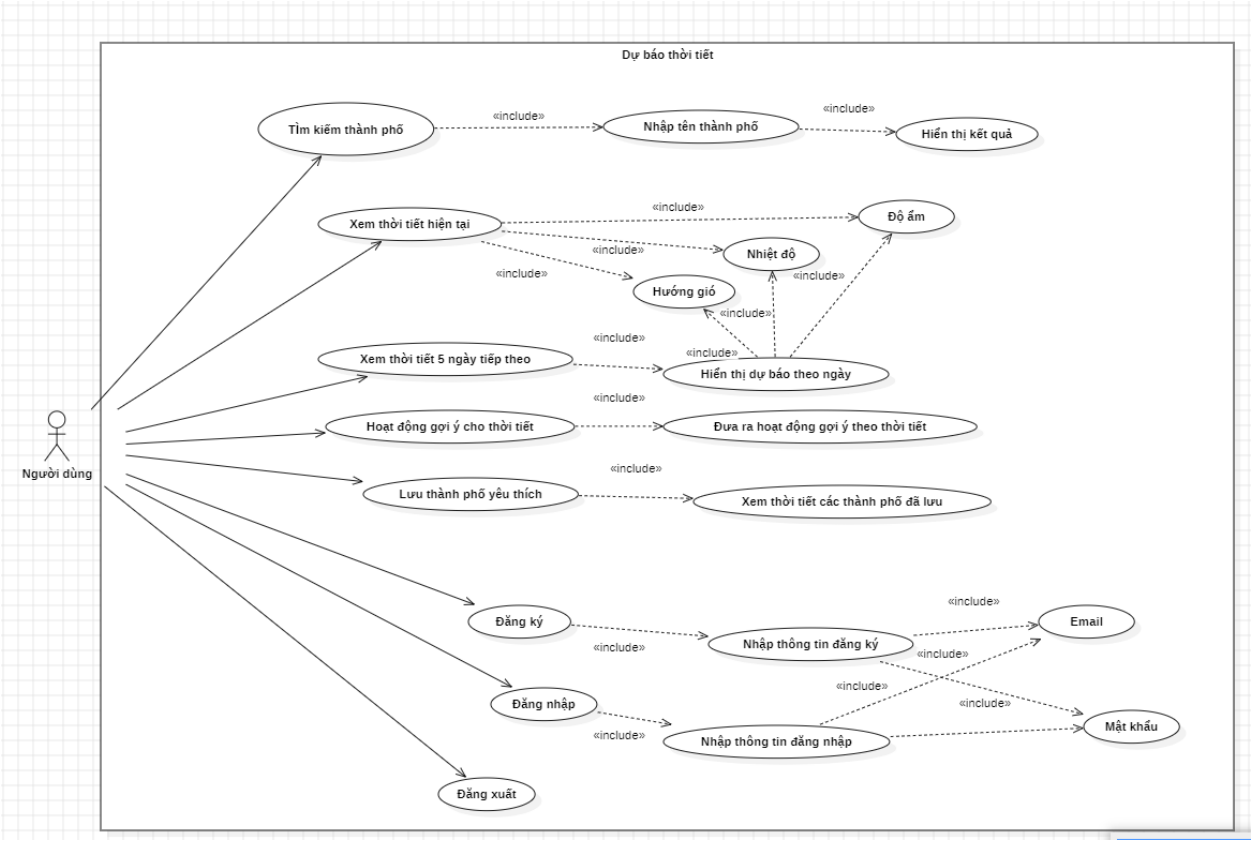
Đề tài mang tính ứng dụng thực tế cao, có thể triển khai và sử dụng rộng rãi trong cộng đồng. Nhóm hy vọng rằng sản phẩm sẽ góp phần giúp người dùng tiếp cận thông tin thời tiết một cách nhanh chóng, chính xác và tiện lợi hơn.

MỤC LỤC

LỜI NÓI ĐẦU 3
DANH MỤC HÌNH ẢNH 6
DANH MỤC CÁC TỪ VIẾT TẮT 7
CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI.....8
1.1. Giới thiệu về đề tài 8
1.2. Mục tiêu của đề tài 8
1.3. Phạm vi của đề tài 9
1.4. Phân chia nhiệm vụ 10
CHƯƠNG 2: KIẾN TRÚC VÀ CÔNG NGHỆ.....11
2.1. Kiến trúc hệ thống 11
2.2. Giới thiệu về Công nghệ phát triển 11
CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG 13
3.1. Thiết kế Figma 13
3.2. Thiết kế CSDL 19
3.3. Giao diện ứng dụng 20
3.3.1. Giao diện tìm kiếm 20
3.3.2. Giao diện đăng nhập 21
3.3.3. Giao diện đăng ký 22
3.3.4. Giao diện thời tiết hiện tại 23
3.3.5. Giao diện thời tiết 5 ngày tới 25
3.3.6. Giao diện các thành phố yêu thích đã lưu 26
3.4. Code minh họa các chức năng cốt lõi 27
3.4.1. Đăng nhập 27
3.4.2. Đăng ký 29
3.4.3. Thời tiết theo giờ 32
3.4.4. Dự báo thời tiết 5 ngày, hoạt động gợi ý 32
3.4.5. Thêm thành phố yêu thích 36
3.4.6. Xóa thành phố yêu thích 36

3.4.7. Thời tiết hiện tại 36
KẾT LUẬN 43
1. Kết quả đạt được 43
2. Nhược điểm 43
3. Hướng phát triển 44
TÀI LIỆU THAM KHẢO 46

DANH MỤC HÌNH ẢNH



DANH MỤC CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	RWD	Responsive Web Design
2	MVVM	Model – View – ViewModel
3	API	Application Programming Interface
4	GPS	Global Positioning System

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu về đề tài

Trong thời đại công nghệ số hiện nay, nhu cầu nắm bắt thông tin thời tiết một cách nhanh chóng, chính xác và tiện lợi ngày càng trở nên quan trọng đối với đời sống hằng ngày cũng như trong các lĩnh vực sản xuất, nông nghiệp, du lịch và giao thông. Nhận thức được điều đó, đề tài "Dự báo thời tiết" được phát triển dưới dạng ứng dụng Android, sử dụng ngôn ngữ Java/Kotlin kết hợp với các thư viện và công nghệ hiện đại nhằm cung cấp cho người dùng một ứng dụng tiện ích, hỗ trợ theo dõi tình hình thời tiết theo thời gian thực, dự báo trong nhiều ngày tới, kèm theo các thông tin chi tiết như nhiệt độ, độ ẩm, lượng mưa, tốc độ gió và chất lượng không khí.

Ứng dụng hướng tới giao diện thân thiện, dễ sử dụng, tích hợp các công nghệ hiện đại như API thời tiết, định vị GPS và trí tuệ nhân tạo để cải thiện độ chính xác trong dự báo. Không chỉ đơn thuần cung cấp thông tin, ứng dụng còn có khả năng đưa ra cảnh báo thời tiết cực đoan, giúp người dùng chủ động hơn trong việc lên kế hoạch và bảo vệ sức khỏe.

Với mong muốn mang lại một công cụ hữu ích, tiện lợi và có giá trị thực tiễn cao, đề tài "Dự báo thời tiết" không chỉ là một sản phẩm công nghệ mà còn là sự kết hợp giữa kỹ thuật lập trình và nhu cầu thiết thực của xã hội hiện đại.

1.2. Mục tiêu của đề tài

Mục tiêu chính của đề tài "Dự báo thời tiết" là xây dựng một ứng dụng di động trên nền tảng Android, sử dụng Java hoặc Kotlin, có khả năng thu thập và hiển thị thông tin thời tiết một cách nhanh chóng, chính xác, trực quan và thân thiện với người dùng. Cụ thể, đề tài hướng đến các mục tiêu kỹ thuật sau:

1. Thiết kế và xây dựng giao diện người dùng (UI/UX) theo tiêu chuẩn Material Design, đảm bảo tính thẩm mỹ và dễ sử dụng.
2. Tích hợp API thời tiết (như OpenWeatherMap API) để lấy dữ liệu thời tiết hiện tại và dự báo trong tương lai (theo giờ, theo ngày).

3. Xử lý dữ liệu thời tiết từ API dưới dạng JSON, chuyển đổi sang mô hình dữ liệu phù hợp trong Java/Kotlin, và hiển thị bằng các thành phần giao diện như RecyclerView, CardView, ViewPager, biểu đồ,…
4. Sử dụng định vị GPS để tự động xác định vị trí hiện tại của người dùng và hiển thị thông tin thời tiết tương ứng.
5. Áp dụng kiến trúc MVVM để phân tách logic nghiệp vụ, giao diện và dữ liệu, giúp ứng dụng dễ mở rộng và bảo trì.
6. Lưu trữ dữ liệu cục bộ (như lịch sử địa điểm, địa điểm yêu thích) sử dụng Room Database.
7. Hỗ trợ cảnh báo thời tiết xấu hoặc các điều kiện đặc biệt (nhiệt độ cao, mưa lớn, gió mạnh,...).
8. Tối ưu hiệu năng ứng dụng, bao gồm: tải dữ liệu nhanh, giảm thiểu tiêu thụ pin và băng thông, quản lý bộ nhớ hiệu quả.
9. Tích hợp các thư viện và công cụ hiện đại như Retrofit, Glide/Picasso, LiveData, ViewModel,… trong quá trình phát triển.

1.3. Phạm vi của đề tài

Đề tài "Dự báo thời tiết" tập trung vào việc thiết kế và phát triển một ứng dụng di động hoạt động trên nền tảng Android, sử dụng ngôn ngữ Java hoặc Kotlin, nhằm cung cấp cho người dùng thông tin thời tiết một cách chính xác và trực quan. Phạm vi của đề tài bao gồm các nội dung chính sau:

1. Nền tảng triển khai:
   - Hệ điều hành Android (API từ 21 trở lên).
   - Thiết bị hỗ trợ: điện thoại và máy tính bảng Android.
   - Ngôn ngữ lập trình: Java hoặc Kotlin.
2. Tính năng chính:
   - Hiển thị thông tin thời tiết hiện tại tại vị trí người dùng.
   - Cho phép tìm kiếm thời tiết tại các thành phố hoặc khu vực khác.
   - Dự báo thời tiết theo giờ, theo ngày (3–5 ngày).

- Cung cấp các chỉ số như: nhiệt độ, độ ẩm, tốc độ gió, áp suất, chỉ số UV,…
- Hiển thị biểu tượng thời tiết và hình ảnh minh họa tương ứng với trạng thái thời tiết.
- Giao diện thân thiện, hỗ trợ chế độ sáng/tối (dark mode).

3. Phạm vi kỹ thuật:
   - Sử dụng API của dịch vụ bên thứ ba (ví dụ: OpenWeatherMap).
   - Áp dụng kiến trúc MVVM.
   - Sử dụng thư viện Retrofit để gọi API và xử lý phản hồi.
   - Sử dụng LiveData và ViewModel để quản lý dữ liệu và vòng đời.
   - Sử dụng Room Database để lưu trữ cục bộ (nếu có).
   - Định vị vị trí người dùng bằng FusedLocationProvider (Google Location Services).
   - Tối ưu hóa hiệu năng: xử lý bất đồng bộ, cache dữ liệu, tiết kiệm pin.
4. Không bao gồm trong phạm vi đề tài:
   - Không xử lý phân tích dữ liệu dự báo chuyên sâu bằng thuật toán học máy.
   - Không xây dựng hệ thống backend riêng mà sử dụng dữ liệu từ dịch vụ thời tiết công khai.
   - Không hỗ trợ đa nền tảng (iOS hoặc web) trong phiên bản hiện tại.

1.4 Phân chia nhiệm vụ

<<Bảng phân chia nhiệm vụ>>

STT	Họ và tên	Nhiệm vụ
1	Lê Đức Hùng	Chịu trách nhiệm về logic xử lý dữ liệu và Firebase Authentication, xử lý sự kiện người dùng
2	Đỗ Trần Hùng	Đảm bảo giao diện app trực quan, đẹp mắt, và tương tác với API OpenWeatherMap và lý sự kiện người dùng
3	Thái Duy Đức	Đảm bảo chất lượng sản phẩm thông qua kiểm thử, quản lý dự án và tích hợp các tính năng nâng cao.

CHƯƠNG 2. KIẾN TRÚC VÀ CÔNG NGHỆ

2.1. Kiến trúc hệ thống

- Ứng dụng "Dự báo thời tiết" được xây dựng theo mô hình client-server, trong đó:

✓ Client là ứng dụng Android do người dùng sử dụng, được phát triển bằng ngôn ngữ Java/Kotlin.

✓ Server là hệ thống cung cấp dữ liệu thời tiết, ví dụ như OpenWeatherMap, thông qua giao tiếp RESTful API.

- Hệ thống sử dụng kiến trúc MVVM (Model – View – ViewModel) để tách biệt ba phần chính:

✓ Model: xử lý dữ liệu từ API và quản lý dữ liệu cục bộ (Room Database).

✓ ViewModel: trung gian giữa View và Model, xử lý logic hiển thị và cung cấp dữ liệu theo vòng đời của UI.

✓ View: giao diện người dùng, được cập nhật tự động thông qua LiveData.

2.2. Giới thiệu về Công nghệ phát triển

Các công nghệ và thư viện được sử dụng trong ứng dụng bao gồm:

- Ngôn ngữ lập trình:

✓ Java hoặc Kotlin – hai ngôn ngữ chính được hỗ trợ trên Android.

- Giao tiếp với API:

✓ Retrofit: thư viện mạnh mẽ dùng để gọi API RESTful và xử lý phản hồi JSON.

✓ Gson/Moshi: dùng để chuyển đổi dữ liệu JSON thành đối tượng Java/Kotlin.

- Quản lý dữ liệu và vòng đời:

✓ LiveData: giúp giao diện tự động cập nhật khi dữ liệu thay đổi.

✓ ViewModel: quản lý dữ liệu và logic tương tác giữa UI và Model.

✓ Room: cơ sở dữ liệu cục bộ để lưu trữ các địa điểm yêu thích, lịch sử.

- Xử lý ảnh:

✓ Glide hoặc Picasso: dùng để tải và hiển thị hình ảnh biểu tượng thời tiết.

- Định vị và vị trí:

✓ FusedLocationProviderClient: lấy vị trí người dùng thông qua Google Play Services.

- Thiết kế giao diện:

✓ Material Design Components: tuân thủ tiêu chuẩn thiết kế giao diện hiện đại của Google.

✓ ConstraintLayout, RecyclerView, ViewPager,...

- Xử lý bất đồng bộ:

✓ Coroutines (Kotlin) hoặc AsyncTask/Executors (Java).

CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG

3.1. Thiết kế Figma

<<Link Gitub và ảnh kết xuất Figma các màn hình>>

Link github: https://github.com/duchungvippro/B-i-t-p-Mobile-Dev

Ảnh kết xuất figma các màn hình:

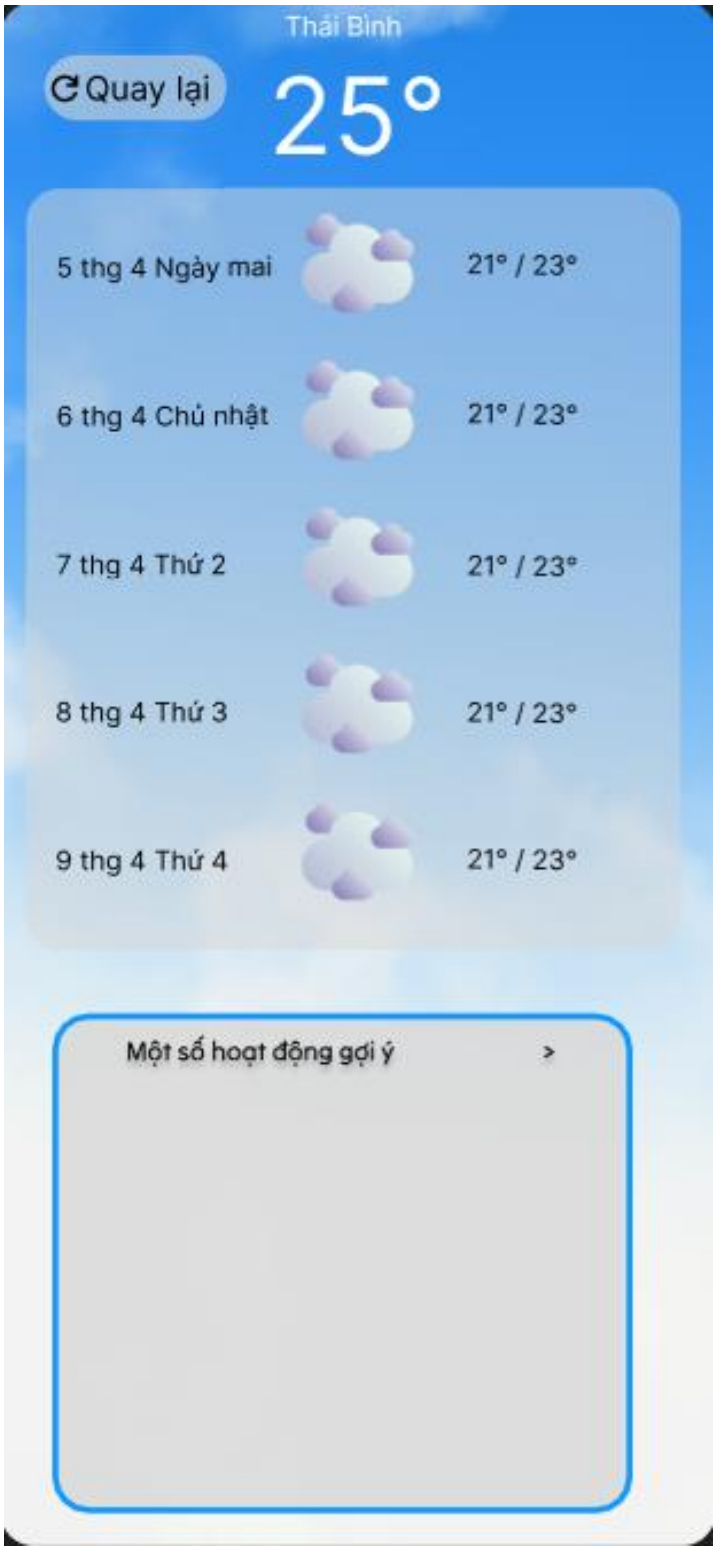
Giao diện đăng nhập:

Giao diện đăng ký:

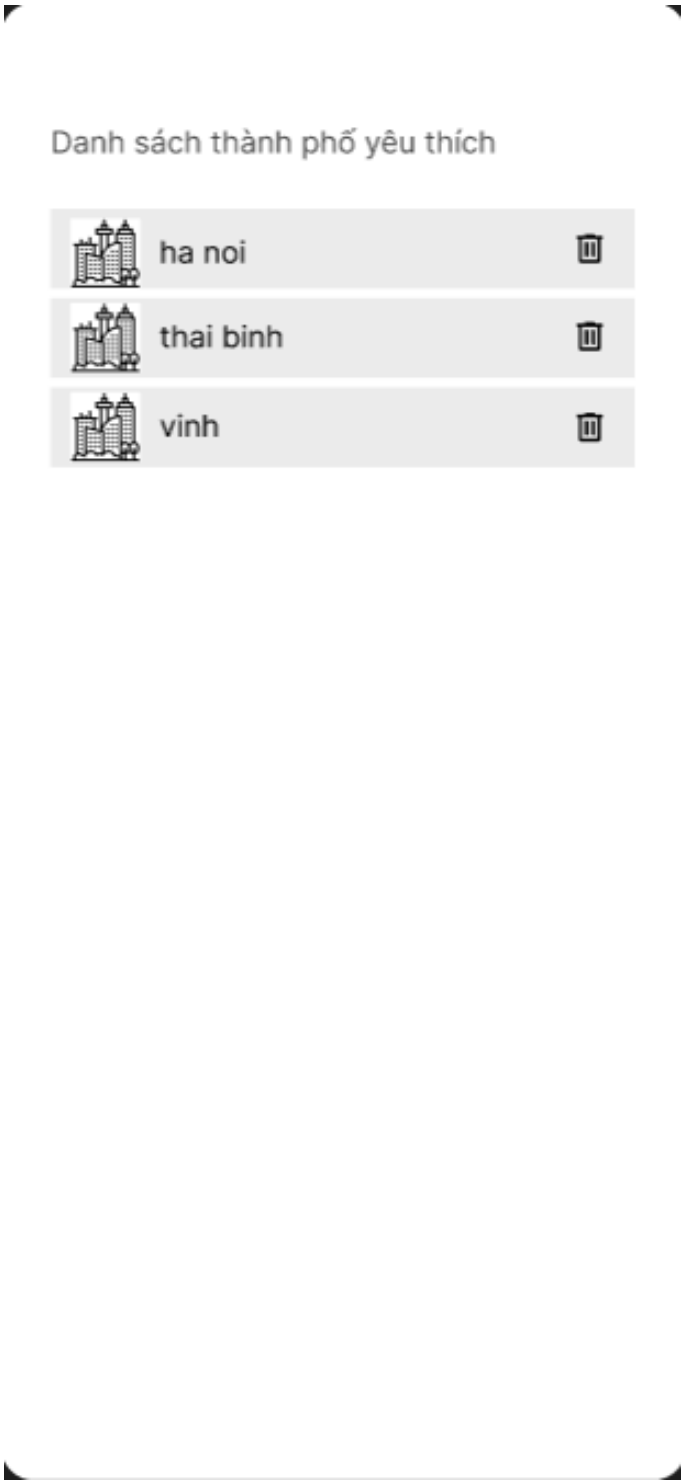
Giao diện tìm kiếm:

Giao diện thời tiết hiện tại:

Giao diện thời tiết 5 ngày tới:

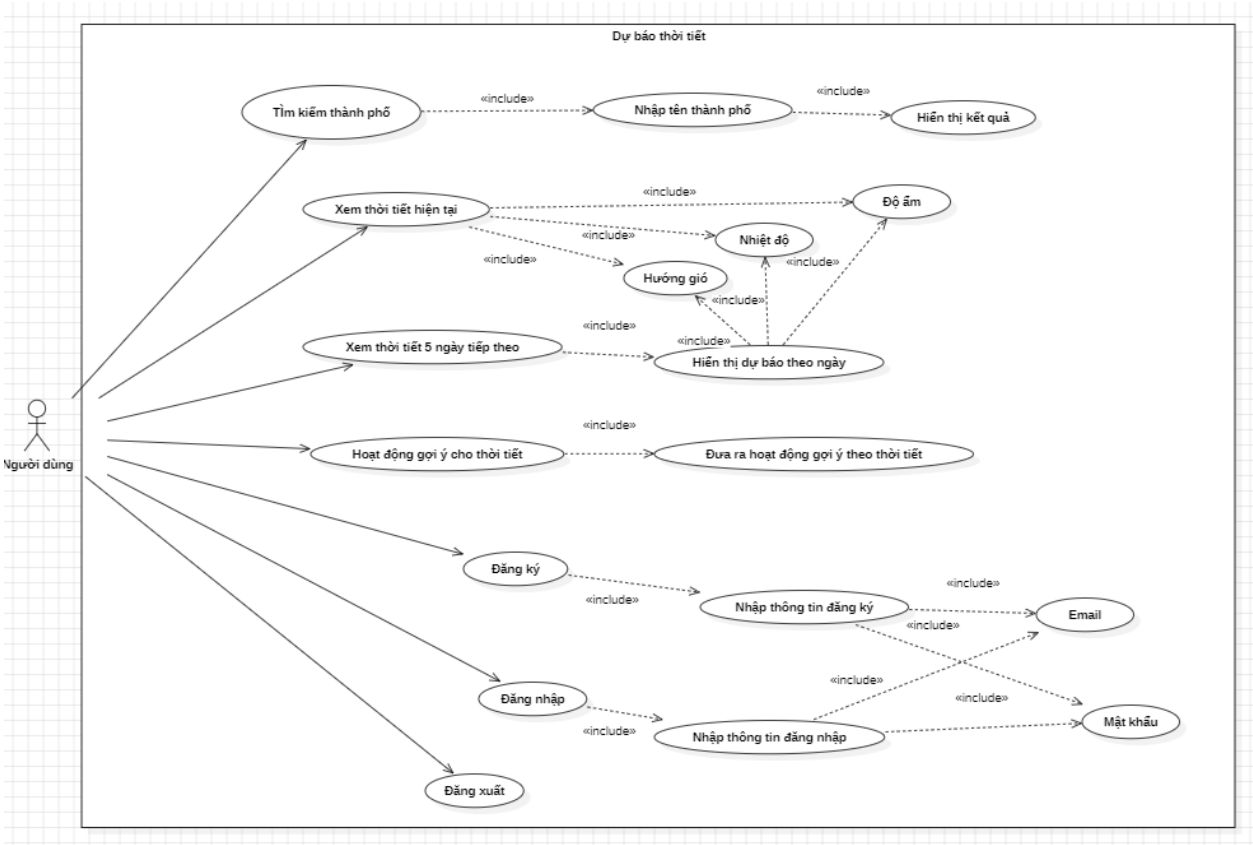


Giao diện các thành phố yêu thích đã lưu:



3.2. Thiết kế CSDL

<<Lược đồ>>



3.3. Giao diện ứng dụng

3.3.1. Giao diện tìm kiếm

- Trường nhập “Nhập tên thành phố”: Đây là nơi người dùng nhập tên thành phố mà họ muốn tra cứu thông tin thời tiết. Trường này hỗ trợ nhập tiếng Việt hoặc tiếng Anh và có thể được tối ưu để gợi ý kết quả khi người dùng gõ.
- Nút “TÌM KIẾM” kèm biểu tượng kính lúp : Sau khi nhập tên thành phố, người dùng bấm vào nút này để thực hiện truy vấn thời tiết. Hệ thống sẽ gửi yêu cầu đến API (ví dụ: OpenWeatherMap) và hiển thị kết quả ở màn hình tiếp theo hoặc bên dưới nếu dùng dạng hiển thị động.

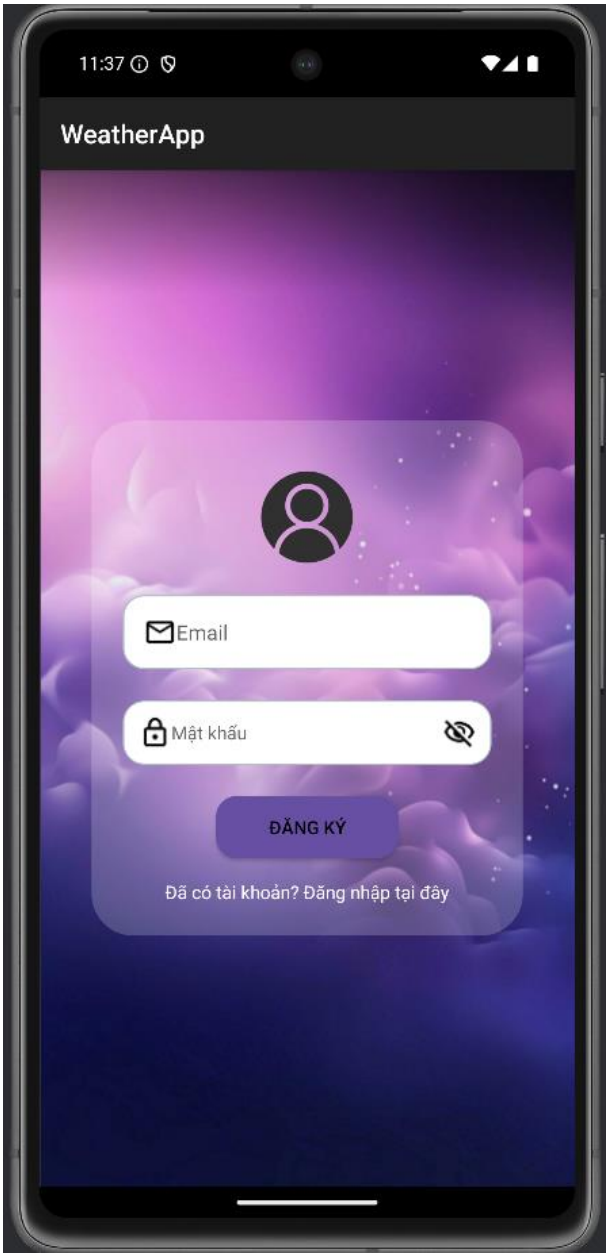
3.3.2. Giao diện đăng nhập

- Biểu tượng người dùng (User icon): Hình ảnh minh họa cho chức năng đăng nhập, tạo cảm giác trực quan và dễ nhận diện.
- Trường nhập Email: Cho phép người dùng nhập địa chỉ email đã đăng ký với ứng dụng. Đây là thông tin bắt buộc để xác định tài khoản cá nhân.
- Trường nhập Mật khẩu: Cho phép người dùng nhập mật khẩu tương ứng với email. Có biểu tượng con mắt (hiển thị/ẩn mật khẩu), giúp người dùng kiểm tra mật khẩu đã nhập để tránh nhầm lẫn.
- Nút "ĐĂNG NHẬP": Khi người dùng đã nhập đúng thông tin email và mật

khẩu, bấm vào nút này để đăng nhập vào hệ thống. Hệ thống sẽ xác thực thông tin và chuyển đến màn hình chính nếu đăng nhập thành công.

- Nút "ĐĂNG KÝ": Dành cho người dùng chưa có tài khoản. Khi bấm vào, ứng dụng sẽ chuyển sang màn hình đăng ký, nơi người dùng có thể tạo một tài khoản mới để sử dụng ứng dụng.

3.3.3. Giao diện đăng ký



- Biểu tượng người dùng (User Icon): Mang tính biểu tượng, giúp người dùng dễ dàng nhận biết đây là màn hình liên quan đến tài khoản.
- Trường nhập Email: Người dùng nhập địa chỉ email cá nhân để đăng ký tài khoản mới. Ứng dụng có thể kiểm tra hợp lệ định dạng email trước khi cho

phép tiếp tục.

- Trường nhập Mật khẩu: Nhập mật khẩu mới cho tài khoản. Bên cạnh có biểu tượng con mắt, giúp người dùng ẩn/hiện mật khẩu trong quá trình nhập liệu để đảm bảo tính chính xác.
- Nút "ĐĂNG KÝ" (màu xanh): Sau khi điền đầy đủ thông tin, người dùng bấm nút này để hoàn tất quá trình tạo tài khoản. Hệ thống sẽ kiểm tra tính hợp lệ, và nếu hợp lệ, sẽ lưu thông tin tài khoản vào Firebase Authentication hoặc một cơ sở dữ liệu khác.
- Liên kết "Đã có tài khoản? Đăng nhập tại đây": Cho phép người dùng đã có tài khoản quay lại màn hình đăng nhập. Khi bấm vào liên kết này, ứng dụng sẽ điều hướng trở lại màn hình login.

3.3.4. Giao diện thời tiết hiện tại

- Tên thành phố (ví dụ: Hanoi): Hiển thị địa điểm hiện tại đang được theo dõi/thể hiện thông tin thời tiết.
- Dòng mô tả thời tiết (ví dụ: overcast clouds): Cung cấp thông tin mô tả chung về trạng thái thời tiết hiện tại (mây mù, nắng, mưa,...).
- Nhiệt độ hiện tại (ví dụ: 28°C): Hiển thị nổi bật tại trung tâm màn hình với font chữ lớn, giúp người dùng dễ quan sát.
- Thông tin gió và độ ẩm: Hai biểu tượng bên trái và phải thể hiện dữ liệu phụ như tốc độ gió và độ ẩm không khí.
- Thanh cuộn thời gian theo giờ: Hiển thị dự báo thời tiết chi tiết theo từng khung giờ trong ngày, bao gồm:
- Thời gian (ví dụ: 12:00, 15:00,...)
- Ngày (ví dụ: 2025-04-14)
- Biểu tượng trạng thái thời tiết (ví dụ: mây)
- Nhiệt độ tương ứng
- Nút "XEM 5 NGÀY TỚI": Khi nhấn vào, người dùng sẽ được chuyển đến màn hình hiển thị dự báo thời tiết trong 5 ngày sắp tới.
- Nút "Menu": Khi ấn vào sẽ hiện râ 3 nút:

  + Nút "ĐĂNG XUẤT": Cho phép người dùng thoát khỏi tài khoản hiện tại.

  + Nút "❤THÊM YÊU THÍCH": Lưu lại thành phố hiện tại vào danh sách các địa điểm yêu thích của người dùng.

  + Nút "⭐ YÊU THÍCH ĐÃ LƯU": Điều hướng người dùng đến màn hình hiển thị danh sách các thành phố yêu thích đã lưu.

3.3.5. Giao diện thời tiết 5 ngày tới

- Nút "QUAY LẠI": Nằm ở góc trên bên trái màn hình. Khi nhấn vào, người dùng sẽ được đưa trở lại màn hình hiển thị thời tiết hiện tại.
- Tên thành phố (ví dụ: Ha Noi): Hiển thị địa điểm mà người dùng đang theo dõi thông tin thời tiết.
- Nhiệt độ hiện tại (ví dụ: 28°C): Hiển thị nhiệt độ thời điểm hiện tại ở trung tâm màn hình.
- Danh sách dự báo thời tiết 5 ngày tiếp theo:
- Hiển thị theo định dạng ngày – nhiệt độ – biểu tượng thời tiết.

- Các biểu tượng thể hiện tình trạng thời tiết: mây, mưa, nắng,... giúp người dùng dễ hình dung.
- Gợi ý hoạt động: Phía cuối màn hình là danh sách các hoạt động phù hợp dựa trên điều kiện thời tiết

3.3.6. Giao diện các thành phố yêu thích đã lưu

- Dòng chữ "Danh sách thành phố yêu thích": Là tiêu đề của phần nội dung chính, cho biết đây là nơi hiển thị các địa điểm mà người dùng đã đánh dấu yêu thích để dễ dàng tra cứu lại thời tiết.
- Danh sách các thành phố đã lưu:

- Mỗi mục gồm:
  - ✓ Biểu tượng thành phố: Giúp người dùng dễ dàng nhận diện trực quan.
  - ✓ Tên thành phố (ví dụ: Thanh Hóa): Cho biết địa điểm được lưu.

    Nút xoá : Cho phép người dùng loại bỏ thành phố khỏi danh sách yêu thích nếu không còn nhu cầu theo dõi nữa.
- Khi nhấn vào tên thành phố, ứng dụng có thể chuyển đến màn hình hiển thị thời tiết hiện tại của địa điểm đó.
- Tác dụng của tính năng yêu thích:
- Tiết kiệm thời gian khi tra cứu lại các địa điểm quen thuộc.
- Tiện lợi trong việc theo dõi thời tiết của nhiều khu vực khác nhau (phù hợp cho người thường xuyên di chuyển, có người thân ở xa...).

3.4. Code minh họa các chức năng cốt lõi

3.4.1. Đăng nhập

```
package com.example.weatherapp


import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import android.text.InputType
import android.util.Patterns
import android.view.MotionEvent
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.weatherapp.databinding.ActivityLoginBinding
import com.google.firebase.auth.FirebaseAuth


class LoginActivity : AppCompatActivity() {


    private lateinit var binding: ActivityLoginBinding
    private lateinit var auth: FirebaseAuth
    private var isPasswordVisible = false


    @SuppressLint("ClickableViewAccessibility")
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)

        auth = FirebaseAuth.getInstance()

        // Xử lý ẩn/hiện mật khẩu bằng icon con mắt
        binding.etPassword.setOnTouchListener { _, event ->
            val drawableEnd = binding.etPassword.compoundDrawables[2]
            if (event.action == MotionEvent.ACTION_UP && drawableEnd != null) {
                val x = event.x.toInt()
                if (x >= binding.etPassword.width - binding.etPassword.paddingRight -
drawableEnd.intrinsicWidth) {
                    isPasswordVisible = !isPasswordVisible
                    if (isPasswordVisible) {
                        binding.etPassword.inputType = InputType.TYPE_CLASS_TEXT
                        binding.etPassword.setCompoundDrawablesWithIntrinsicBounds(
                            null, null, getDrawable(R.drawable.eye), null
                        )
                    } else {
                        binding.etPassword.inputType = InputType.TYPE_CLASS_TEXT or
InputType.TYPE_TEXT_VARIATION_PASSWORD
                        binding.etPassword.setCompoundDrawablesWithIntrinsicBounds(
                            null, null, getDrawable(R.drawable.eyeoff), null
                        )
                    }
                    binding.etPassword.setSelection(binding.etPassword.text.length)
                    return@setOnTouchListener true
                }
            }
            false
        }

       // Nút đăng nhập
        binding.btnLogin.setOnClickListener {
            val email = binding.etEmail.text.toString().trim()
            val password = binding.etPassword.text.toString().trim()

            if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
                binding.etEmail.error = "Email không hợp lệ"
                return@setOnClickListener
            }
```

```
            if (password.length < 6) {
                binding.etPassword.error = "Mật khẩu phải từ 6 ký tự"
                return@setOnClickListener
            }

            auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        startActivity(Intent(this, MainActivity::class.java))
                        finish()
                    } else {
                        Toast.makeText(this, "Đăng nhập thất bại:
${task.exception?.message}", Toast.LENGTH_SHORT).show()
                    }
                }
        }

        // Nút chuyển sang màn hình đăng ký
        binding.btnRegister.setOnClickListener {
            startActivity(Intent(this, RegisterActivity::class.java))
        }
    }
}
```

3.4.2. Đăng ký

```
package com.example.weatherapp

import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import android.text.InputType
import android.widget.Toast
import android.widget.ImageView
import android.widget.EditText
import android.view.MotionEvent
import androidx.appcompat.app.AppCompatActivity
import com.example.weatherapp.databinding.ActivityRegisterBinding
import com.google.firebase.auth.FirebaseAuth
```

```
class RegisterActivity : AppCompatActivity() {
    private lateinit var binding: ActivityRegisterBinding
    private lateinit var auth: FirebaseAuth
    private var isPasswordVisible = false  // Đưa biến này ra ngoài onCreate

    @SuppressLint("ClickableViewAccessibility")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityRegisterBinding.inflate(layoutInflater)
        setContentView(binding.root)

        auth = FirebaseAuth.getInstance()

        binding.etRegisterPassword.setOnTouchListener { _, event ->
            val drawableEnd = binding.etRegisterPassword.compoundDrawables[2]
            if (event.action == MotionEvent.ACTION_UP && drawableEnd != null) {
                val x = event.x.toInt()
                if (x >= binding.etRegisterPassword.width -
binding.etRegisterPassword.paddingRight - drawableEnd.intrinsicWidth) {
                    isPasswordVisible = !isPasswordVisible
                    if (isPasswordVisible) {
                        // Hiển thị mật khẩu
                        binding.etRegisterPassword.inputType = InputType.TYPE_CLASS_TEXT

binding.etRegisterPassword.setCompoundDrawablesWithIntrinsicBounds(
                            null, null, getDrawable(R.drawable.eye), null
                        )
                    } else {
                        // Ẩn mật khẩu
                        binding.etRegisterPassword.inputType = InputType.TYPE_CLASS_TEXT
or InputType.TYPE_TEXT_VARIATION_PASSWORD

binding.etRegisterPassword.setCompoundDrawablesWithIntrinsicBounds(
                            null, null, getDrawable(R.drawable.eyeoff), null
                        )
                    }
                    // Đặt lại con trỏ ở cuối nội dung

binding.etRegisterPassword.setSelection(binding.etRegisterPassword.text.length)
                    return@setOnTouchListener true
                }
            }
```

```
            false
        }

        binding.btnDoRegister.setOnClickListener {
            val email = binding.etRegisterEmail.text.toString().trim()
            val password = binding.etRegisterPassword.text.toString().trim()

            if (!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
                binding.etRegisterEmail.error = "Email không hợp lệ"
                return@setOnClickListener
            }

            if (password.length < 6) {
                binding.etRegisterPassword.error = "Mật khẩu phải từ 6 ký tự"
                return@setOnClickListener
            }

            // Đăng ký tài khoản
            auth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        Toast.makeText(this, "Đăng ký thành công",
Toast.LENGTH_SHORT).show()
                        startActivity(Intent(this, LoginActivity::class.java))
                        finish()
                    } else {
                        Toast.makeText(this, "Đăng ký thất bại:
${task.exception?.message}", Toast.LENGTH_SHORT).show()
                    }
                }
        }

        // Xử lý khi bấm vào dòng "Đã có tài khoản? Đăng nhập tại đây"
        binding.tvGoToLogin.setOnClickListener {
            startActivity(Intent(this, LoginActivity::class.java))
            finish()
        }
    }
}
```

3.4.3. Dự báo thời tiết theo giờ

```
package com.example.weatherapp

data class ForecastResponse(
    val list: List<ForecastItem>
)

data class ForecastItem(
    val dt_txt: String,
    val main: ForecastMain,
    val weather: List<HourlyWeather> // ✅Dùng HourlyWeather, không dùng Weather trùng
)

data class ForecastMain(
    val temp: Float
)

data class HourlyWeather(
    val description: String,
    val icon: String
)
```

3.4.4. Dự báo thời tiết 5 ngày, hoạt động gợi ý

```
package com.example.weatherapp

import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.weatherapp.databinding.ActivityFiveDaysForecastBinding
import java.util.Calendar



class FiveDaysForecastActivity : AppCompatActivity() {

    private lateinit var binding: ActivityFiveDaysForecastBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        binding = ActivityFiveDaysForecastBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val forecastList = intent.getSerializableExtra("forecastList") as?
ArrayList<DailyForecast>
        val cityName = intent.getStringExtra("city_name")
        val currentTemp = intent.getStringExtra("current_temp")

        if (forecastList == null) {
            Toast.makeText(this, "Không có dữ liệu dự báo thời tiết",
Toast.LENGTH_SHORT).show()
            return
        }

        // Gán tên thành phố
        binding.tvLocation.text = cityName

        // Gán nhiệt độ hiện tại
        binding.tvCurrentTemp.text = currentTemp

        // Gợi ý hoạt động theo nhiệt độ
        val tempValue = currentTemp?.replace("°", "")?.toIntOrNull() ?: 0
        val activitySuggestions = suggestActivity(tempValue)
        val suggestionText = activitySuggestions.joinToString(separator = "\n") { "• $it"
}
        binding.tvSuggestion.text = suggestionText

        // Thiết lập danh sách 5 ngày
        binding.recyclerViewFiveDays.layoutManager = LinearLayoutManager(this)
        binding.recyclerViewFiveDays.adapter = FiveDaysAdapter(forecastList)

        // Nút quay lại
        binding.btnBack.setOnClickListener {
            finish()
        }
    }



   private fun suggestActivity(tempC: Int): List<String> {
        // Lấy thời gian hiện tại
```

```
        val currentHour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY)

        // Kiểm tra xem đó là ban ngày hay ban đêm
        val isDaytime = currentHour in 6..18 // Từ 6 AM đến 6 PM là ban ngày, còn lại là
ban đêm

        return when {
            tempC < 10 -> {
                if (isDaytime) {
                    // Ban ngày
                    listOf(
                        "Đi cà phê với bạn bè",
                        "Dạo phố",
                        "Chụp ảnh ngoài trời"
                    )
                } else {
                    // Ban đêm
                    listOf(
                        "Đọc sách trong nhà",
                        "Uống trà nóng",
                        "Xem phim",
                        "Nghe nhạc thư giãn"
                    )
                }
            }
            tempC in 10..20 -> {
                if (isDaytime) {
                    // Ban ngày
                    listOf(
                        "Đi dạo công viên",
                        "Uống café với bạn bè",
                        "Chụp ảnh ngoài trời",
                        "Tản bộ cùng thú cưng"
                    )
                } else {
                    // Ban đêm
                    listOf(
                        "Đi dạo dưới ánh đèn đường",
                        "Ngồi quán cà phê ngoài trời",
                        "Thư giãn tại nhà với âm nhạc"
                    )
                }
```

```
            }
            tempC in 21..30 -> {
                if (isDaytime) {
                    // Ban ngày
                    listOf(
                        "Chơi thể thao ngoài trời",
                        "Dã ngoại hoặc picnic",
                        "Đi bơi",
                        "Lướt ván/Chơi bóng chuyền bãi biển"
                    )
                } else {
                    // Ban đêm
                    listOf(
                        "Xem phim ngoài trời",
                        "Tham gia các hoạt động vui chơi nhẹ nhàng"
                    )
                }
            }
            else -> {
                if (isDaytime) {
                    // Ban ngày
                    listOf(
                        "Ở trong phòng máy lạnh",
                        "Đi bơi hoặc công viên nước",
                        "Tránh vận động mạnh ngoài trời"
                    )
                } else {
                    // Ban đêm
                    listOf(
                        "Nghỉ ngơi trong phòng máy lạnh",
                        "Uống nước mát và thư giãn"
                    )
                }
            }
        }
    }

}
```

3.4.5. Thêm thành phố yêu thích

```
private fun addCity(cityName: String) {
    val newCity = FavoriteCity(cityName)

    // Thêm thành phố vào database
    CoroutineScope(Dispatchers.IO).launch {

AppDatabase.getDatabase(this@FavoriteCitiesActivity).favoriteCityDao().insert(newCity)

        // Cập nhật lại danh sách thành phố yêu thích
        withContext(Dispatchers.Main) {
            cities.add(newCity)
            adapter.notifyItemInserted(cities.size - 1) // Cập nhật RecyclerView
        }
    }
}
```

3.4.6. Xóa thành phố yêu thích

```
private fun deleteCity(city: FavoriteCity) {
    // Xóa thành phố khỏi database
    CoroutineScope(Dispatchers.IO).launch {

AppDatabase.getDatabase(this@FavoriteCitiesActivity).favoriteCityDao().delete(city)

        // Cập nhật lại danh sách thành phố yêu thích
        withContext(Dispatchers.Main) {
            cities.remove(city)
            adapter.notifyDataSetChanged() // Cập nhật RecyclerView
        }
    }
}
```

3.4.7. Thời tiết hiện tại

```
package com.example.weatherapp

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.view.View
```

```
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.weatherapp.databinding.ActivityMainBinding
import com.google.firebase.auth.FirebaseAuth
import retrofit2.*
import retrofit2.converter.gson.GsonConverterFactory
import com.example.weatherapp.data.AppDatabase
import com.example.weatherapp.data.FavoriteCity
import kotlinx.coroutines.*



class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private lateinit var auth: FirebaseAuth
    private val apiKey = "d754023544590c6ced61ca8f5582dce3"
    private val fiveDaysForecastData = mutableListOf<DailyForecast>()
    private var currentSearchedCity: String = ""

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val cityFromIntent = intent.getStringExtra("city_name")
        if (!cityFromIntent.isNullOrEmpty()) {
            binding.cityInput.setText(cityFromIntent)
            // Delay nhẹ để giao diện load xong rồi mới click
            binding.cityInput.post {
                binding.btnSearch.performClick()
            }
        }



        intent.getStringExtra("city_name")?.let { city ->
            binding.cityInput.setText(city)
            binding.btnSearch.performClick()
        }

        binding.btnFavorites.setOnClickListener {
```

```
            val intent = Intent(this, FavoriteCitiesActivity::class.java)
            startActivity(intent)
        }




        binding.btnAddToFavorite.visibility = View.VISIBLE

        binding.btnAddToFavorite.setOnClickListener {
            if (currentSearchedCity.isNotEmpty()) {
                CoroutineScope(Dispatchers.IO).launch {
                    AppDatabase.getDatabase(this@MainActivity)
                        .favoriteCityDao()
                        .insert(FavoriteCity(currentSearchedCity))
                }
                Toast.makeText(this, "$currentSearchedCity đã được thêm vào yêu thích",
Toast.LENGTH_SHORT).show()
            }
        }

// Khởi tạo Firebase Auth
auth = FirebaseAuth.getInstance()

// Nếu chưa đăng nhập thì chuyển đến LoginActivity
if (auth.currentUser == null) {
    val intent = Intent(this, LoginActivity::class.java)
    startActivity(intent)
    finish()
    return
}

// Nút logout
binding.btnLogout.setOnClickListener {
    auth.signOut()
    val intent = Intent(this, LoginActivity::class.java)
    startActivity(intent)
    finish()
}

val retrofit = Retrofit.Builder()
    .baseUrl("https://api.openweathermap.org/data/2.5/")
```

```
    .addConverterFactory(GsonConverterFactory.create())
    .build()

val service = retrofit.create(WeatherService::class.java)

binding.recyclerHourly.layoutManager = LinearLayoutManager(this,
LinearLayoutManager.HORIZONTAL, false)
showSearchUI(true)

binding.btnSearch.setOnClickListener {
    val city = binding.cityInput.text.toString().trim()
    if (city.isEmpty()) {
        Toast.makeText(this, getString(R.string.enter_city), Toast.LENGTH_SHORT).show()
        return@setOnClickListener
    }

    val lang = resources.configuration.locales.get(0).language
    currentSearchedCity = city

    service.getWeatherByCity(city, apiKey, lang = lang)
        .enqueue(object : Callback<WeatherResponse> {
            override fun onResponse(
                call: Call<WeatherResponse>,
                response: Response<WeatherResponse>
            ) {
                if (response.isSuccessful && response.body() != null) {
                    val data = response.body()!!

                    binding.tvCity.text = data.name
                    binding.tvDescription.text = data.weather[0].description
                    binding.tvTemp.text = "${data.main.temp.toInt()}°"
                    binding.tvTempMax.text = "↑${data.main.tempMax.toInt()}°"
                    binding.tvTempMin.text = "↓${data.main.tempMin.toInt()}°"

                    setBackgroundByWeather(data.weather[0].description)
                    showSearchUI(false)

                    service.getHourlyForecast(city, apiKey, lang = lang)
                        .enqueue(object : Callback<ForecastResponse> {
                            override fun onResponse(
                                call: Call<ForecastResponse>,
```

```
                                response: Response<ForecastResponse>
                            ) {
                                if (response.isSuccessful) {
                                    val forecastList = response.body()?.list?.take(5) ?:
return
                                    val adapter = HourlyAdapter(forecastList)
                                    binding.recyclerHourly.adapter = adapter
                                }
                            }

                override fun onFailure(call: Call<ForecastResponse>, t: Throwable) {
                    Log.e("FORECAST", "Error loading hourly forecast", t)
                }
            })

        binding.btnFiveDay.setOnClickListener {
            if (fiveDaysForecastData.isNotEmpty()) {
                val intent = Intent(this@MainActivity,
FiveDaysForecastActivity::class.java)
                intent.putExtra("forecastList", ArrayList(fiveDaysForecastData))
                intent.putExtra("city_name", currentSearchedCity)
                intent.putExtra("current_temp", binding.tvTemp.text.toString())

                startActivity(intent)
            } else {
                Toast.makeText(this@MainActivity, getString(R.string.city_not_found),
Toast.LENGTH_SHORT).show()
            }
        }

        loadFiveDayForecast(city)
    } else {
        Toast.makeText(this@MainActivity, getString(R.string.city_not_found),
Toast.LENGTH_SHORT).show()
    }
}

                override fun onFailure(call: Call<WeatherResponse>, t: Throwable) {
                    Log.e("WEATHER_API_ERROR", "Failed API call", t)
                    Toast.makeText(this@MainActivity, getString(R.string.api_error),
Toast.LENGTH_SHORT).show()
```

```
                }
            })
    }

    binding.btnAddCity.setOnClickListener {
        showSearchUI(true)
        binding.cityInput.setText("")
    }
}

private fun showSearchUI(show: Boolean) {
    binding.cityInput.visibility = if (show) View.VISIBLE else View.GONE
    binding.btnSearch.visibility = if (show) View.VISIBLE else View.GONE

    binding.tvCity.visibility = if (show) View.GONE else View.VISIBLE
    binding.tvDescription.visibility = if (show) View.GONE else View.VISIBLE
    binding.tvTemp.visibility = if (show) View.GONE else View.VISIBLE
    binding.tvTempMax.visibility = if (show) View.GONE else View.VISIBLE
    binding.tvTempMin.visibility = if (show) View.GONE else View.VISIBLE

    binding.hourlyContainer.visibility = if (show) View.GONE else View.VISIBLE
    binding.btnAddCity.visibility = if (show) View.GONE else View.VISIBLE

    binding.iconWind.visibility = if (show) View.GONE else View.VISIBLE
    binding.iconHumidity.visibility = if (show) View.GONE else View.VISIBLE
}
   private fun setBackgroundByWeather(description: String) {
        val desc = description.lowercase()
        val bgRes = when {
            "clear" in desc || "sun" in desc || "nắng" in desc -> R.drawable.bg_day
            "rain" in desc || "drizzle" in desc || "mưa" in desc -> R.drawable.bg_rain
            "cloud" in desc || "mây" in desc -> R.drawable.bg_clouds
            else -> R.drawable.bg_night
        }
        binding.bgImage.setImageResource(bgRes)
    }

    private fun loadFiveDayForecast(city: String) {
        val retrofit = Retrofit.Builder()
            .baseUrl("https://api.openweathermap.org/data/2.5/")
            .addConverterFactory(GsonConverterFactory.create())
```

```
            .build()

        val service = retrofit.create(WeatherService::class.java)
        val lang = resources.configuration.locales.get(0).language

        service.getFiveDayForecast(city, apiKey, lang = lang).enqueue(object :
Callback<ForecastResponse> {
            override fun onResponse(call: Call<ForecastResponse>, response:
Response<ForecastResponse>) {
                if (response.isSuccessful && response.body() != null) {
                    val allForecasts = response.body()!!.list

                    val dailyList = mutableListOf<DailyForecast>()
                    for (item in allForecasts) {
                        val date = item.dt_txt.substring(0, 10)
                        if (dailyList.none { it.date == date } && dailyList.size < 5) {
                            dailyList.add(
                                DailyForecast(
                                    date = date,
                                    temp = item.main.temp,
                                    icon = item.weather[0].icon
                                )
                            )
                        }
                    }

                    fiveDaysForecastData.clear()
                    fiveDaysForecastData.addAll(dailyList)
                }
            }

            override fun onFailure(call: Call<ForecastResponse>, t: Throwable) {
                Log.e("FIVEDAY", "Error loading 5 day forecast", t)
            }
        })
    }
}
```

KẾT LUẬN

1. Kết quả đạt được

Qua quá trình tìm hiểu và thực hiện đề tài "Ứng dụng dự báo thời tiết trên Android", nhóm đã đạt được nhiều kết quả thiết thực. Ứng dụng đã đáp ứng được các chức năng cốt lõi như: tra cứu thời tiết hiện tại, hiển thị dự báo 5 ngày theo từng thành phố, trực quan hóa lượng mưa bằng biểu đồ, và lưu lại lịch sử các địa điểm đã tìm kiếm.

Thông qua đề tài này, nhóm không chỉ củng cố kiến thức về lập trình Android với Kotlin, mô hình giao tiếp mạng qua API (Retrofit), mà còn tiếp cận và ứng dụng hiệu quả các công nghệ hiện đại như Firebase Authentication, Realtime Database và thư viện MPAndroidChart. Bên cạnh đó, nhóm đã cũng nâng cao được kỹ năng làm việc nhóm, kỹ năng thiết kế giao diện người dùng thân thiện, trực quan.

Tuy nhiên, do giới hạn về thời gian và kinh nghiệm, ứng dụng vẫn còn một số điểm cần cải thiện như: giao diện chưa tối ưu trên tất cả thiết bị, thiếu tính năng tự động định vị vị trí hiện tại của người dùng, và chưa hỗ trợ thông báo thời tiết khẩn cấp. Dù vậy, nhóm tin rằng ứng dụng đã đạt được mục tiêu đề ra ban đầu: cung cấp một công cụ tiện lợi, dễ sử dụng, giúp người dùng theo dõi và lập kế hoạch phù hợp với điều kiện thời tiết hàng ngày.

2. Nhược điểm

Mặc dù ứng dụng “Dự báo thời tiết” đã đáp ứng được các chức năng cơ bản và mang lại trải nghiệm thuận tiện cho người dùng, tuy nhiên trong quá trình xây dựng và vận hành, vẫn còn tồn tại một số nhược điểm và hạn chế sau:

1. Phụ thuộc vào API bên thứ ba

Ứng dụng sử dụng dữ liệu thời tiết từ dịch vụ công khai như OpenWeatherMap API, vì vậy:

- Nếu API bị giới hạn truy cập, lỗi hoặc thay đổi cấu trúc dữ liệu, ứng dụng sẽ không thể hoạt động bình thường.

- Tài khoản miễn phí bị giới hạn số lượng request/ngày, ảnh hưởng khi lượng người dùng tăng cao.

2. Chưa hỗ trợ ngoại tuyến (offline)
   - Ứng dụng yêu cầu kết nối Internet để lấy dữ liệu thời tiết mới.
   - Trong điều kiện không có mạng, ứng dụng không thể hiển thị thông tin thời tiết hoặc chỉ hiển thị dữ liệu cũ (nếu được cache lại).
3. Giao diện còn đơn giản
   - Dù đã áp dụng Material Design, nhưng một số thành phần giao diện chưa thực sự sinh động hoặc hấp dẫn với người dùng (chưa có animation, hiệu ứng mượt,…).
4. Thiếu tính năng mở rộng nâng cao
   - Ứng dụng hiện tại chỉ tập trung vào thông tin cơ bản như nhiệt độ, độ ẩm, gió,...
   - Chưa hỗ trợ các tiện ích bổ sung như: cảnh báo thời tiết nguy hiểm, biểu đồ trực quan, tích hợp bản đồ mưa, radar,…
5. Chưa tối ưu pin và dữ liệu mạng
   - Việc cập nhật liên tục dữ liệu thời tiết trong khoảng thời gian ngắn có thể gây hao pin hoặc tốn dữ liệu nếu không được tối ưu.
6. Chưa hỗ trợ đa nền tảng
   - Ứng dụng hiện chỉ chạy trên hệ điều hành Android, chưa có phiên bản dành cho iOS hoặc web, do đó giới hạn đối tượng người dùng.

3. Hướng phát triển

Nhằm khắc phục các nhược điểm còn tồn tại và nâng cao chất lượng ứng dụng, trong tương lai nhóm phát triển hướng đến việc cải tiến và mở rộng ứng dụng “Dự báo thời tiết” theo các hướng sau:

1. Nâng cấp giao diện người dùng (UI/UX)
   - Thiết kế lại giao diện với nhiều hiệu ứng chuyển động mượt mà, biểu tượng sinh động và dễ nhận diện.
   - Cập nhật chế độ Dark Mode linh hoạt theo thời gian thực (ngày/đêm).

- Tối ưu giao diện cho các kích thước màn hình khác nhau (điện thoại, tablet).

2. Hỗ trợ hoạt động ngoại tuyến (Offline Mode)
   - Lưu trữ dữ liệu thời tiết gần nhất bằng Room Database để có thể hiển thị khi mất kết nối mạng.
   - Cho phép người dùng xem lại lịch sử thời tiết đã tra cứu.
3. Thêm tính năng cảnh báo thời tiết khẩn cấp
   - Hiển thị thông báo khi có hiện tượng thời tiết nguy hiểm như: mưa lớn, giông bão, nắng nóng cực đoan,…
   - Đồng bộ với hệ thống cảnh báo quốc gia (nếu có API hỗ trợ).
4. Tích hợp bản đồ và biểu đồ trực quan
   - Sử dụng Google Maps API để hiển thị tình trạng mưa và nhiệt độ theo khu vực.
   - Thêm biểu đồ dự báo nhiệt độ, độ ẩm, áp suất theo giờ và theo ngày.
5. Hỗ trợ đa ngôn ngữ và đa nền tảng
   - Cung cấp giao diện song ngữ (Việt – Anh), có thể mở rộng sang các ngôn ngữ khác.
   - Xây dựng phiên bản ứng dụng cho iOS và web để mở rộng lượng người dùng hơn.
6. Tối ưu hiệu năng và tiết kiệm tài nguyên
   - Giảm tần suất gọi API không cần thiết.
   - Sử dụng kỹ thuật cache thông minh, cập nhật theo thời gian thực một cách linh hoạt.
   - Áp dụng công nghệ Firebase để gửi thông báo, lưu trữ dữ liệu người dùng (nếu cần).
7. Ứng dụng trí tuệ nhân tạo (AI) trong tương lai
   - Phân tích thói quen người dùng để gợi ý thông tin thời tiết phù hợp.
   - Dự đoán xu hướng thời tiết bằng mô hình học máy (machine learning) nếu có đủ dữ liệu.

TÀI LIỆU THAM KHẢO

[1] David Flanagan, JavaScript: The Definitive Guide, 7th Edition, O'Reilly Media, 2020.

[2] Adam Freeman, “Pro jQuery”, Apress, 2018.

[3] Benjamin Jakobus, “Mastering Bootstrap 5”, Packt Publishing, 2018.