1. App Features & Design Decisions

1. **User Authentication**
    - **What**: Email/password sign-up & login via Firebase Auth.
    - **Why**: Offloads secure credential management to Firebase; integrates seamlessly with Firestore for per-user data. Allows user access across multiple devices.
    - **Design**:
        - `AuthViewModel` exposes an `AuthState` LiveData.
        - `HomePage` observes `AuthState` and redirects to “login” if unsigned.
        - Sign-out button calls `authViewModel.signout()`.
2. **Weather Forecast**
    - **What**: 5-day forecast for the map’s pinned location.
    - **Why**: Gives users more access to information regarding their trip
    - **Design**:
        - Separate `WeatherScreen` composable with `WeatherViewModel` (MVVM).
        - Fetches from Google Weather API (`/v1/forecast/days:lookup`) using OkHttp + coroutines.
        - Robust error handling: HTTP-status checks, JSON-field validation.
        - UI: `LazyColumn` of `Card` items showing date, description, high/low.
3. **Map Search**
    - **What**:
        - **Place-level** autocomplete search (single pin).
        - **City-level** search returning multiple markers, filterable by category (museums, restaurants, attractions).
    - **Why**: Enables exploratory and targeted discovery.
    - **Design**:
        - `MapWithSearchScreen` wraps a `MapView` in Compose via `AndroidView`.
        - Uses Google Places SDK for Autocomplete predictions and Place Details.
        - On map-click or search-selection, calls back `HomePage` with `LatLng`.

        - (City-level search & category filters: requires implementing Places "Nearby Search" or "Text Search" endpoints; please specify which categories/API calls you prefer.)
4. **Auto-Recommendation**
    - **What**: Suggests nearby places based on the user's current GPS location and selected category/distance.
    - **Why**: Quick "Nearby Places" functionality.
    - **Design**:
        - FusedLocationProvider to get real-time location.
        - Query Places Nearby Search with `location=…&radius=…&type=[category]`.
        - Display results in a scrollable list of cards with name, icon, and distance.
        - (Need to know: default radius values, exact category list, and UI layout for recommendation list.)
5. **Settings**
    - **What**:
        - Font-size adjustment.
        - Dark mode toggle.
        - 12h/24h time format switch.
    - **Why**: Improves accessibility and user comfort.
    - **Design**:
        - A Compose "Settings" screen backed by DataStore (or `SharedPreferences`).
        - UI controls: Slider for font size, Switch for dark/light, Toggle for time format.
        - App theme and Text sizes respond to stored preferences at startup.

Architecture/Technologies used

1. Google Weather API
2. Google Places/Maps API

Layer	**Technology & Why**
UI	Jetpack Compose – declarative, concise UI code; Material 3 components for consistent styling.
Navigation	Navigation-Compose – type-safe, parameterized routes (`"weather/{lat}/{lon}"`).

State & Logic	MVVM (ViewModel + LiveData) – separation of UI & business logic; `WeatherViewModel`, `AuthViewModel`.
Networking	OkHttp + Coroutines – lightweight HTTP client; async threading via `viewModelScope.launch {…}`.
JSON Parsing	org.json (`JSONObject`) – simple one-off parsing of Google Weather API responses.
Maps & Places	Google Maps SDK for Android & Places SDK – rich map UI, autocomplete, place details.
Weather Data	Google Weather API Preview – global 10-day forecasts; unified vendor ecosystem.
Authentication	Firebase Auth – secure, low-friction email/password & provider sign-in.
Data Storage	Firestore – real-time, per-user document store for notes and schedules.
Preferences	DataStore or SharedPreferences – simple key/value storage for user settings.
Build & Dependency	Gradle Kotlin DSL – typed, auto-complete build scripts; Compose BOM for unified versions.

Challenges Faced/Solutions

- Application crashed when user instantly switched from Homepage to the ‘Places Nearby’
    - Set a loading state to give time for the application to render the images

- Images in the ‘Places Nearby’ page failed to load. Issue with API not initializing properly.
    - Yielded the program in order to enable API to actually load

- Issue with NavHost, signup was crashing immediately after authentication.
    - Followed the flow of the user journey and made sure NavHost actually transferred from the signup page to the homepage.

- Problems combining backend and frontend aspects of our code smoothly.
    - We ended up merging the frontend to the backend, meaning we used the responses returned from the API to reconstruct our UI.

Potential Improvements

1. **Itinerary feature (New Itinerary)**
    a. **What:** Allows users to create an itinerary which will act as a folder of events
    b. **Why:** Improved planning feature, allowing users to plan an itinerary for a trip. Also brings better organization
2. **Sharing user schedule**
    a. **What:** Users are able to view other people's schedules and compare it against their own
    b. **Why:** Enables users to communicate with each other their availabilities
3. **Further customization of schedule**
    a. **What:** Custom color (via color palette) and display special information (e.g. buying a flight ticket or making a reservation)
    b. **Why:** Users can color code their schedules and access more information easily