Design Document: Smart Activity Planner

1. UI prototype

Prototype demonstrates how our application will work

https://www.canva.com/design/DAGQtEhXzFY/pj-tcfqURWFwoSvLvAqjrg/edit?utm_content=DAGQtEhXzFY&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

2. Structure

The application retrieves weather data using the provided location and time slot and fetches nearby places based on the location. It then generates activity suggestions using the weather and place information. Finally, the app displays the recommended activities and a graph visualizing weather changes during the selected time.

1. User input: Activity filters, Weather filters, Location, Timeslot
2. Fetch Weather Data: Using location and time slot
3. Fetch Nearby Places: Using location
4. Logic to Create Suggestions: Based on weather and places information
5. Show suggestions and Graph: Display recommendations and visualize data

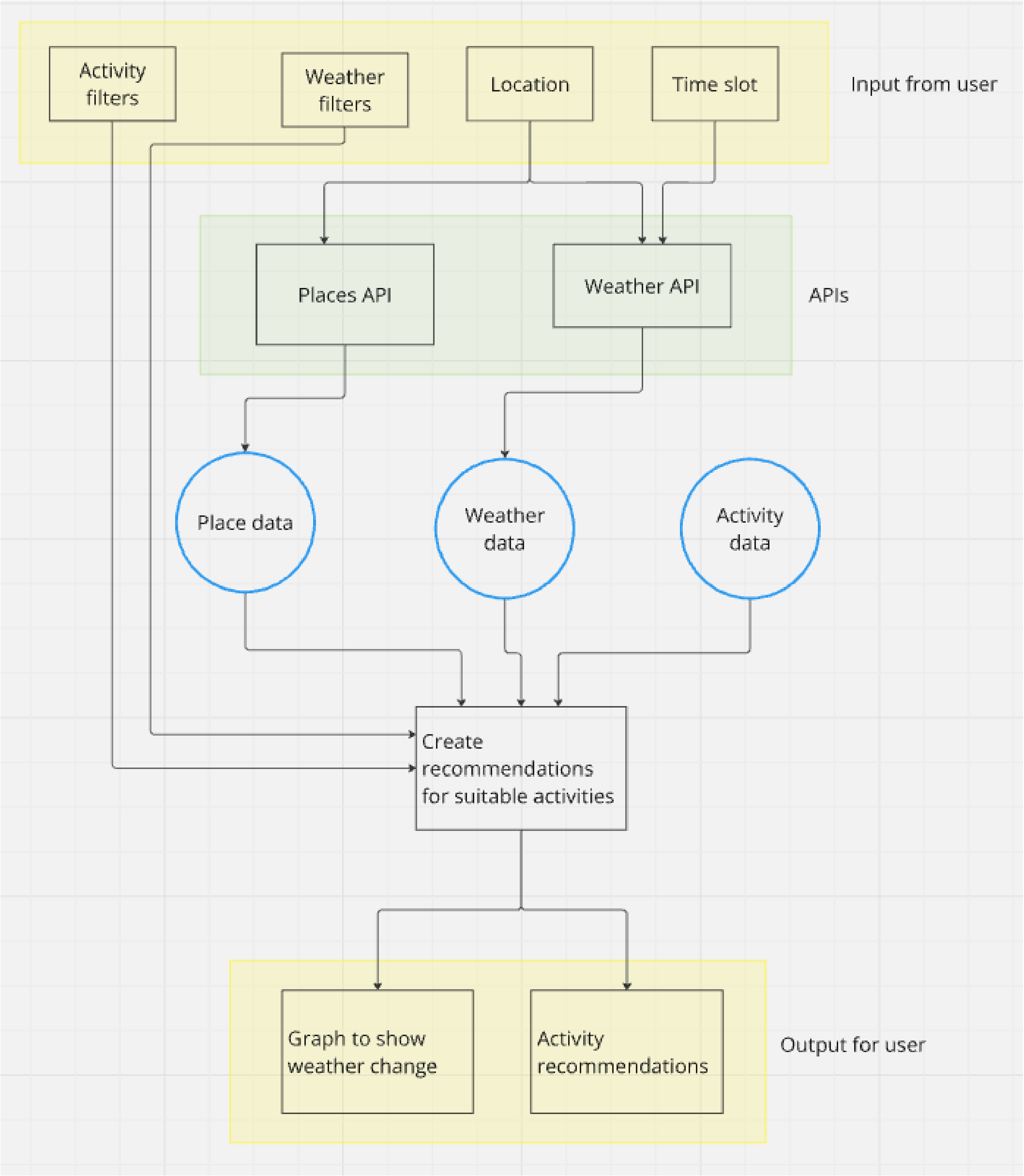
APIs

https://open-meteo.com/en/docs/geocoding-api to fetch weather data.

https://location.foursquare.com/products/places-api/ to fetch data of places.

Weather data will be fetched using the following inputs: location, start time, and end time. We will use the following data from the weather: precipitation, temperature, and wind speed.

Nearby places data will be fetched using the following inputs: location and categories (which can be empty if the user hasn't filtered them). We will use the following data from the places: the name of the place and its category.

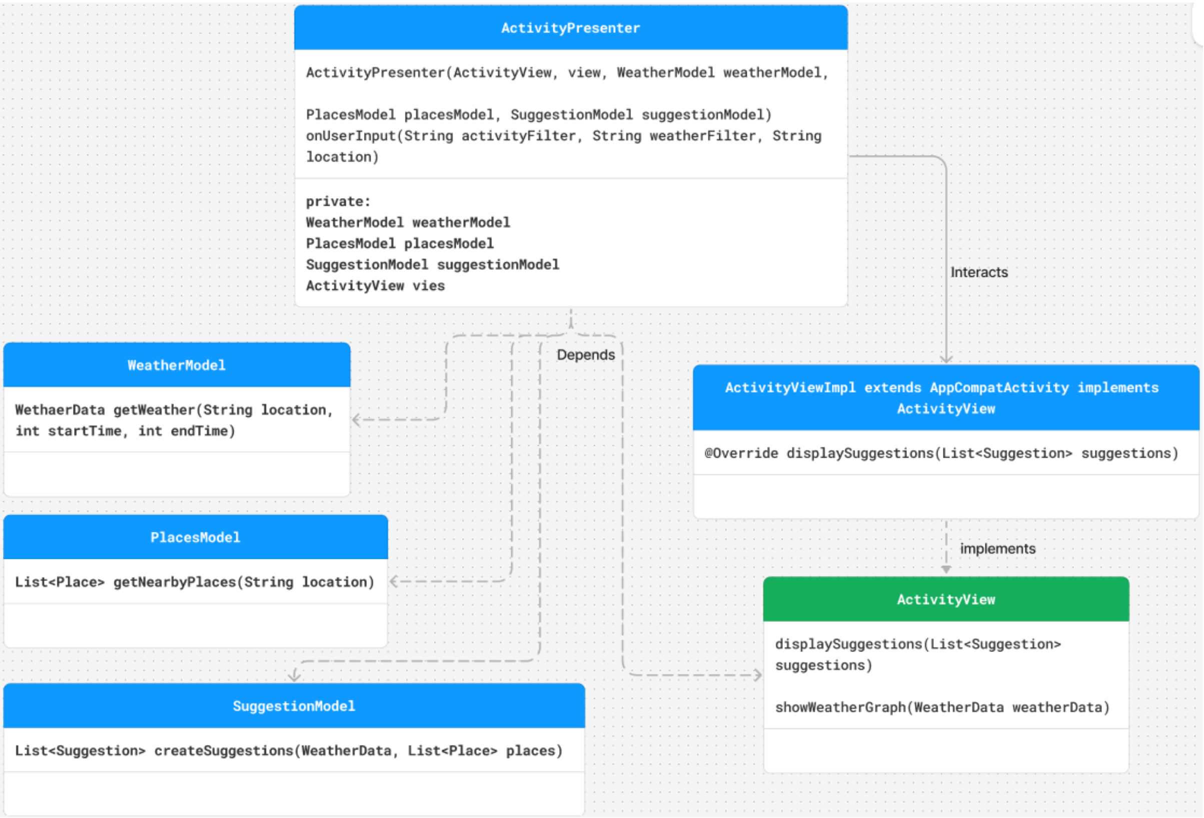


High level diagram of the application

3. Design patterns and classes

Model – View –Controller for architectural design. Clean separation of concerns between the data (API calls), the user interface, and the interaction logic.

Link for Figma: https://www.figma.com/board/s6ib0QdSPnEAYeVZFhMncp/SWDesignUML?t=i4RG5YBcPT1qgChu-1


ActivityPresenter
ActivityPresenter(ActivityView, view, WeatherModel weatherModel,
PlacesModel placesModel, SuggestionModel suggestionModel)
onUserInput(String activityFilter, String weatherFilter, String location)
private:
WeatherModel weatherModel
PlacesModel placesModel
SuggestionModel suggestionModel
ActivityView vies
Interacts
Depends
WeatherModel
WethaerData getWeather(String location, int startTime, int endTime)
ActivityViewImpl extends AppCompatActivity implements ActivityView
@Override displaySuggestions(List<Suggestion> suggestions)
implements
PlacesModel
List<Place> getNearbyPlaces(String location)
ActivityView
displaySuggestions(List<Suggestion> suggestions)
showWeatherGraph(WeatherData weatherData)
SuggestionModel
List<Suggestion> createSuggestions(WeatherData, List<Place> places)

UML Diagram of the application

4. Future planning and open issues

- Additionally, the following design patterns could be used in the application:
  1. **Builder pattern** for searching.
  2. **Factory pattern** for handling the different API services.
  3. **Singleton pattern** for instantiating new classes or repository classes.
- There are two potential approaches for creating logic for activity suggestions:
  1. **Hardcoded logic**: This can be demonstrated with a few activities to show how the application could work.
  2. **Creating a "database"**: For example, using JSON to define activities and leveraging that data to create suggestions.

    Activity data could look something like this:

    - **Activity**: Ice skating
    - **Weather**:
        - Temperature: -5°C to -15°C
        - Wind speed: 0 to 10 km/h
        - Rain: 0 to 1 mm
    - **Place**: Skating rink

- How the places data from FourSquare will be parsed. Is in JSON format. Chat GPT recommends Jackson Library or Gson.