📸 Pikkit

This is a prototype of **Pikkit**, a gamified, social photo-sharing app designed to be fun, addictive, and authentic.

The concept is simple: users receive a new photo challenge every day and earn points when they participate and receive likes from the community.

Core Mechanics & Rules

- Challenges are easy and accessible, so everyone can join the game.
- The goal is to capture the funniest or most beautiful photo—the community decides what's good.
- Completing a challenge rewards users with points, which vary based on the difficulty.
- Users earn additional points when their revealed photo is appreciated by others (likes, reactions).
- To ensure authenticity, all submitted photos are revealed simultaneously once the challenge ends.
- Points are converted into experience to allow users to **level up**.
- Levels unlock **badges** and can later grant access to **filters**, **stickers**, **emojis**, and even the ability to **add music or effects** to their Pikkits.
- Users can join or create **communities** (friends, family, colleagues, neighbors, fanbases, etc.).
    - Communities can be **public or private**.
    - Admins can send **custom private challenges** to members (no rewards for these).
- A **Ranking** feature allows users to compare levels, streaks, and accomplishments using filters like location or community.
- **In-app purchases and subscriptions** will unlock premium content such as exclusive filters, emojis, and sound packs.
- As DAU grows, Pikkit can offer **brand partnership campaigns**, where users participate in sponsored challenges and compete for real prizes.
- Additional gamification and reward ideas will evolve to enhance engagement and experience.

Technical Choices

For this prototype, I opted for a **modular architecture** using Gradle modules to clearly separate features and technical concerns.
Dependency injection is handled with **Koin**, which allows feature modules to define interfaces and implementations independently, keeping the architecture clean and scalable.

Modules Overview

`app`

The `app` module includes Android-specific components (Application, Activities, Services, etc.) and wires up all features using Koin modules. It stays lightweight—complex logic is delegated to feature modules.

`core`

The `core` modules house shared technical utilities and APIs that are reused across the app, promoting consistency and keeping the code DRY.

`features`

The `features` modules represent the actual product features from the user's perspective.

Each feature is broken into submodules to follow Clean Architecture:

- **Domain**: Pure business logic—defines entities and UseCase interfaces.
- **Data**: Implements repositories and UseCases using data sources (local or remote).
- **UI**: Exposes user interface components using Compose.
- **DI**: Ensures feature wiring is complete and consistent across layers.

Example – Feature Module Breakdown

Domain

Exposes feature-specific **intents** and **entities**, with clean UseCase interfaces. This layer contains no implementation or framework-specific code.

UI

Defines the **user experience**, exposing composable functions for each feature screen. These components are combined and orchestrated in the `app` layer.

Data

Implements the core feature logic using repositories, data sources, and UseCases. This is where the actual behavior lives.

DI

Defines the wiring for all components within the feature: ViewModels, UseCases, Repositories, etc. This ensures everything works end-to-end at runtime.

Conclusion

Designing and building an innovative photo-sharing app in under 48 hours is a real challenge.
I believe **Pikkit** offers a strong concept with a lightweight yet compelling core loop that has real potential to reach **1M daily active users**—thanks to its simplicity, gamification, and social mechanics.

While I couldn't implement everything I envisioned within the time constraints, I focused on creating a solid architecture and a scalable design that can evolve.

I hope you enjoy the result.