ChatGPT

Reverse Engineering mixi2's Flutter UI/UX Design

Mixi2 is a modern short-text social networking app (released Dec 2024) built entirely with Flutter [1]. It's often cited as proof that Flutter apps can be **polished and visually appealing**, countering the notion that Flutter UIs lack refinement. Below, we break down mixi2's design in three key areas – **UI Component Structure**, **Layout & Animation Design**, and **Design System** – and suggest how to achieve similar results in Flutter. (Notably, the mixi2 team built many custom widgets and architectures to realize their unique design [2], so we'll also highlight Flutter packages and techniques that can help emulate these effects.)

1. UI Component Structure

Mixi2's interface is organized into conventional social-app sections, each composed of **Flutter widgets tailored for a clean, friendly look**:

- **Bottom Navigation Bar:** The app uses a bottom tab bar with five main sections (Home feed, Search, Communities, Notifications, Messages), each represented by a simple icon and label. Unlike a default `BottomNavigationBar`, mixi2's nav icons have custom behavior – when tapped, an icon animates (e.g. the bell icon rotates with a wobble and inverts color on the Notifications tab) [3]. Each tab icon has a distinct motion, which adds a playful personality to the navigation. The Home icon, for example, only gently pulses (more subdued than others) [4], likely to suit its role as a return point. Implementing this in Flutter might involve a custom `BottomAppBar` with `IconButton` widgets and animation controllers for each icon, or using an animated icon library (mixi2 includes Rive and Lottie for complex animations [5]). Flutter's routing (mixi2 uses `go_router` for navigation [6]) swaps out the page content while the bottom bar persists.

- **Home Feed (Timeline):** The Home timeline screen presents a vertically scrolling feed of posts (similar to Twitter/Facebook timelines). Each post can be viewed as a **card-like component** containing user avatar, username, timestamp, text content, and action buttons (like, comment, share counts). In mixi2, posts are likely implemented with a custom `ListItem` or `Card` (rounded corners) holding a `Row` of avatar & name, and post text below. If a post includes images, it might show a grid of thumbnails or a swipable carousel (Flutter's `PageView` with a `smooth_page_indicator` for multiple images [7]). Reaction counts and icons are displayed consistently (likely using `Row` of Icon+Text). To replicate, Flutter's `ListView.builder` or `CustomScrollView` with `SliverList` can be used for an infinite scroll feed, and `Card`/`Container` with `BorderRadius` for the soft card UI.

- **Reactions and Emojis:** Mixi2 distinguishes itself with a rich **reaction sticker** system. Users can react to posts with emoji or short text stickers (e.g. "いいね", "最高", "おつかれ" in pastel bubble text). These are rendered as small pill-shaped graphics rather than standard Unicode emojis. **Figure 1** shows a variety of these reaction stickers, in both pictorial and Japanese text styles, all in a friendly pastel color palette:

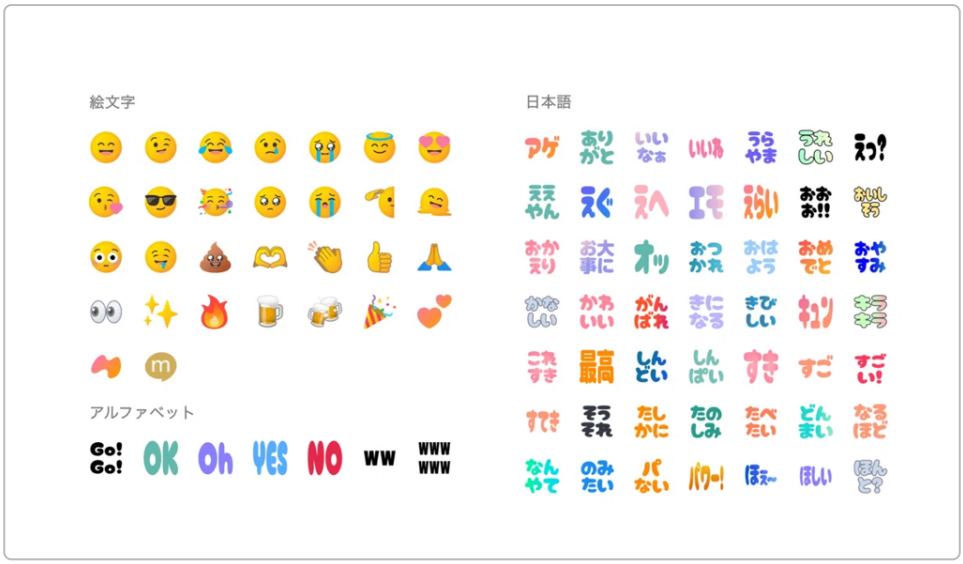

Figure 1: Examples of mixi2's reaction stickers (emoji and Japanese text) that users can attach to posts. The reaction set avoids harsh expressions – even "sad" or "tired" are phrased in gentle, empathetic terms, reinforcing a positive community vibe [8].

In Flutter, such reactions could be implemented as image assets or drawn on the fly. Mixi2 reportedly allows users to even create custom reaction images by choosing text and font, which suggests the app might dynamically render text into an image or canvas. For a developer, an easy approach is to use pre-made assets for common reactions and an `Image` widget to display them. Ensuring these reaction chips have consistent styling (rounded shape, pastel background) contributes to the polished feel. When the user taps to add a reaction, an overlay or menu (perhaps a `showModalBottomSheet` or popover) likely presents these options in a grid.

- **Communities & Events:** Mixi2 revived the classic "Communities" feature of the original mixi [1]. The Communities screen probably lists groups or topics the user can join, each represented by a row or card with the community name and icon. A grid layout (`GridView`) might be used for discovery of featured communities. Another major feature is real-time **Events** (for spontaneous gatherings or live chats). The Events interface could include a calendar icon and list of ongoing or upcoming events, possibly with bright indicators for live events. These might be shown as cards or list tiles with bold highlights when active. Flutter's flexible layout widgets (`Column`, `Expanded`, etc.) would help arrange community info and event details. Given mixi2's emphasis on real-time, the UI likely indicates live updates (e.g. a "live" badge) – a custom `Container` with a colored rounded rectangle and text could serve as a badge widget.

- **Notifications & Messages:** The Notifications screen shows interactions (likes, follows, replies), presumably in a list form (using `ListTile` widgets with an avatar, message, and time). To allow quick actions (e.g. swipe to delete or mark read), mixi2 uses the `flutter_slidable` package [9] for iOS-like swipeable list items. This adds polished behavior like revealing "Delete" or "Hide" buttons on a notification entry. The Messages section likely resembles a chat inbox; each conversation could be a clickable list item. Flutter's `Dismissible` could also achieve swipe actions, but `flutter_slidable` provides a more refined, multi-action swipe experience (as seen in mixi2). Both notifications and messages would use consistent spacing, typography, and icons for a cohesive look.

- **Profile & Settings:** Users have profile pages (with their avatar, bio, stats, recent posts). Often this pattern uses a **collapsing header** – mixi2 might employ a `NestedScrollView` (they include

`nested_scroll_view_plus` for extended scrolling features [10]) so that the profile header (with cover image or avatar) scrolls away to a toolbar. The Settings pages are accessible via a profile menu; one notable settings feature is **Text Size adjustment**. Mixi2 includes a screen where users can drag a slider to change the global font size, with a live preview sample of a post [11]. Figure 2 shows this Text Size setting screen in mixi2:

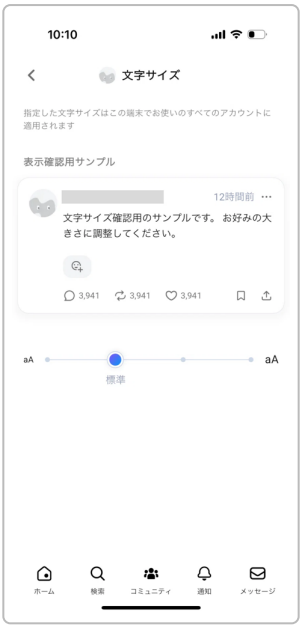

Figure 2: Mixi2's "Text Size" settings screen with a preview post. A slider (aA – Aa) lets the user adjust font scaling; the sample post updates immediately for feedback [11]. This screen also reveals mixi2's bottom navigation bar (Home, Search, Communities, Notifications, Message) with simple outline icons.

Implementing such a preview in Flutter can be done by reading the slider value and applying it to a `TextStyle` on the sample content (using `MediaQuery.textScaleFactor` or a custom state that rebuilds Text with a larger font size). Mixi2 likely stores the user's preference and applies it app-wide (e.g., using a state management solution to update the app's `MediaQuery` or theme dynamically).

Technical structure: Under the hood, mixi2's Flutter app uses a robust architecture to support these UI components. They manage state with **Riverpod** (along with `flutter_hooks` and `Freezed` for immutable models) [6], which ensures a clean separation of logic and UI and makes rebuilding specific widgets efficient. Navigation is handled with **go_router** [6] for declarative routing between pages (tabs, profile, settings, etc.), enabling smooth transitions and deep linking. This kind of architecture means each UI section (timeline, notifications, etc.) can be developed somewhat independently with consistent state management, which contributes to the polished feel (no glitchy state or unexpected UI pops as you navigate).

2. Layout and Animation Design

Mixi2's layouts leverage Flutter's responsive and flexible design capabilities to create an interface that feels both **dynamic and coherent** across different devices. Key layout strategies include:

- **Flexible, Responsive Layouts:** The app uses standard Flutter layout widgets like `Row`, `Column`, and `Expanded` to adapt to different content sizes. For example, in a post card, the text content might wrap naturally and push other elements, thanks to Flutter's flexible `Column` layout. The

design likely follows a **4- or 8-point grid** for spacing – e.g. consistent padding around cards and between UI elements – which Flutter makes easy via `EdgeInsets.symmetric(horizontal:16)` etc. For multi-column layouts (perhaps in community discovery or a grid of images in a post), Flutter's `GridView` or `Wrap` widget would be used. Mixi2's UI on phones is primarily single-column, but being built with Flutter means it can scale to tablets or web with minimal changes. One can use `LayoutBuilder` and `MediaQuery` to adjust widget sizes or switch to a two-column layout if needed (though mixi2 at launch was mobile-only [12]).

- **Hierarchical Scrolling:** In places like profile screens or perhaps a combined feed with multiple sections, mixi2 takes advantage of nested scrolling. The inclusion of `nested_scroll_view_plus` [9] suggests they implement an **interactive app bar** that collapses or floats as you scroll, improving the spatial organization of content (common in polished apps). This means the UI has a sense of depth – e.g., a community page might have a banner image that shrinks into a smaller toolbar as you scroll through posts. Flutter's `NestedScrollView` with a `SliverAppBar` can achieve this effect, keeping the experience smooth.

- **Animations & Transitions:** Arguably the standout aspect of mixi2's UI is its **delightful animations and micro-interactions**. Flutter's animation framework is fully utilized to give immediate feedback and personality to the app. For instance, tapping various buttons triggers small animations: the bottom nav icons animate on tap (shaking or rotating as noted earlier) [3], and there are likely subtle transitions when switching tabs or opening modals. Page transitions in Flutter by default are platform-specific (Cupertino-style on iOS, Material fade-through on Android), which mixi2 likely keeps to feel native. In addition, custom page transitions could be defined via `PageRouteBuilder` in `go_router` if they wanted a unique animation between certain screens. Overall, navigation feels smooth and not abrupt.

Another area of animation is in content itself. Mixi2 introduced "エモテキ" ("Emotional Text") – a feature where post text can animate (e.g. wobble, zoom, or show animated background effects) [13]. This is an advanced UI touch: for example, a post can have its text bouncing up and down (ホッピー effect) or shaking side to side (シェイク effect), or the background of the text might sparkle or show hearts for extra emphasis [13]. Implementing this in Flutter might involve animating text using widgets like `AnimatedDefaultTextStyle` or even converting text to a renderable widget that can be moved. Simpler effects (like scaling text for "BIG" text) can be done by just increasing font size or using a `Transform.scale`. More complex jitter or particle effects (like sparkles behind text) could use **Rive or Lottie animations**: indeed mixi2's package list confirms it uses `rive` and `lottie` [5], likely to render these custom animations efficiently. A developer aiming to replicate such effects could design vector animations in Rive (for icon/button animations) and trigger them in Flutter using the Rive runtime. Lottie can play designer-made JSON animations (e.g. a confetti burst when a user successfully creates an event).

- **Micro-Interactions with Flutter:** Many of mixi2's polished UI moments are micro-interactions that can be achieved with Flutter's animation APIs. For instance, to recreate the bottom navigation icon animation without external assets, you can use **implicit animations** or **explicit animations**. Flutter offers implicit widgets like `AnimatedRotation`, `AnimatedScale`, etc., which animate a property over a given duration when a state changes. For example, to spin an icon when tapped, one could do:

```
// Pseudocode for a rotating icon on tap:
IconButton(
  icon: AnimatedRotation(
```

```
    turns: _isTapped ? 1 : 0,        // 1 turn = 360°
    duration: Duration(milliseconds: 500),
    curve: Curves.easeOut,
    child: Icon(Icons.notifications),
  ),
  onPressed: () {
    setState(() {
      _isTapped = true;
    });
    // Optionally reset _isTapped to false after animation if we want one-shot behavior
  },
),
```

In the code above, tapping the icon triggers a smooth 360° rotation (with easing) [3]. We could also combine this with an `AnimatedSwitcher` to swap the icon's image or color at the same time (to emulate the color invert on the notification bell). For more control, Flutter's **explicit animation** classes (`AnimationController` with `AnimatedBuilder`) let you sequence animations (e.g., a slight jiggle followed by rotation). Mixi2's unique icon motions likely use tuned curves and sequential transforms – something Flutter handles with ease (e.g., using multiple `Animation<double>` for scale, rotation, opacity etc. combined in a `AnimatedBuilder`).

- **Smooth Scrolling and Platform Feel:** To ensure the app feels native, mixi2 accounts for small platform differences. On iOS, for instance, scroll physics have bounce overscroll; Flutter's `CupertinoScrollBehavior` or using `ScrollConfiguration` can enable that behavior globally. Mixi2 even uses a plugin to support the iOS **Scroll-to-top** gesture (tapping the status bar to scroll a list to start) [14], via the `scrolls_to_top` package. This is a subtle UX detail that many cross-platform apps overlook. By handling it, mixi2's scrolling experience matches user expectations on each platform, contributing to the polished impression.

- **Performance Considerations:** All these animations and custom layouts could risk jank, but mixi2's team designed carefully. Flutter's Skia rendering ensures even complex UI is performant, but they also likely used techniques like `VisibilityDetector` (which they include [15]) to only animate what's on screen, and perhaps cached images/avatars using `cached_network_image` (not explicitly listed, but they use `extended_image` which provides caching and advanced image handling [16]). The result is a UI that not only looks good but feels smooth during interaction.

3. Design System and Visual Consistency

Mixi2's visual design follows a well-defined **design system** that makes the app feel cohesive and high-quality. Key elements of this system include **color palette**, **typography**, **iconography**, and consistent **spacing & shapes**:

- **Color Palette:** Mixi2's colors are gentle and friendly. The primary brand color extracted from the UI appears to be a vibrant, slightly playful hue (possibly an orange or other warm tone), but it's used sparingly – mostly for accent elements – so the overall UI doesn't feel loud [17]. The app heavily uses neutral backgrounds (white or very light gray) with plenty of whitespace, which makes content clear and keeps the interface from feeling cluttered [17]. A secondary color is a **bluish gray** which serves for secondary buttons or backgrounds, complementing the primary color [17]. Notably, mixi2 chose pure black (`#000000`) for certain text and icon elements (like the tab bar icons) to ensure good contrast [18]. Normally designers avoid absolute black (preferring

softer dark gray), but here the decision was intentional: against the soft gray secondary UI, a semi-transparent black might appear too faint, so true black gives needed definition [18]. This shows an attention to context in color usage. When building a similar app, you would define a custom `ColorScheme` in `ThemeData` – for example, `primary = Color(0xFFFFA726)` (just an example warm color), `secondary = Color(0xFF90A4AE)` (a gray-blue), etc., and apply these to widgets consistently (e.g., ElevatedButtons use primary for their fill, text links use primary, whereas surfaces use neutral colors). Mixi2 likely also supports dark mode, but their primary aesthetic is a light theme that aligns with the "warm and gentle" concept.

- **Typography:** A striking aspect of mixi2 is its use of fonts. Instead of default Material fonts, they use **Google Fonts** – specifically Noto Serif Japanese for Japanese text and Roboto for Latin text [19]. This pairing is interesting: Noto Serif JP is a serif font, giving a humanistic, approachable feel for Japanese content, while Roboto is a clean sans-serif that pairs well for English or UI elements. The serif adds a touch of friendliness or nostalgia (perhaps nodding to mixi's origins), setting it apart from the typical all-sans tech look. All text elements (posts, menus, headings) are styled for clarity: likely the app uses a base TextStyle with a comfortable size (perhaps 14-16pt for body text) and applies the custom font via Flutter's `google_fonts` package [19]. For example, they might set `theme.textTheme` using `GoogleFonts.notoSerifJapaneseTextTheme()` so that all Text widgets automatically use it. Mixi2 also emphasizes accessibility: the **Text Size** setting (Figure 2) allows users to scale font size easily [11]. Implementing such a feature in Flutter means using `MediaQuery.textScaleFactor` or updating the app's `ThemeData.textTheme` based on user choice and ensuring widgets respect it. The preview in Figure 2 shows how even the sample post updates instantly – likely achieved by rebuilding that widget with the new font size on the fly. By providing this option, mixi2 ensures users can tailor readability to their needs, which is a mark of a user-first, polished design.

- **Icons and Imagery:** Mixi2 uses simple line icons for its UI (as seen in the bottom nav, which has outline icons). These icons are likely custom-designed or chosen to match the brand's style (they appear slightly more rounded than stock Material icons, aligning with mixi2's friendly vibe). The reaction images (Figure 1) and other graphics (like animated stickers or emojis) follow a **consistent style** – colorful but soft, with a uniform flat look and rounded forms. Consistency in icon style and illustration style throughout the app contributes greatly to perceived polish. A developer cloning this should pick an icon pack that is cohesive (or use a design resource to create icons). Flutter makes it easy to use custom SVG or PNG icons (via `Image.asset` or `SvgPicture` from Flutter SVG). Mixi2 also incorporates tiny illustrations in subtle places – for example, maybe an empty state graphic when a list is empty, or celebratory icons when an action is done. Using such imagery (possibly via Lottie for subtle animation) can add delight. Because mixi2's icons animate (the bottom icons literally spin or bounce), those might be implemented as Rive animations or by using two states of an SVG and animating between them. The use of **Rive** suggests some icons or mascot characters might be animated using that tool, which allows for intricate, high-performance animations (much smoother than traditional GIFs or heavier than doing it in pure Dart code) [5].

- **Spacing and Layout Consistency:** Throughout mixi2, **generous spacing** is a hallmark. Screens have ample padding at the edges and between elements, avoiding a cramped feeling [17]. Likely, they use at least an 8px base grid – e.g., a card might have 16px padding inside, list items might be separated by 8px gaps, etc. All cards and containers have **rounded corners**, reinforcing the approachable aesthetic [20]. We can infer a standard border radius (perhaps 12 or 16 logical pixels on major surfaces like cards and modals). The design avoids any sharp corners – even buttons and text input fields probably have rounded edges. In Flutter, one would set `RoundedRectangleBorder(borderRadius: BorderRadius.circular(12))` on Card and dialog themes

to apply this app-wide. The **consistent use of radius and padding** in mixi2 means every screen feels part of the same family. For example, the modal dialogs (like confirmation dialogs or the bottom-sheet style composer for a new post) presumably share the same corner rounding and shadow style as other components. By defining these in the ThemeData (e.g., `theme.shape` for buttons, or using Material 3's shape scheme), the developers ensure uniformity without repetitive styling.

- **User Experience Cohesion:** Beyond visual elements, mixi2's design system reflects in the UX choices. The app's tone is "soft and empathetic" – this influenced UI details like the reactions (no toxic reactions, only supportive ones) [8] and the color/shape choices that "soften" the interface. The result is that the UI itself reinforces the community guidelines (be kind, have fun). For a developer, this is a reminder that design system is not only about colors and fonts, but also about the feel of interactions. Mixi2's team likely iterated with designers using Figma to create custom components that embody this feel, and the Flutter engineers implemented many widgets from scratch to match the design exactly [21] [22]. For instance, instead of using a default Material `TextField` for input (which might not have matched their desired style or had some platform quirks), they used a `flutter_native_text_input` plugin to get truly native text field behavior on each platform [23]. This ensured things like text selection handles, cursor behavior, and autocorrect feel just right on iOS and Android – a level of polish users notice during typing.

- **Suggested Packages and Techniques:** To emulate mixi2's polished UI, developers can leverage several Flutter packages (many of which mixi2 itself uses) and best practices:

- **State Management & Architecture:** Adopt a scalable state management like **Riverpod** with **Freezed** models for predictable state and UI updates [6]. This will help keep your UI in sync and avoid inconsistent states that break polish. Use **go_router** or `Router` API for smooth navigation and deep link handling [6].
- **Animation Libraries:** Integrate **Lottie** and **Rive** for complex or custom animations [5]. For example, use Lottie for an onboarding illustration or confetti effect, and Rive for interactive icons or a mascot animation. These tools let designers contribute animations that you can directly use in Flutter.
- **Advanced UI Components:** Use packages like `flutter_slidable` for polished list actions [9], `smooth_page_indicator` for nice-looking page dots on carousels [7], and `super_tooltip` for any helpful tooltips or coach marks [5] to onboard users. These save time and match the level of polish users expect.
- **Custom Styling:** Take control of theming: use **google_fonts** to easily apply custom fonts [19], and adjust the ThemeData to customize widget styles (e.g., set `appBarTheme` for a consistent header look, `bottomNavigationBarTheme` for the nav bar style, etc.). Mixi2's use of Material 3's color utilities [24] hints they might derive tonal palettes dynamically (perhaps for user-generated content or theming). The `material_color_utilities` package can generate tonal palettes from a single color [24], which you could use if allowing user theme customization.
- **Platform Adaptation:** Don't forget the little iOS/Android differences. Mixi2's use of `scrolls_to_top` (for iOS) [14] and native text fields [23] shows their attention to platform-native feel. Using `cupertino_widgets` for certain iOS-styled controls or at least adjusting behaviors via platform checks can elevate the UX. Flutter makes it easy to detect `Theme.of(context).platform` or use packages like `flutter_platform_widgets` if needed.
- **Testing and Iteration:** Lastly, a polished UI comes from iterative refinement. Mixi2's team likely wrote extensive UI tests and did design reviews to catch inconsistencies. Flutter's hot-reload would have helped tweak spacing and animations in real-time. Incorporating user feedback (like mixi2 did with accessibility improvements after launch [25]) is also crucial.

By combining these approaches – thoughtful component structure, responsive layouts with delightful animations, and a strong visual design system – you can create a Flutter app that achieves a **mixi2-level** of UI/UX polish. Mixi2 stands as an inspiring example that Flutter apps can be both beautiful and performant, provided you sweat the details and use the right tools [22]. Following its blueprint of custom-tailored widgets, cohesive design decisions, and user-centric features will set you on the path to building a similarly refined Flutter application.

Sources: The analysis above is based on public insights from mixi2's developers and designers, including tech talks and articles (e.g., Developers Summit 2025 sessions, mixi2 team interviews) and community write-ups that examined mixi2's app structure and design [1] [3] [17] [5]. These sources have been cited inline to provide reference for specific details.

---

[1] 1週間で120万人突破の「mixi2」小規模チームで構築・運用を可能にしたNewSQL「TiDB」とは | PingCAP株式会社
https://pingcap.co.jp/case-study/mixi2/

[2] [21] [22] インターンでmixi2を開発してきたよ｜くりきん
https://note.com/_kurikin/n/n6785b8f20e21

[3] [4] [8] [11] [17] [18] [20] mixi2のUIデザインが素敵｜Akio Sakamoto
https://note.com/akio_sakamoto/n/nb68fed54d7c7

[5] [6] [7] [9] [10] [14] [15] [16] [19] [23] [24] mixi2ライセンスページから学ぶ、便利なFlutterパッケージ〖120over〗 - くらげになりたい。
https://memory-lovers.blog/entry/2024/12/19/085505

[12] mixi2は何の技術で作られてる？（Flutter製？）&初め方 &何これ？
https://qiita.com/nekojs/items/20709623b5e307a54b26

[13] 〖mixi2〗文字を震わせる・大きくするエモテキ機能の使い方を解説 | SBAPP
https://sbapp.net/appnews/sns/mixi2/emotext-howto-166650

[25] 「本気のユーザーサプライズファースト」を追い求めて。「mixi2」の開発で描く成長軌跡 | ミクシル
https://mixil.mixi.co.jp/8104/