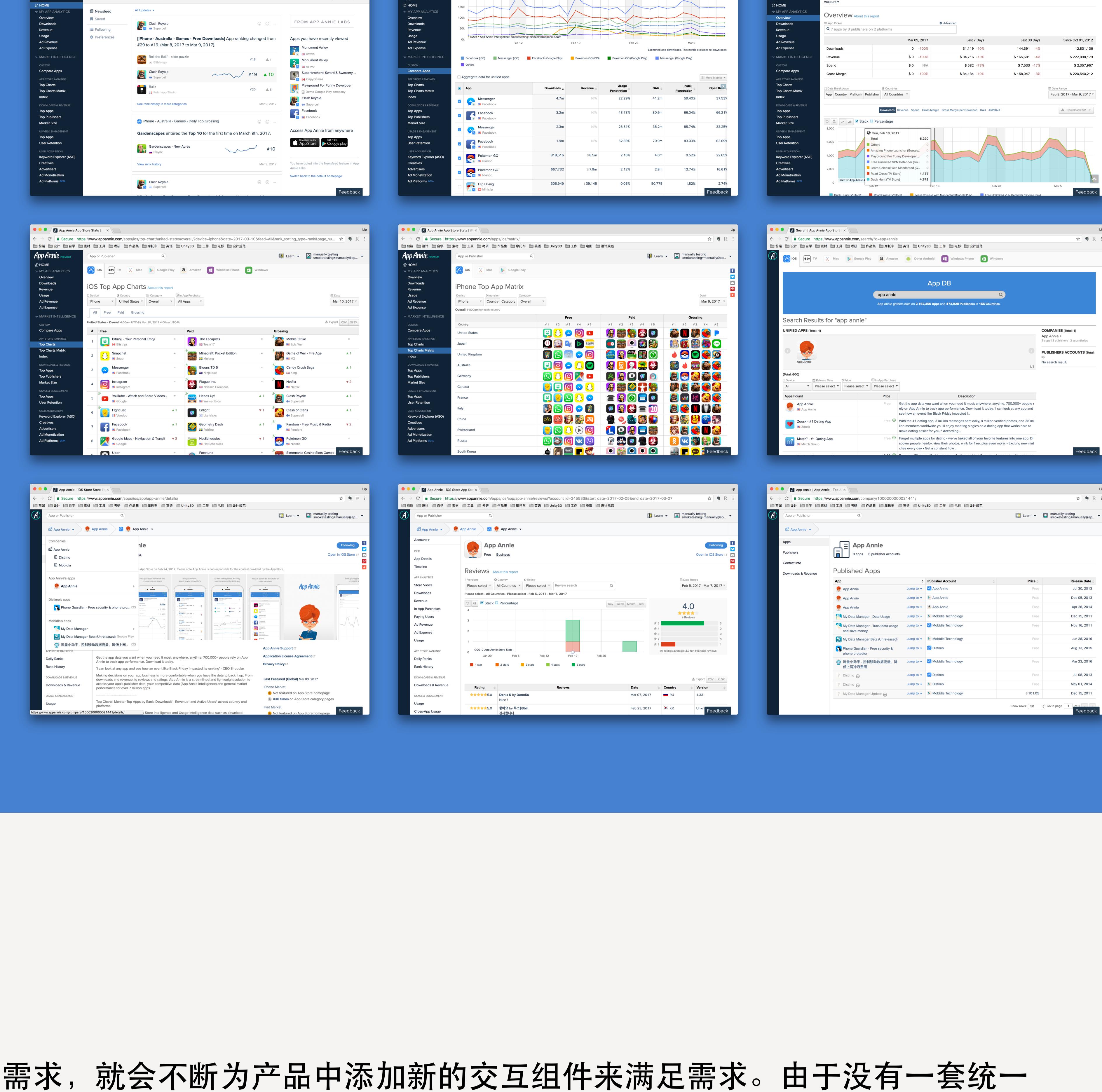


App Annie 组件库

公司现有的复杂产品线存在许多组件交互行为与样式不一致的情况，我的工作是协助设计合理的组件交互行为，为每个组件制定设计规范，并制作规范可复用的设计素材供其他设计师使用，同时与开发人员密切协作更新旧有的网站组件。



背景

公司在快速扩张时，会不断面临新的产品需求，就会不断为产品中添加新的交互组件来满足需求。由于没有一套统一的组件库与组件规范，会出现：

- 已知的组件没有被合理的复用，反而为产品去开发新组件，这样不仅拖慢了新功能的开发进度，也使代码库更加冗余。

- 即使在不同产品线中使用了类似的组件，但由于没有统一的视觉与交互标准，造成用户体验不一致性。

因此建立一套组件库的意义是为了解决上述问题，通过把现有的产品样式、组件、模式收集起来，为常见的设计问题提供可循环复用的解决方案。当有了这样一套组件库，在之后的产品设计与开发工作中要遵循组件库中的通用规则，从而保证用户体验的一致性、代码的高可复用性以及设计团队日常工作的可参照性。

简单来说就是构建一套让产品、设计与开发都可以参照的设计标准。与此同时，当现有的组件库不能满足业务需求时也需要不断完善。

组件库的建立

在构建复杂产品的组件库时，需要有一个整体的框架，我们可以从内容与行为这两个角度出发：

Contents

Style: 定义了产品中最基本的视觉样式，包含字体、颜色，以及一些产品中所出现的基本的元素的视觉形式，如图片、图表、图标等。

Data Pattern: 定义了产品中数据信息的展示信息、格式。

Layout Pattern: 定义了组件的布局形式。

Behaviour

UI Controls: 定义了产品中所出现的较为通用的交互组件。

Widgets: 定义了产品中与业务逻辑结合较为密切的组件，偏向于产品中特有的组件。

System Behavior: 定义了产品中应该出现的标准的交互行为，往往与一个固定的业务逻辑密切结合。



组件设计规范的建立

当有了一套组件库之后，便需要站在组件复用的角度为不同的组件制定详细的设计规范。这时需要一个更为详细的框架：

Contents

Elements: 定义了组件中应该所出现的元素、元素的含义。

Style: 组件基本的视觉样式，如字体、颜色、组件元素的样式。

Data Pattern: 组件中数据信息的展示信息、格式。

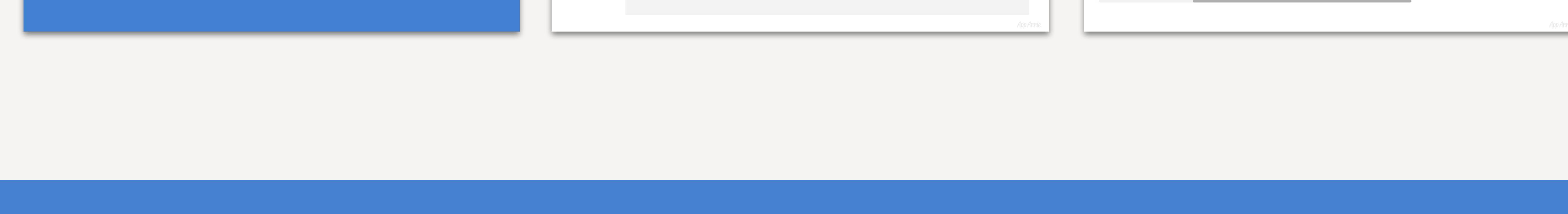
Components Layout Pattern: 组件的布局形式。

Behavior

Global Components Behavior: 定义组件所具备的整体的交互行为。

Elements Behavior: 定义组件中元素中所具备的交互行为。

Related Components Behavior: 定义与相关组件所产生的交互行为。



App Annie Components Library