Assignment VI: Memorize Themes

Цель

Цель этого задания - узнать, как иметь несколько MVVM в вашем приложении и как представлять группы **Views** с помощью навигации или модально.

Обязательно посмотрите ниже раздел подсказок **Подсказки (Hints)**!

Посмотрите также последние изменения в разделе **Оценка (Evaluation)**, чтобы вы понимали, что будет оцениваться в Домашнем Задании.

Срок сдачи

Это последнее Задание в семестре перед тем, как вы начнете свой финальный проект. По объему оно похоже на ваше Задание 3, поэтому не забудьте соответствующим образом распланировать свое время..

Материалы

Вы начнете это Задание с Memorize из вашего Задания №5.

Обязательные пункты задания

1. Теперь ваше приложение Memorize должно при запуске отображать UI «выбора темы». Примеры см. в прикрепленных изображениях, но вы можете проявить творческий подход к тому, как показывать каждую тему.

2. Используйте **List** для показа тем (*themes*).

3. Каждая строка в **List** показывает имя темы (*theme*), цвет темы, сколько карт в теме и некоторую выборку эмоджи.

4. Касание темы (*theme*) в **List** переводит (то есть **List** находится внутри **NavigationView**) вас на игру с этой темой.

5. Во время игры название темы должно где-то отображаться на экране, и вы также должны продолжать поддерживать существующие функции, такие как счет, новая игра и т. д. (Но вы можете изменить UI, если хотите).

6. Это нормально, если при переходе от игры обратно к выбору темы в **List**, а затем обратно к текущей игре в полном разгаре все-таки игра перезапускается, хотя разумные реализации, вероятно, этого не сделают (кроме случаев, когда тема (*theme*) изменена (см. Ниже), поскольку в этом случае вы все равно захотите перезапустить игру.

7. Предоставьте некоторый UI для добавления новой темы (*theme*) в список тем **List**.

8. **View** для выбора темы (*theme*) должен поддерживать режим редактирования Edit Mode, при котором вы можете удалять темы (*themes*) и при котором у вас есть доступ к некоторому UI (то есть кнопка **Button** или изображение **Image** в каждой строке), который выводит вас модально (через **sheet** или **popover**) на UI Редактора Темы и дает возможность редактировать эту тему (*theme*).

9. Редактор Темы должен использовать форму **Form**.

10. В Редакторе Темы пользователю разрешается редактировать имя темы (*theme*), добавлять эмоджи к теме, удалять эмоджи из темы и устанавливать количество карт

в этой теме. (Есть Дополнительный Пункт, который связан с возможностью редактирования цвета темы).

11. Темы (*themes*) должны постоянно сохраняться (перезапуск вашего приложения не должно вызывать потерю всех отредактированных тем).

12. Ваш UI должен хорошо выглядеть как на iPhone, так и на iPad.

13. Заставьте ваше приложение работать на любом физическом устройстве iOS.

Подсказки (Hints)

1. Вы, вероятно, захотите начать с создания ViewModel для вашего **View** выбора темы (*theme*). Это просто хранилище для ваших тем (*themes*) (даже проще, чем для **EmojiArt**, поскольку вы не поддерживаете переименование на месте, поэтому вам даже не нужен словарь имен). Вам понадобится простой массив **Array** тем (*themes*), который вы будете сохранять в **UserDefaults** как **JSON** (вы сделали это возможным в вашем Домашнем Задании № 5).

2. Поскольку эта новая ViewModel является всего лишь хранилищем (т. е. не выполняет никакой логики), то нет никаких причин не делать её массив тем (*themes*) не-**private** (т. е. вам не нужно описывать все виды изменений, которые вы собираетесь делать с темой (*theme*) с помощью функции в этой новой ViewModel, просто позвольте **View** напрямую управлять массивом **Array** тем (*themes*)).

3. Вы определенно захотите «автосохранять» любые изменения в вашем массиве **Array** тем (*themes*) (что мы делали в EmojiArt два раза, поэтому вы должны быть знакомы с тем, как использовать для этого “издателя” **Publisher**).

4. В **View** выбора документов **EmojiArt** у нас был только текст **Text** (а затем редактируемый текст **EditableText**), представляющий строки в списке **List**. Но нет абсолютно никаких причин, по которым вы не можете вместо этого создать собственное **View**. На самом деле вы обязательно захотите это сделать..

5. Вы можете значительно упростить «добавление темы», просто создав простую тему (*theme*) по умолчанию (похожую на «Untitled» документ) с несколькими эмоджи на ней, а затем пользователь может просто отредактировать ее по своему вкусу, привлекая позже Редактор Палитры. Другими словами, кнопка добавления темы не обязана вызывать собственную модальную панель.

6. Не делайте код в вашем **View** для Редакторе Темы гигантским. Разбивайте его на меньшие **Views** (может быть одно **View** для каждой секции **Section**, например). Точно так же, аккуратно организуйте код для вашего кастомного **View**, которое показывает каждую строку в списке **List**.

7. Есть две главные стратегии для модального редактора похожего на Редактор Темы. Первая - «редактирование в реальном времени», а вторая - «**done** / **cancel**»...

8. При «живом редактировании» изменения, которые пользователь вносит в тему (*theme*), немедленно помещаются в хранилище. Если вы используете этот подход (а это обычно предпочтительнее в iOS), вы можете передать свою ViewModel (ваше хранилище) вместе с темой (*theme*), которую хотите отредактировать, в свой Редактор Темы (подумайте об изменениях, которые вносит пользователь, как о «намерениях пользователя», и создайте Intent функции в вашем хранилище для этих изменений). Или вы можете использовать **Binding** напрямую к массиву **Array** тем (*themes*) в ViewModel. Последний подход может быть немного сложным, поскольку вы также поддерживаете удаление тем (*themes*) из этого массива **Array**, и поэтому у вас могут возникнуть обстоятельства, когда вы удаляете тему (*theme*) прямо из **Binding** какого-либо другого **View** (в зависимости от того, передаете ли вы свою тему (*theme*) другим **Views** с помощью **Binding**).

9. В модальном **View**, выполненном в стиле «**done** / **cancel**» должна быть кнопка **Done** где-то в Редакторе Темы, которая убирает (*dismisses*) с экрана Редактор Темы и копирует все изменения, сделанные в теме (*theme*) во время нахождения Редактора Темы на экране, обратно в хранилище. У вас также будет кнопка **Cancel**, которая уберет (*dismisses*) с экрана Редактор Темы *без изменения* хранилища.

10. Если вы все-таки выполняете «живое редактирование», передайте свою ViewModel (хранилище тем) Редактору Темы с помощью **.environmentObject ()** (т.е. не передавайте его в качестве аргумента и не используйте **@ObservedObject**). Это (в настоящее время) требуется при передаче ViewModels модально представленным **Views**, и, кроме того, **Views**, составляющим ваш Редактор Темы, все равно будут нуждаться в этом, поэтому они все могут просто использовать **@EnvironmentObject** для получения ViewModel.

11. Не позволяйте тому UI в Редакторе Темы, который выбирает количество пар карточек, выбирать число, превышающее число эмоджи, доступных в теме! И не позволяйте ему выбирать одну пару (это было бы слишком просто для игры).

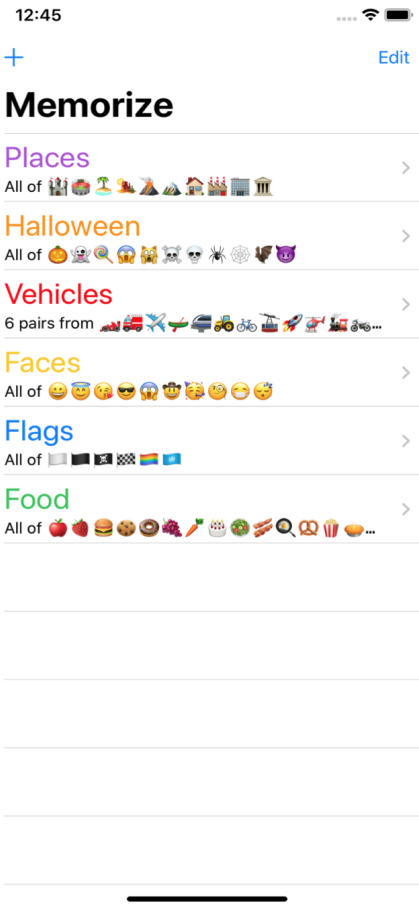
12. Вам нужно будет решить, что делать, если в какой-либо момент во время редактирования темы присутствует (или есть угроза их появления) менее двух эмодзи. Есть несколько разумных подходов к этой ситуации.

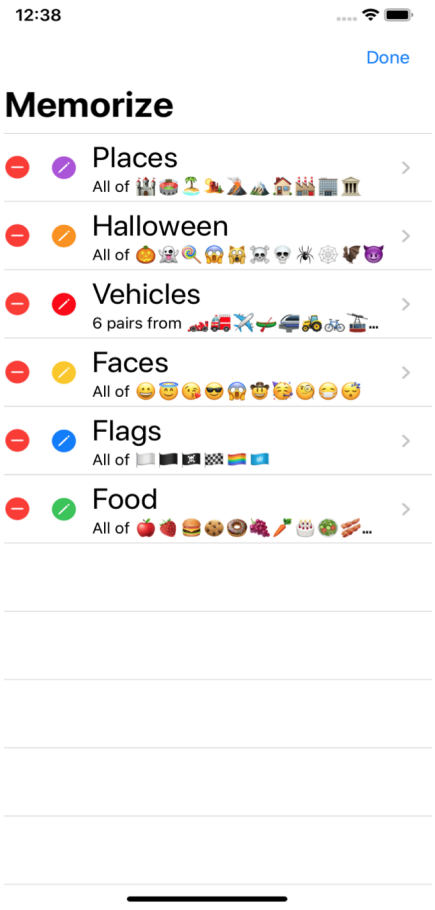
13. В разделе «Обязательные задачи» ничего не говорится о том, какой UI нужно использовать для добавления или удаления эмодзи из вашей темы (*theme*) в Редакторе Тем. Это вам решать.

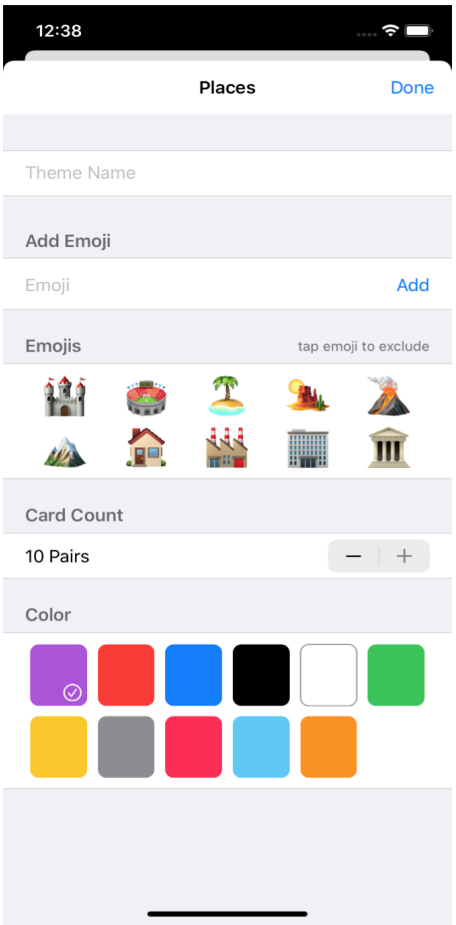
Скриншоты

Мы ненавидим включать снимки экрана или видео с заданиями. Это потому, что нам не нравится склонять вас к тому или иному решению. Однако мы понимаем, что многим из вас сложно «представить», что от вас просят, из письменных описаний. Так что мы делаем исключение для этого задания.

Однако обратите внимание, что этот раздел «Скриншоты» в описании не является обязательной задачей. Обязательные задачи включены выше, а не здесь. Это чисто «примерные» UIs, чтобы дать вам общее представление о том, что вас просят сделать. Вам не нужно создавать свои UIs, чтобы они выглядели именно так (на самом деле, нам бы хотелось увидеть другие творческие подходы)!







Что нужно изучать

Это частичный список концепций, в которых это задание увеличивает ваш опыт работы или демонстрирует ваши знания.

1. **List**
2. **Form**
3. **NavigationView**
4. Модальное представление
5. **TextField**
6. **EditMode**
7. Многочисленные MVVM
8. “издатели” **Publisher**s
9. **UserDefaults**

Оценка (Evaluation)

Во всех заданиях этого семестра требуется написание качественного кода, на основе которого строится приложение без ошибок и предупреждений, следовательно вы должны тестировать полученное приложение до тех пор, пока оно не начнет функционировать правильно согласно поставленной задачи.

Приведем наиболее общие соображения, по которым задание может быть отклонено:

- Приложение не создается.
- Один или более пунктов в разделе **Обязательные задания (Required Tasks)** не выполнены.
- Не понята фундаментальная концепция задания.
- Приложение не создается без предупреждений.
- Код — небрежный или тяжелый для чтения (например, нет отступов и т.д.).
- Ваше решение тяжело (или невозможно) прочитать и понять из-за отсутствия комментариев, из-за плохого наименования методов и переменных, из-за непонятной структуры и т.д.

Часто студенты спрашивают: “Сколько комментариев кода нужно сделать?” Ответ — ваш код должен легко и полностью быть понятным любому, кто его читает.

Дополнительные пункты (Extra Credit)

Мы постарались создать Дополнительные задания так, чтобы они расширили ваши познания в том, что мы проходили на этой неделе. Попытка выполнения по крайней мере некоторых из них приветствуется в плане получения максимального эффекта от этого курса.
Сколько дополнительных кредитов вы заработаете, зависит от объема пункта.
Если вы решите заняться дополнительным пунктом, отметьте его в своем коде комментариями, чтобы ваш проверяющий мог его найти.

1. Поддержка выбора цвета темы в Редакторе Тем.

2. Отслеживайте любыми эмоджи, которые пользователь удаляет из темы, как «удаленные» или «не включенные». Затем улучшите свой Редактор Тем так, чтобы пользователь мог вернуть «удаленные» эмоджи, если передумает. Запомните эти «удаленные» эмоджи навсегда (то есть вам нужно будет добавить их в структуру **struct** вашей темы).