LAB 3

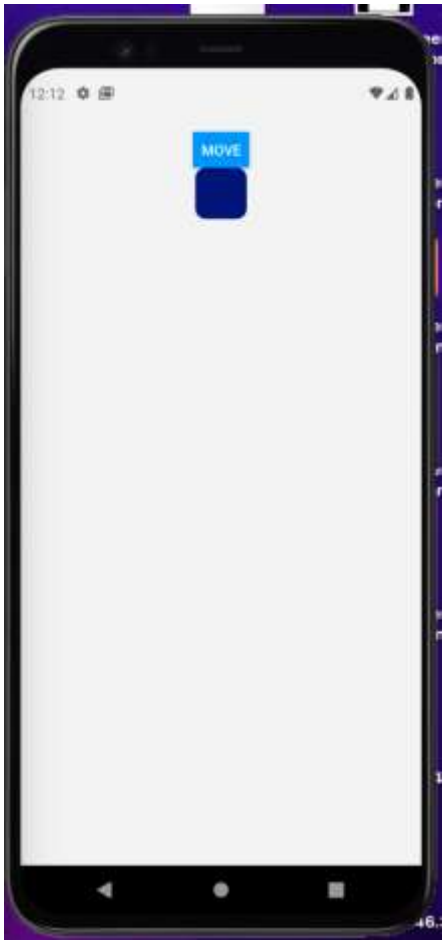
MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

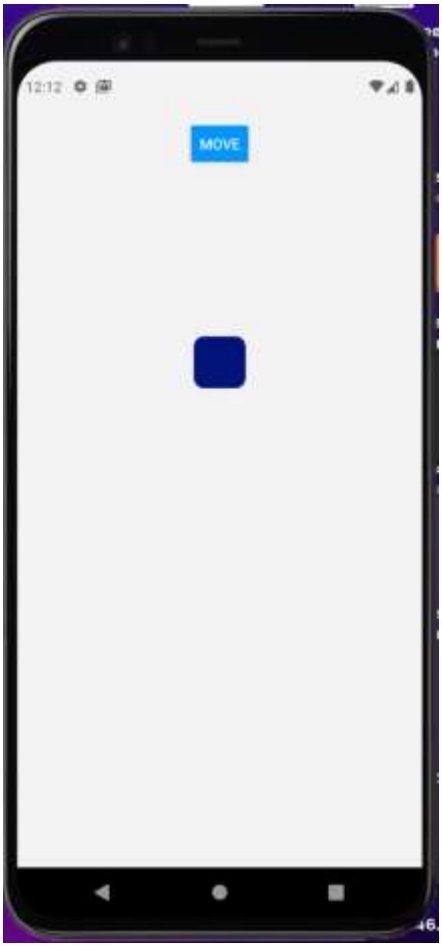
- ✓ Hiểu và xây dựng animation với thư viện Reanimated
- ✓ Tăng trải nghiệm người dùng khi trải nghiệm với animaition
- ✓ Kết hợp, sử dụng nhiều navigation phức tạp

NỘI DUNG

BÀI 1: XÂY DỰNG ANIMATION CHO COMPONENT DI CHUYỂN LÊN XUỐNG



Khi màn hình khởi động



Sau khi nhấn "Move"

Yêu cầu:

- Viết animation di chuyển vị trí cục màu xanh theo chiều dọc, ví trí của cục này được random bất kỳ.

- Hiệu ứng di chuyển phải mượt mà, bắt mắt.

Hướng dẫn:

- Khai báo một biến **offset** được gán giá trị bằng **SharedValue.** Biến này sẽ là vị trí của cục màu xanh của chúng ta.

- Sử dụng hook **useAnimatedStyle** để tạo style cho animation

```
const animatedStyles = useAnimatedStyle(() => {
  return {
    transform: [{translateY: offset.value}],
  };
});
```

- Gán **animatedStyles** vào component hình vuông màu xanh của chúng ta và gán lại giá trị cho **offset** mỗi khi nhấn nút **Move.**

BÀI 2: TẠO ANIMATION VỚI FLATLIST

Màn hình khởi động

Sau khi vuốt xuống

Yêu cầu:

- Khi vuốt lên hoặc vuốt xuống thì item cũng phải từ từ hiện ra giống như trên hình

Hướng dẫn:

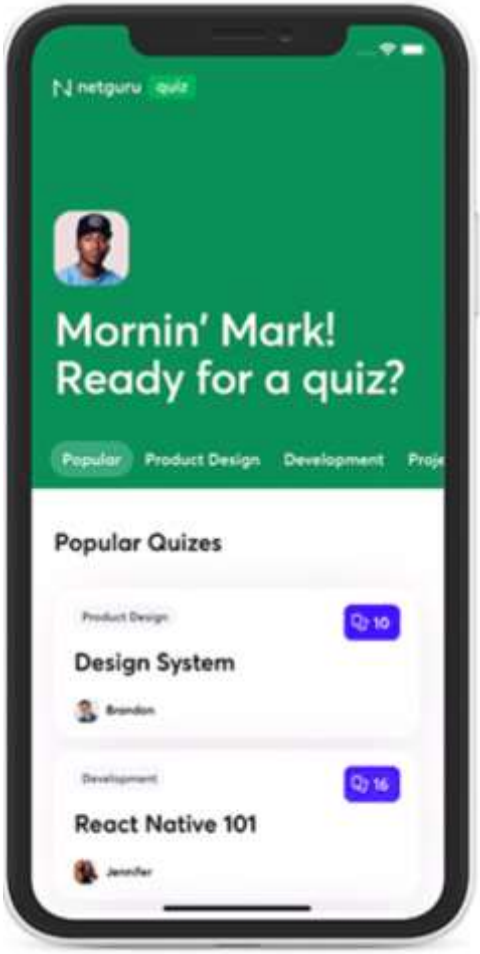
- Trong FlatList có params **onViewableItemsChanged,** đây là params mỗi khi vị trí số lượng item trong danh sách hiển thị thì nó sẽ return ra một danh sách, bao gồm các item đang hiển thị trên màn hình của bạn. Chúng ta sẽ dựa vào danh sách này để tạo animation cho FlatList.

```
const viewableItems = useSharedValue<ViewToken[]>([]);

return (
  <View style={styles.container}>
    <FlatList
      data={data}
      onViewableItemsChanged={({viewableItems: vItems}) => {
        viewableItems.value = vItems;
      }}
      renderItem={({item}) => {
        return <ListItem item={item} viewableItems={viewableItems} />;
      }}
    />
  </View>
```

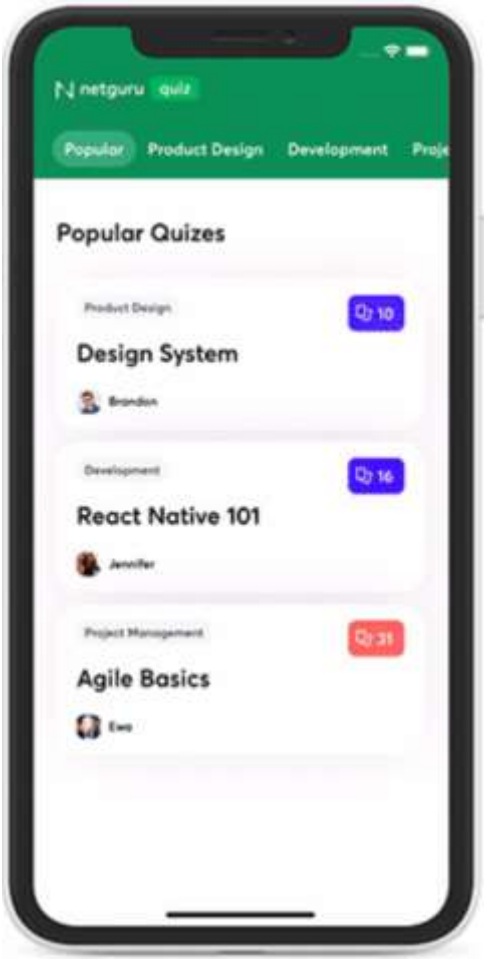
- Ở phần item danh sách này chúng ta sẽ bọc memo lại để trách việc render item lại không cần thiết. Phần đáng chú ý ở đây là biến **isVisible** nó sẽ kiểm tra xem item này có đang hiển thị trên UI mà người dùng thấy được không. Sau đó, sẽ dựa vào nó để tạo hiệu ứng làm mờ **(opacity)**, và tỉ lệ (**scale)** cho item.

```
const ListItem: React.FC<ListItemProps> = React.memo(
  ({item, viewableItems}) => {
    const rStyle = useAnimatedStyle(() => {
      const isVisible = Boolean(
        viewableItems.value
          .filter(item => item.isViewable)
          .find(viewableItem => viewableItem.item.id === item.id),
      );

      return {
        opacity: withTiming(isVisible ? 1 : 0),
        transform: [
          {
            scale: withTiming(isVisible ? 1 : 0.6),
          },
        ],
      };
    }, []);

    return <Animated.View style={[styles.container, rStyle]} />;
  },
);
```

BÀI 3: TẠO SCROLL HEADER NHƯ GIAO DIỆN DƯỚI ĐÂY



Màn hình mặc định



Khi vuốt danh sách lên

Yêu cầu:

- Khi vuốt lên hìnhavatar và tiêu đề trong header phải từ từ thu nhỏ và mờ dần.

- Header luôn trạng thái như hình thứ hai khi nó cuộn lại, header cuộn lại này không được biến mất.

- Tạo animation mượt mà, tự nhiên và không giựt lag.

Hướng dẫn:

- Phần danh sách màu trắng ở dưới các bạn xây dựng **FlatList**, với phần header của **FlatList** là **"Popular Quizes".**
- Phần **header màu xanh** ở trên sẽ là nơi animation của bạn hoạt động, bạn sẽ bắt sự kiện **onScroll** của FlatList để lấy được độ dài khi người dùng cuộn, khi có được độ dài này, bạn thay đổi chiều cao cho header màu xanh đúng theo tỉ lệ người dùng cuộn. Sau đó thêm **animation opacity, scale** và **withSpring** để được animation theo yêu cầu của đề bài

BÀI 4: GV CHO THÊM

*** YÊU CẦU NỘP BÀI:

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.

--- Hết