Лекція-практикум 5

Створити модальне вікно

Оголосіть змінну стану для відображення кнопок

Перш ніж впроваджувати модальне вікно, ми додамо три нові кнопки. Ці кнопки стають видимими після того, як користувач вибере зображення з медіатеки або використає зображення-заповнювач. Одна з цих кнопок запустить модальне вікно вибору емодзі.

У app/(tabs)/index.tsx :

Оголосіть булеву змінну стану , showAppOptionsщоб показати або приховати кнопки, що відкривають модальне вікно, а також кілька інших опцій. Коли екран програми завантажується, ми встановлюємо її на , falseщоб опції не відображалися перед вибором зображення. Коли користувач вибирає зображення або використовує зображення-заповнювач, ми встановлюємо її на true.

Оновіть функцію , pickImageAsync()щоб вона встановлювала значення після того, як користувач вибере зображення.showAppOptionstrue

Оновіть кнопку без теми, додавши onPressprop з наступним значенням.

```tsx
import { View, StyleSheet } from 'react-native';
import * as ImagePicker from 'expo-image-picker';
import { useState } from 'react';

import Button from '@/components/Button';
import ImageViewer from '@/components/ImageViewer';

const PlaceholderImage = require('@/assets/images/background-image.png');

export default function Index() {
  const [selectedImage, setSelectedImage] = useState<string | undefined>(undefined);
  const [showAppOptions, setShowAppOptions] = useState<boolean>(false);

  const pickImageAsync = async () => {
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ['images'],
      allowsEditing: true,
      quality: 1,
    });

    if (!result.canceled) {
      setSelectedImage(result.assets[0].uri);
      setShowAppOptions(true);
    } else {
      alert('You did not select any image.');
    }
  };

  return (
    <View style={styles.container}>
      <View style={styles.imageContainer}>
        <ImageViewer imgSource={PlaceholderImage} selectedImage={selectedImage} />
      </View>
      {showAppOptions ? (
        <View />
      ) : (
        <View style={styles.footerContainer}>
          <Button theme="primary" label="Choose a photo" onPress={pickImageAsync} />
          <Button label="Use this photo" onPress={() => setShowAppOptions(true)} />
        </View>
      )}
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#25292e',
    alignItems: 'center',
  },
  imageContainer: {
    flex: 1,
  },
  footerContainer: {
    flex: 1 / 3,
    alignItems: 'center',
  },
});
```

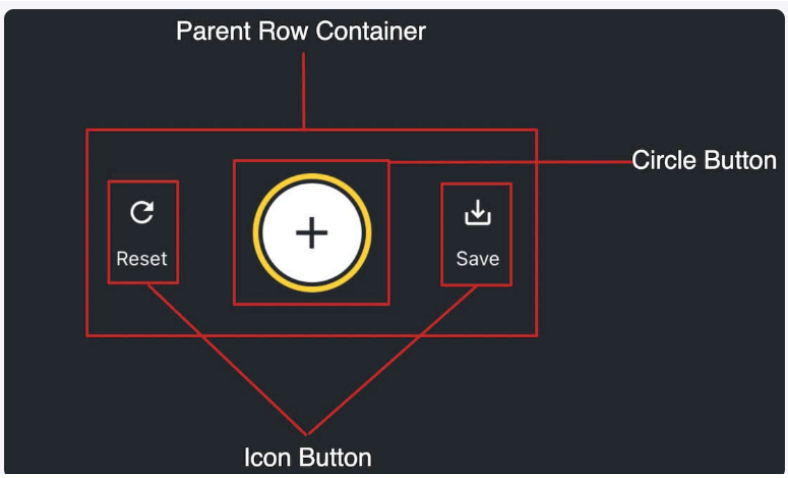
У наведеному вище фрагменті ми рендеримо `Button`компонент на основі значення `showAppOptions`та переміщуємо кнопки в блоці тернарних операторів. Коли значення `showAppOptions` дорівнює `true`, рендеримо порожній `<View>`компонент. Ми розглянемо цей стан на наступному кроці.

Тепер ми можемо видалити ``alert на Button компоненті`` та оновити `onPress`prop`` під час рендерингу другої кнопки в **components/Button.tsx** :

```
<Pressable style={styles.button} onPress={onPress}>
```

Додати кнопки

Давайте розглянемо розташування перемикачів, які ми реалізуємо в цьому розділі. Дизайн виглядає так:



Він містить батьківський елемент <View>із трьома кнопками, вирівняними в ряд. Кнопка посередині зі значком плюса (+) відкриває модальне вікно та має інший стиль, ніж дві інші кнопки.

Усередині каталогу **компонентів** створіть новий файл **CircleButton.tsx** з наступним кодом:

```tsx
import { View, Pressable, StyleSheet } from 'react-native';
import MaterialIcons from '@expo/vector-icons/MaterialIcons';

type Props = {
  onPress: () => void;
};

export default function CircleButton({ onPress }: Props) {
  return (
    <View style={styles.circleButtonContainer}>
      <Pressable style={styles.circleButton} onPress={onPress}>
        <MaterialIcons name="add" size={38} color="#25292e" />
      </Pressable>
    </View>
  );
}

const styles = StyleSheet.create({
  circleButtonContainer: {
    width: 84,
    height: 84,
    marginHorizontal: 60,
    borderWidth: 4,
    borderColor: '#ffd33d',
    borderRadius: 42,
    padding: 3,
  },
  circleButton: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    borderRadius: 42,
    backgroundColor: '#fff',
  },
});
```

Для відображення значка плюса ця кнопка використовує <MaterialIcons>набір значок з @expo/vector-iconsбібліотеки.

Дві інші кнопки також використовуються <MaterialIcons>для відображення вертикально вирівняних текстових підписів та значків. Створіть файл з назвою **IconButton.tsx** у каталозі **компонентів** . Цей компонент приймає три властивості:

icon: назва, що відповідає MaterialIconsзначку бібліотеки.

label: текстовий напис, що відображається на кнопці.

onPress: ця функція викликається, коли користувач натискає кнопку.

```tsx
import { Pressable, StyleSheet, Text } from 'react-native';
import MaterialIcons from '@expo/vector-icons/MaterialIcons';

type Props = {
  icon: keyof typeof MaterialIcons.glyphMap;
  label: string;
  onPress: () => void;
};

export default function IconButton({ icon, label, onPress }: Props) {
  return (
    <Pressable style={styles.iconButton} onPress={onPress}>
      <MaterialIcons name={icon} size={24} color="#fff" />
      <Text style={styles.iconButtonLabel}>{label}</Text>
    </Pressable>
  );
}

const styles = StyleSheet.create({
  iconButton: {
    justifyContent: 'center',
    alignItems: 'center',
  },
  iconButtonLabel: {
    color: '#fff',
    marginTop: 12,
  },
});
```

Усередині app/(tabs)/index.tsx :

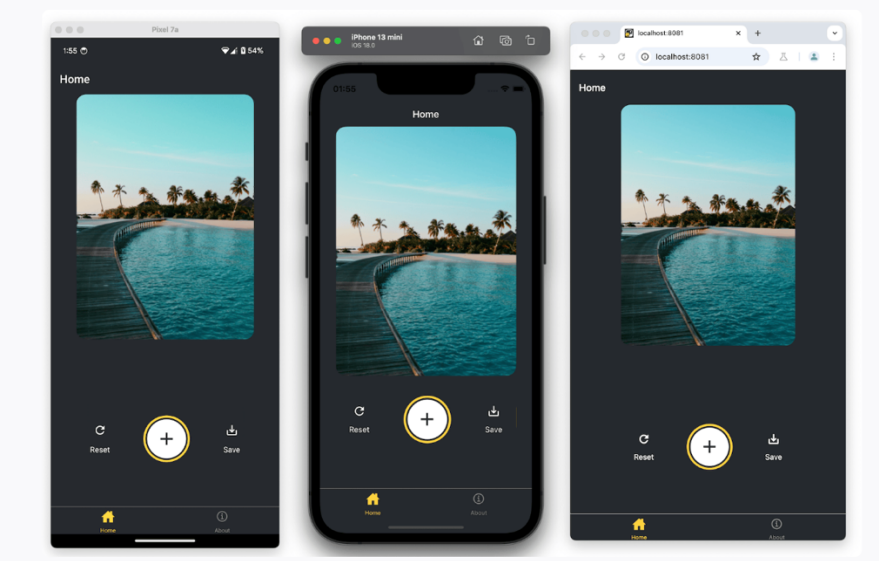
Імпортуйте компоненти `CircleButton`та `IconButton`, щоб відобразити їх.

Додайте три функції-заповнювачі для цих кнопок. `onReset()`Функція викликається, коли користувач натискає кнопку скидання, що призводить до повторного появи кнопки вибору зображення. Ми додамо функціональність для двох інших функцій пізніше.

```
import { View, StyleSheet } from 'react-native';
import * as ImagePicker from 'expo-image-picker';
import { useState } from 'react';

import Button from '@/components/Button';
import ImageViewer from '@/components/ImageViewer';
import IconButton from '@/components/IconButton';
import CircleButton from '@/components/CircleButton';

const PlaceholderImage = require('@/assets/images/background-image.png');

export default function Index() {
  const [selectedImage, setSelectedImage] = useState<string | undefined>(undefined);
  const [showAppOptions, setShowAppOptions] = useState<boolean>(false);

  const pickImageAsync = async () => {
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ['images'],
      allowsEditing: true,
      quality: 1,
    });

    if (!result.canceled) {
      setSelectedImage(result.assets[0].uri);
      setShowAppOptions(true);
    } else {
      alert('You did not select any image.');
    }
  };

  const onReset = () => {
    setShowAppOptions(false);
  };

  const onAddSticker = () => {
    // we will implement this later
  };

  const onSaveImageAsync = async () => {
    // we will implement this later
  };
```

```
  return (
    <View style={styles.container}>
      <View style={styles.imageContainer}>
        <ImageViewer imgSource={PlaceholderImage} selectedImage={selectedImage} />
      </View>
      {showAppOptions ? (
        <View style={styles.optionsContainer}>
          <View style={styles.optionsRow}>
            <IconButton icon="refresh" label="Reset" onPress={onReset} />
            <CircleButton onPress={onAddSticker} />
            <IconButton icon="save-alt" label="Save" onPress={onSaveImageAsync} />
          </View>
        </View>
      ) : (
        <View style={styles.footerContainer}>
          <Button theme="primary" label="Choose a photo" onPress={pickImageAsync} />
          <Button label="Use this photo" onPress={() => setShowAppOptions(true)} />
        </View>
      )}
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#25292e',
    alignItems: 'center',
  },
  imageContainer: {
    flex: 1,
  },
  footerContainer: {
    flex: 1 / 3,
    alignItems: 'center',
  },
  optionsContainer: {
    position: 'absolute',
    bottom: 80,
  },
  optionsRow: {
    alignItems: 'center',
    flexDirection: 'row',
  },
});
```

Давайте розглянемо наш додаток на Android, iOS та в інтернеті:



Створіть модальне вікно вибору емодзі

Модальне вікно дозволяє користувачеві вибрати емодзі зі списку доступних емодзі. Створіть файл EmojiPicker.tsx у каталозі components . Цей компонент приймає три властивості (props):

isVisible: логічне значення для визначення стану видимості модального вікна.

onClose: функція для закриття модального вікна.

children: використовується пізніше для відображення списку емодзі.

```tsx
import { Modal, View, Text, Pressable, StyleSheet } from 'react-native';
import { PropsWithChildren } from 'react';
import MaterialIcons from '@expo/vector-icons/MaterialIcons';

type Props = PropsWithChildren<{
  isVisible: boolean;
  onClose: () => void;
}>;

export default function EmojiPicker({ isVisible, children, onClose }: Props) {
  return (
    <View>
    <Modal animationType="slide" transparent={true} visible={isVisible}>
      <View style={styles.modalContent}>
        <View style={styles.titleContainer}>
          <Text style={styles.title}>Choose a sticker</Text>
          <Pressable onPress={onClose}>
            <MaterialIcons name="close" color="#fff" size={22} />
          </Pressable>
        </View>
        {children}
      </View>
    </Modal>
    </View>
  );
}

const styles = StyleSheet.create({
  modalContent: {
    height: '25%',
    width: '100%',
    backgroundColor: '#25292e',
    borderTopRightRadius: 18,
    borderTopLeftRadius: 18,
    position: 'absolute',
    bottom: 0,
  },
  titleContainer: {
    height: '16%',
    backgroundColor: '#464C55',
    borderTopRightRadius: 10,
    borderTopLeftRadius: 10,
    paddingHorizontal: 20,
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
  },
  title: {
    color: '#fff',
    fontSize: 16,
  },
});
```

Давайте дізнаємося, що робить наведений вище код:

Компонент `<Modal>` відображає заголовок і кнопку закриття.

Його `visible` властивість приймає значення `isVisible` та контролює, чи є модальне вікно відкритим чи закритим.

Його `transparent` властивість — це логічне значення, яке визначає, чи заповнює модальне вікно весь вигляд.

Його `animationType` опора визначає, як він з'являється на екрані та залишає його. У цьому випадку він ковзає знизу екрана.

Нарешті, проп `<EmojiPicker>` викликається `onClose`, коли користувач натискає клавішу close `<Pressable>`.

Тепер давайте змінимо **app/(tabs)/index.tsx** :

Імпортуйте `<EmojiPicker>` компонент.

Створіть `isModalVisible` змінну стану за допомогою `useState` хука. Її значення за замовчуванням — `false`, що приховує модальне вікно, доки користувач не натисне кнопку, щоб відкрити його.

Замініть коментар у `onAddSticker()` функції, щоб оновлювати `isModalVisible` змінну, `true` коли користувач натискає кнопку. Це відкриє вибірник емодзі.

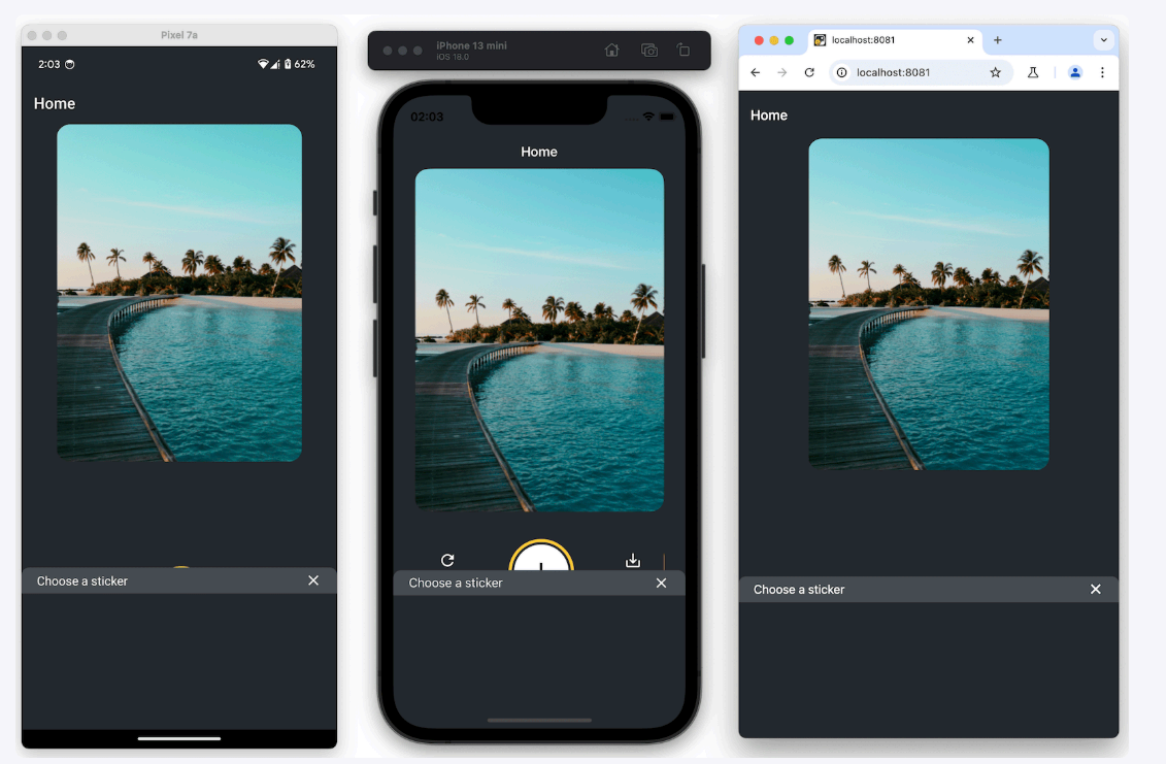
Створіть `onModalClose()` функцію для оновлення `isModalVisible` змінної стану.

Розмістіть `<EmojiPicker>` компонент у нижній частині компонента `Index`.

```
import { View, StyleSheet } from 'react-native';
import * as ImagePicker from 'expo-image-picker';
import { useState } from 'react';

import Button from '@/components/Button';
import ImageViewer from '@/components/ImageViewer';
import IconButton from '@/components/IconButton';
import CircleButton from '@/components/CircleButton';
import EmojiPicker from '@/components/EmojiPicker';

const PlaceholderImage = require('@/assets/images/background-image.png');

export default function Index() {
  const [selectedImage, setSelectedImage] = useState<string | undefined>(undefined);
  const [showAppOptions, setShowAppOptions] = useState<boolean>(false);
  const [isModalVisible, setIsModalVisible] = useState<boolean>(false);

  const pickImageAsync = async () => {
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ['images'],
      allowsEditing: true,
      quality: 1,
    });

    if (!result.canceled) {
      setSelectedImage(result.assets[0].uri);
      setShowAppOptions(true);
    } else {
      alert('You did not select any image.');
    }
  };

  const onReset = () => {
    setShowAppOptions(false);
  };

  const onAddSticker = () => {
    setIsModalVisible(true);
  };

  const onModalClose = () => {
    setIsModalVisible(false);
  };

  const onSaveImageAsync = async () => {
    // we will implement this later
  };
```

```
  return (
    <View style={styles.container}>
      <View style={styles.imageContainer}>
        <ImageViewer imgSource={PlaceholderImage} selectedImage={selectedImage} />
      </View>
      {showAppOptions ? (
        <View style={styles.optionsContainer}>
          <View style={styles.optionsRow}>
            <IconButton icon="refresh" label="Reset" onPress={onReset} />
            <CircleButton onPress={onAddSticker} />
            <IconButton icon="save-alt" label="Save" onPress={onSaveImageAsync} />
          </View>
        </View>
      ) : (
        <View style={styles.footerContainer}>
          <Button theme="primary" label="Choose a photo" onPress={pickImageAsync} />
          <Button label="Use this photo" onPress={() => setShowAppOptions(true)} />
        </View>
      )}
      <EmojiPicker isVisible={isModalVisible} onClose={onModalClose}>
        {/* Emoji list component will go here */}
      </EmojiPicker>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#25292e',
    alignItems: 'center',
  },
  imageContainer: {
    flex: 1,
  },
  footerContainer: {
    flex: 1 / 3,
    alignItems: 'center',
  },
  optionsContainer: {
    position: 'absolute',
    bottom: 80,
  },
  optionsRow: {
    alignItems: 'center',
    flexDirection: 'row',
  },
});
```

Ось результат після цього кроку:



Відобразити список емодзі

Давайте додамо горизонтальний список емодзі до вмісту модального вікна. Ми використаємо <FlatList>для цього компонент з React Native. Створіть файл **EmojiList.tsx** у каталозі **components** та додайте наступний код:

```
import { useState } from 'react';
import { ImageSourcePropType, StyleSheet, FlatList, Platform, Pressable } from 'react-native';
import { Image } from 'expo-image';

type Props = {
  onSelect: (image: ImageSourcePropType) => void;
  onCloseModal: () => void;
};

export default function EmojiList({ onSelect, onCloseModal }: Props) {
  const [emoji] = useState<ImageSourcePropType[]>([
    require("../assets/images/emoji1.png"),
    require("../assets/images/emoji2.png"),
    require("../assets/images/emoji3.png"),
    require("../assets/images/emoji4.png"),
    require("../assets/images/emoji5.png"),
    require("../assets/images/emoji6.png"),
  ]);

  return (
    <FlatList
      horizontal
      showsHorizontalScrollIndicator={Platform.OS === 'web'}
      data={emoji}
      contentContainerStyle={styles.listContainer}
      renderItem={({ item, index }) => (
        <Pressable
          onPress={() => {
            onSelect(item);
            onCloseModal();
          }}>
          <Image source={item} key={index} style={styles.image} />
        </Pressable>
      )}
    />
  );
}

const styles = StyleSheet.create({
  listContainer: {
    borderTopRightRadius: 10,
    borderTopLeftRadius: 10,
    paddingHorizontal: 20,
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
  },
  image: {
    width: 100,
    height: 100,
    marginRight: 20,
  },
});
```

Давайте дізнаємося, що робить наведений вище код:

Компонент <FlatList>вище рендерить усі зображення емодзі за допомогою Imageкомпонента , обгорнутого елементом <Pressable>. Пізніше ми вдосконалимо його, щоб користувач міг торкнутися емодзі на екрані, щоб він відображався як стікер на зображенні.

Він також приймає масив елементів, наданих emoj ізмінною масиву, як значення властивості data. renderItemВластивість бере елемент зі списку datata повертає елемент зі списку. Нарешті, ми додали компоненти Imageта <Pressable>для відображення цього елемента.

Цей horizontalпроп відображає список горизонтально, а не вертикально. showsHorizontalScrollIndicatorВикористовується модуль React Native Platformдля перевірки значення та відображення горизонтальної смуги прокручування у веб-сторінці.

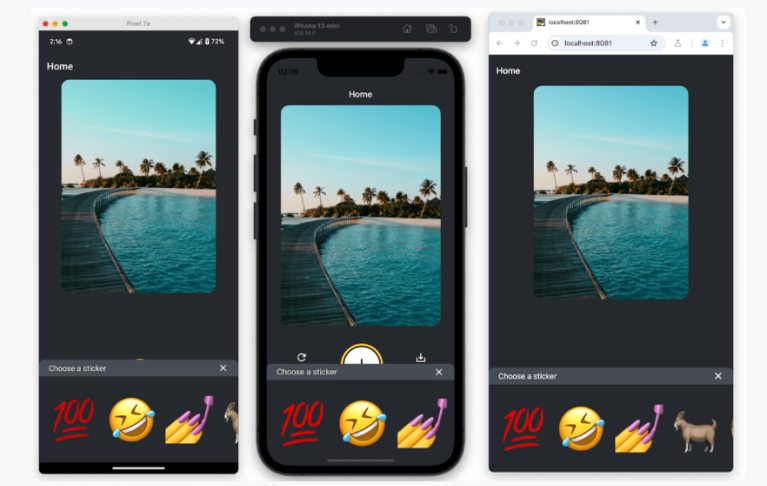
Тепер оновіть **app/(tabs)/index.tsx** , щоб імпортувати <EmojiList>компонент, та замініть коментарі всередині <EmojiPicker>компонента наступним фрагментом коду:

```
import { ImageSourcePropType, View, StyleSheet } from 'react-native';
import * as ImagePicker from 'expo-image-picker';
import { useState } from 'react';

import Button from '@/components/Button';
import ImageViewer from '@/components/ImageViewer';
import IconButton from '@/components/IconButton';
import CircleButton from '@/components/CircleButton';
import EmojiPicker from '@/components/EmojiPicker';
import EmojiList from '@/components/EmojiList';

const PlaceholderImage = require('@/assets/images/background-image.png');

export default function Index() {
  const [selectedImage, setSelectedImage] = useState<string | undefined>(undefined);
  const [showAppOptions, setShowAppOptions] = useState<boolean>(false);
  const [isModalVisible, setIsModalVisible] = useState<boolean>(false);
  const [pickedEmoji, setPickedEmoji] = useState<ImageSourcePropType | undefined>(undefined);

  const pickImageAsync = async () => {
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ['images'],
      allowsEditing: true,
      quality: 1,
    });

    if (!result.canceled) {
      setSelectedImage(result.assets[0].uri);
      setShowAppOptions(true);
    } else {
      alert('You did not select any image.');
    }
  };

  const onReset = () => {
    setShowAppOptions(false);
  };

  const onAddSticker = () => {
    setIsModalVisible(true);
  };

  const onModalClose = () => {
    setIsModalVisible(false);
  };

  const onSaveImageAsync = async () => {
    // we will implement this later
  };
```

```
  return (
    <View style={styles.container}>
      <View style={styles.imageContainer}>
        <ImageViewer imgSource={PlaceholderImage} selectedImage={selectedImage} />
      </View>
      {showAppOptions ? (
        <View style={styles.optionsContainer}>
          <View style={styles.optionsRow}>
            <IconButton icon="refresh" label="Reset" onPress={onReset} />
            <CircleButton onPress={onAddSticker} />
            <IconButton icon="save-alt" label="Save" onPress={onSaveImageAsync} />
          </View>
        </View>
      ) : (
        <View style={styles.footerContainer}>
          <Button theme="primary" label="Choose a photo" onPress={pickImageAsync} />
          <Button label="Use this photo" onPress={() => setShowAppOptions(true)} />
        </View>
      )}
      <EmojiPicker isVisible={isModalVisible} onClose={onModalClose}>
        <EmojiList onSelect={setPickedEmoji} onCloseModal={onModalClose} />
      </EmojiPicker>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#25292e',
    alignItems: 'center',
  },
  imageContainer: {
    flex: 1,
  },
  footerContainer: {
    flex: 1 / 3,
    alignItems: 'center',
  },
  optionsContainer: {
    position: 'absolute',
    bottom: 80,
  },
  optionsRow: {
    alignItems: 'center',
    flexDirection: 'row',
  },
});
```

У EmojiListкомпоненті onSelectprop вибирає емодзі, і після його вибору onCloseModalзакриває модальне вікно.

Давайте розглянемо наш додаток на Android, iOS та в інтернеті:



Відобразити вибраний емодзі

Тепер ми розмістимо наклейку емодзі на зображенні. Створіть новий файл у каталозі components та назвіть його EmojiSticker.tsx . Потім додайте наступний код:

```
import { ImageSourcePropType, View } from 'react-native';
import { Image } from 'expo-image';

type Props = {
  imageSize: number;
  stickerSource: ImageSourcePropType;
};

export default function EmojiSticker({ imageSize, stickerSource }: Props) {
  return (
    <View style={{ top: -350 }}>
      <Image source={stickerSource} style={{ width: imageSize, height: imageSize }} />
    </View>
  );
}
```

Цей компонент отримує два пропи:

imageSize: значення, визначене всередині Indexкомпонента. Ми використовуватимемо це значення в наступному розділі для масштабування розміру зображення при торканні.

stickerSource: джерело вибраного зображення емодзі.

Імпортуйте цей компонент у файл app/(tabs)/index.tsx та оновіть Indexкомпонент, щоб він відображав стікер емодзі на зображенні. Ми перевіримо, чи pickedEmojiстан не відповідає undefined:

```
import { ImageSourcePropType, View, StyleSheet } from 'react-native';
import * as ImagePicker from 'expo-image-picker';
import { useState } from 'react';


import Button from '@/components/Button';
import ImageViewer from '@/components/ImageViewer';
import IconButton from '@/components/IconButton';
import CircleButton from '@/components/CircleButton';
import EmojiPicker from '@/components/EmojiPicker';
import EmojiList from '@/components/EmojiList';
import EmojiSticker from '@/components/EmojiSticker';

const PlaceholderImage = require('@/assets/images/background-image.png');

export default function Index() {
  const [selectedImage, setSelectedImage] = useState<string | undefined>(undefined);
  const [showAppOptions, setShowAppOptions] = useState<boolean>(false);
  const [isModalVisible, setIsModalVisible] = useState<boolean>(false);
  const [pickedEmoji, setPickedEmoji] = useState<ImageSourcePropType | undefined>(undefined);


  const pickImageAsync = async () => {
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ['images'],
      allowsEditing: true,
      quality: 1,
    });

    if (!result.canceled) {
      setSelectedImage(result.assets[0].uri);
      setShowAppOptions(true);
    } else {
      alert('You did not select any image.');
    }
  };

  const onReset = () => {
    setShowAppOptions(false);
  };

  const onAddSticker = () => {
    setIsModalVisible(true);
  };

  const onModalClose = () => {
    setIsModalVisible(false);
  };

  const onSaveImageAsync = async () => {
    // we will implement this later
  };
```

```
  return (
    <View style={styles.container}>
      <View style={styles.imageContainer}>
        <ImageViewer imgSource={PlaceholderImage} selectedImage={selectedImage} />
        {pickedEmoji && <EmojiSticker imageSize={40} stickerSource={pickedEmoji} />}
      </View>
      {showAppOptions ? (
        <View style={styles.optionsContainer}>
          <View style={styles.optionsRow}>
            <IconButton icon="refresh" label="Reset" onPress={onReset} />
            <CircleButton onPress={onAddSticker} />
            <IconButton icon="save-alt" label="Save" onPress={onSaveImageAsync} />
          </View>
        </View>
      ) : (
        <View style={styles.footerContainer}>
          <Button theme="primary" label="Choose a photo" onPress={pickImageAsync} />
          <Button label="Use this photo" onPress={() => setShowAppOptions(true)} />
        </View>
      )}
      <EmojiPicker isVisible={isModalVisible} onClose={onModalClose}>
        <EmojiList onSelect={setPickedEmoji} onCloseModal={onModalClose} />
      </EmojiPicker>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#25292e',
    alignItems: 'center',
  },
  imageContainer: {
    flex: 1,
  },
  footerContainer: {
    flex: 1 / 3,
    alignItems: 'center',
  },
  optionsContainer: {
    position: 'absolute',
    bottom: 80,
  },
  optionsRow: {
    alignItems: 'center',
    flexDirection: 'row',
  },
});
```

Давайте розглянемо наш додаток на Android, iOS та в інтернеті: