LAB 2

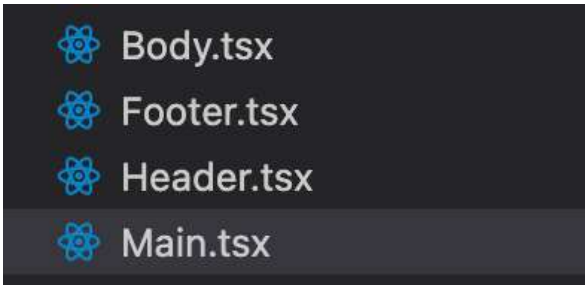
MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Làm quen với hooks trong React Native
- ✓ Ứng dụng kiến thực được học vào ứng dụng

NỘI DUNG

BÀI 1: CẤU TRÚC PROJECT ỨNG DỤNG



Hướng dẫn:
- Tạo mới 4 file bao gồm: **Body.tsx, Footer.tsx, Header.tsx, Main.tsx.**
- File **Main.tsx** sẽ là giao diện chính, chứa 3 file thành phần còn lại
- Code file **Main.tsx** theo hướng dẫn bên dưới đây:

```tsx
export type UserType = {
  name: string;
  avatar: string;
};

export default function Main() {
  return (
    <View style={styles.container}>
      <Header />
      <Body />
      <Footer />
    </View>
  );
}
```

- Code file **Header.tsx** theo hướng dẫn bên dưới đây:

```tsx
type HeaderType = {
  user: UserType;
};

export const Header = () => {
  console.log('re-render header');
  return (
    <View>
      <Text>Header</Text>
    </View>
  );
};
```

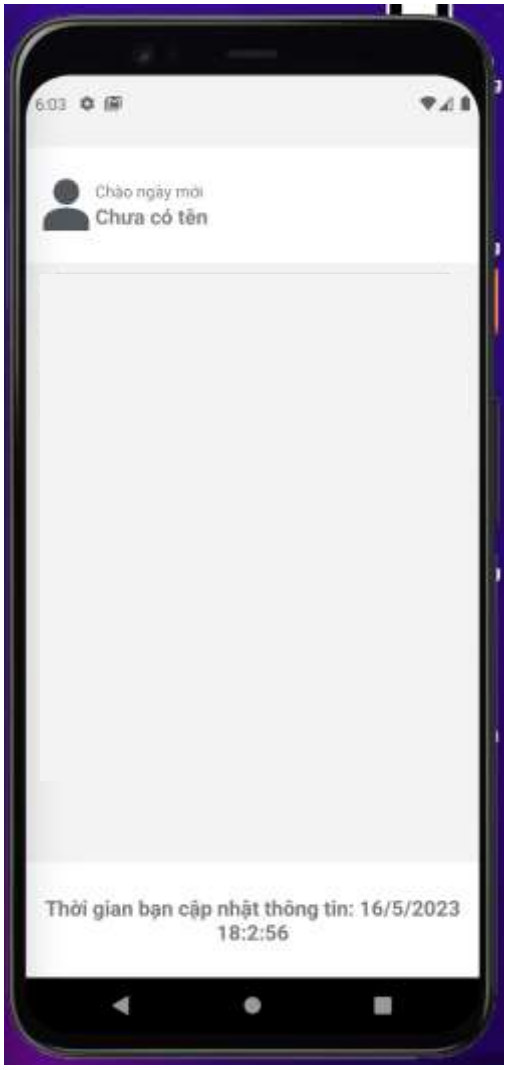
- Code file **Body.tsx** theo hướng dẫn bên dưới đây:

```tsx
export const Body = () => {
  return (
    <View>
      <Text>Body</Text>
    </View>
  );
};
```

- Code file **Footer.tsx** theo hướng dẫn bên dưới đây:

```tsx
export const Footer = () => {
  return (
    <View>
      <Text>Footer</Text>
    </View>
  );
};
```

BÀI 2: XÂY DỰNG ỨNG DỤNG CÓ GIAO DIỆN NHƯ BÊN DƯỚI



Yêu cầu:
- Bọc memo vào **Header** và **Footer**
- Viết các hàm xử lý có **useCallBack**, **useEffect** xử lý hàm và dữ liệu.

Hướng dẫn:

- Viết các hàm xử lý cho file **Main.tsx**:

```tsx
const colors = ['white', 'gray', 'yellow', 'red', 'blue', 'orange'];

export type UserType = {
  name: string;
  avatar: string;
};

export default function Main() {
  const [user, setUser] = useState<UserType>({
    name: 'Chưa có tên',
    avatar:
      'https://upload.wikimedia.org/wikipedia/commons/thumb/5/59/
      User-avatar.svg/2048px-User-avatar.svg.png',
  });
  const [lastTimeUpdate, setLastTimeUpdate] = useState(
    'Bạn chưa cập nhật thông tin',
  );
  const [footerColor, setFooterColor] = useState(colors[0]);

  // Cập nhật thông tin cho tài khoản
  const handleUpdateInfor = useCallback((_user: UserType) => {
    setUser(_user);
  }, []);

  // Hàm random màu cho background của Footer
  const handleRandomColor = useCallback(() => {
    const numberRan = Math.floor(Math.random() * colors.length);
    setFooterColor(colors[numberRan]);
  }, []);
```

```
// Hàm random màu cho background của Footer
const handleRandomColor = useCallback(() => {
  const numberRan = Math.floor(Math.random() * colors.length);
  setFooterColor(colors[numberRan]);
}, []);

// Mỗi lần thông tin user thay đổi, sẽ cập nhật lại thời gian sửa đổi
useEffect(() => {
  const currentdate = new Date();
  const datetime =
    currentdate.getDate() +
    '/' +
    (currentdate.getMonth() + 1) +
    '/' +
    currentdate.getFullYear() +
    ' ' +
    currentdate.getHours() +
    ':' +
    currentdate.getMinutes() +
    ':' +
    currentdate.getSeconds();
  setLastTimeUpdate(datetime);
}, [user]);

return (
  <View style={styles.container}>
    <Header user={user} />
    <Body
      onUpdateInfor={handleUpdateInfor}
      onClickChangeBgFooter={handleRandomColor}
    />
    <Footer timeUpdate={lastTimeUpdate} backgroundColor={footerColor} />
  </View>
);
}
```

- Viết các hàm xử lý cho file **Header.tsx** có bọc memo:

```tsx
type HeaderType = {
  user: UserType;
};

export const Header: FC<HeaderType> = memo(props => {
  const {user} = props;
  console.log('re-render header');
  return (
    <View
      style={containerStyle({
        height: 100,
        backgroundColor: 'white',
        padding: 10,
        flexDirection: 'row',
        alignItems: 'center',
      })}>
      <Image
        resizeMode="center"
        style={styles.avatar}
        source={{uri: user.avatar}}
      />
      <View>
        <Text>Chào ngày mới</Text>
        <Text style={styles.name}>{user.name}</Text>
      </View>
    </View>
  );
});
```

- Viết các hàm xử lý cho file **Footer.tsx** có bọc memo:

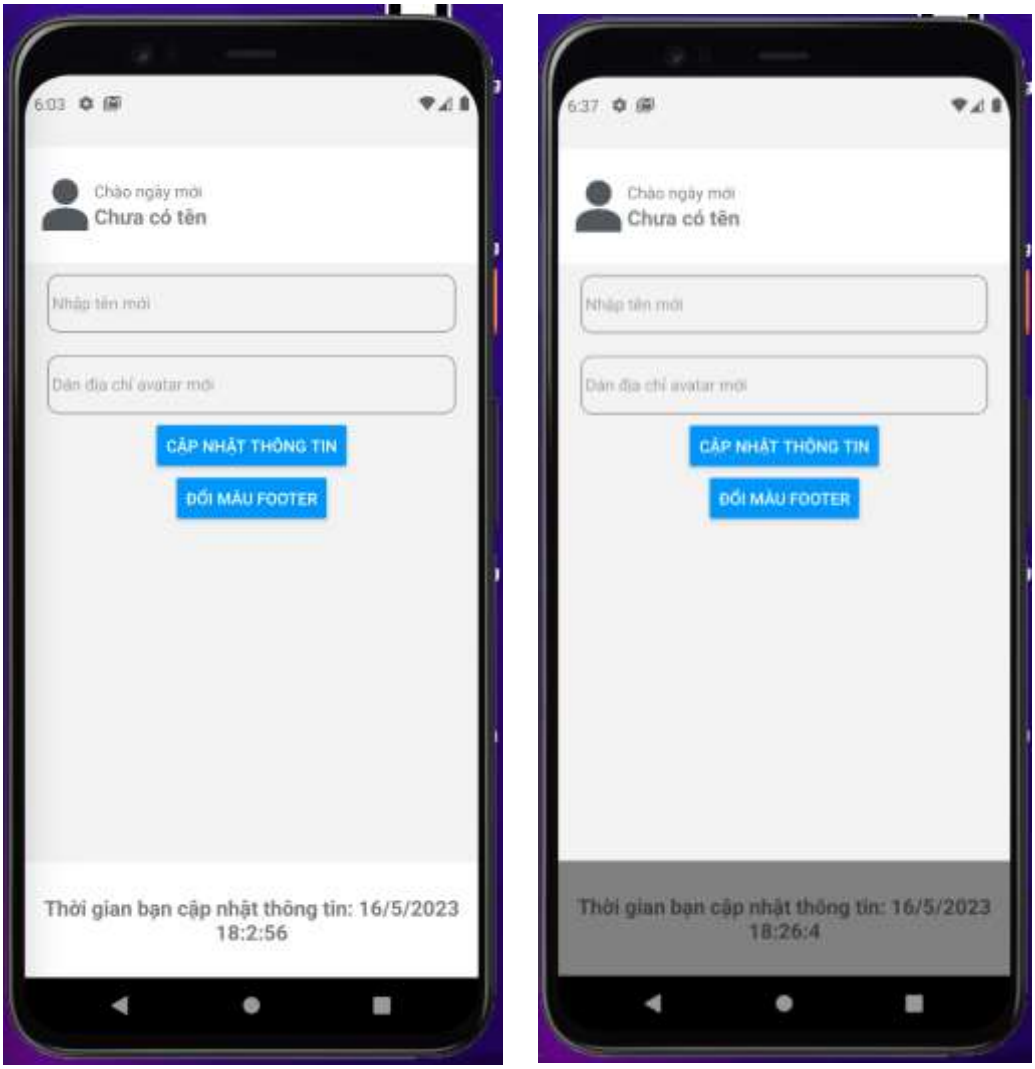
```tsx
type FooterType = {
  timeUpdate: string;
  backgroundColor: string;
};

export const Footer: FC<FooterType> = memo(props => {
  const {timeUpdate, backgroundColor} = props;
  return (
    <View
      style={containerStyle({
        height: 100,
        backgroundColor: backgroundColor,
        alignItems: 'center',
        justifyContent: 'center',
      })}>
      <Text style={styles.text}>
        Thời gian bạn cập nhật thông tin: {timeUpdate}
      </Text>
    </View>
  );
});

const styles = StyleSheet.create({
  text: {
    fontSize: 18,
    fontWeight: 'bold',
    textAlign: 'center',
  },
});

// styleText này sẽ nhận tất cả props style mà thẻ Text có
const containerStyle = (props: ViewStyle) => ({
  ...props,
});
```

BÀI 3: VIẾT BODY CỦA ỨNG DỤNG CÓ GIAO DIỆN NHƯ SAU:



Yêu cầu:

- Nhấn nút **"ĐỔI MÀU FOOTER",** background sẽ tự random đổi màu.

- User có thể thay đổi thông tin trên **Header** bằng cách nhập vào thông tin.

- Khi nhập thông tin vào TextInput, component Header và Footer không bị re-render.

- Khi nhấn “Cập nhật thông tin” mới re-render Header và Footer để đẩy dữ liệu lên hiển thị.

Hướng dẫn:

- Tạo 2 state lưu trữ giá trị của name và linkImage.
- Kiểm tra field đã được nhập chưa, nếu chưa nhập thông báo không được để trống.
- Gọi hàm **onUpdateInfor** để cập nhật lại thông tin header.
- Gọi hàm **onClickChangeBgFooter** để cập nhật lại màu nền **Footer.**

```
type BodyType = {
  onUpdateInfor: (user: UserType) => void;
  onClickChangeBgFooter: () => void;
};

export const Body: FC<BodyType> = memo(props => {
  const {onUpdateInfor, onClickChangeBgFooter} = props;

  const [name, setName] = useState('');
  const [linkImage, setLinkImage] = useState('');

  const handleChangeInfo = () => {
    if (name.length > 0 && linkImage.length > 0) {
      onUpdateInfor({name, avatar: linkImage});
    } else {
      ToastAndroid.show('Không được để trống', ToastAndroid.SHORT);
    }
  };
```

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