ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO CUỐI KÌ

ĐỒ ÁN PHÁT TRIỂN ỨNG DỤNG ĐA PHƯƠNG TIỆN

Đề tài: HABIT TRACKER

Giảng viên hướng dẫn:

PHẠM NGUYỄN TRƯỜNG AN

Nhóm sinh viên thực hiện:

CAO THẾ THUẬN – 20520793

NGUYỄN QUỐC THÁI – 20520304

NGUYỄN TRUNG NHÂN – 20520670

ĐINH PHƯƠNG NAM – 20520641

Link repo Github: https://github.com/MickeyNeo/Habit_Tracker

Thành phố Hồ Chí Minh, 2022

MỤC LỤC

LỜI CẢM ƠN

Lời đầu tiên nhóm thực hiện đồ án xin gửi lời cảm ơn đến thầy Phạm Nguyễn Trường An, thầy đã nhiệt tình giảng dạy trên lớp, hỗ trợ những thông tin cần thiết, giải đáp những thắc mắc và góp ý cho nhóm và các bạn trong suốt quá trình thực hiện đề tài.

Đồng thời nhóm gửi lời cảm ơn các anh chị khóa trên, đặc biệt là anh chị trong khoa đã chia sẻ kinh nghiệm quý báu về môn học cũng như những kiến thức liên quan. Cũng như bạn bè đã tạo những điều kiện thuận lợi, mọi người đã đưa ra nhận xét và góp ý chân thành vô cùng quý giá, những người đã động viên, hỗ trợ nhóm hoàn thành đề tài.

Do đây là lần đầu tiên chúng em xây dựng một ứng dụng di động và thời gian, kiến thức còn hạn chế nên không thể tránh khỏi những thiếu sót. Chính vì vậy nhóm rất mong được nhận những góp ý nhằm hoàn thiện hơn những kiến thức mà nhóm đã học tập và làm hành trang để thực hiện các đề tài khác trong tương lai.

Chân thành cảm ơn!

Nhóm thực hiện

Thủ Đức, tháng 2 năm 2023

CHƯƠNG 1: GIỚI THIỆU ĐỒ ÁN - HABIT TRACKER

Giới thiệu app:

Mỗi ngày, cuộc sống của chúng ta được quản lý bởi những thói quen. Các thói quen này có thể là các việc làm nhỏ như đánh răng, gấp chăn, … cũng như cách thức làm việc của bạn, học từ vựng, đọc sách,… Thật là bất ngờ khi có một số người thậm chí không để ý tới thói quen hàng ngày của mình.

Một số habit như hút thuốc có thể ảnh hưởng xấu tới sức khỏe của chúng ta. Trong khi đó, một số thói quen khác như trì hoãn có thể phá hoại công việc của chúng ta.

Vì thế nên chúng ta phải bắt đầu làm chủ cuộc sống của mình. Tuy nhiên, để thành công trong việc này sẽ rất khó vì hầu hết các thói quen của một người đã in hằn trong tiềm thức của người đó, đến mức họ có thể làm việc đó mà không suy nghĩ gì cả.

Hiểu được điều đó, nên các app Habit tracker ra đời, để giúp người dùng có thể làm chủ ngày hôm nay thông qua việc tạo điều kiện thực hiện các thói quen tốt và nhắc nhở tránh xa các thói quen xấu. Với những chuỗi ngày hoàn hảo như thế thì người dùng có thể đạt được tất cả mục tiêu họ đặt ra một cách dễ dàng.

Nhóm người dùng:

- Những người có hoài bão, có mục tiêu cần hướng đến.
- Những người muốn làm chủ bản thân, thay thế các thói quen độc hại bằng các thói quen tốt.
- Những người hay quên, dễ bị phân tâm, hoặc khó tập trung, cần một thứ gì đó để note lại việc làm hằng ngày.

Các chức năng cần có:

- Tạo habit
- Phân loại thói quen: sức khỏe, công việc, thư giãn, ...
- Thống kê thành tích bằng số liệu, bảng, hình ảnh, ...
- Nhắc nhở làm habit
- Nhật kí thói quen
- Tiện ích: đồng hồ bấm giờ, nhạc thư giãn, …

So sánh với app trên App Store:

- Các chức năng làm được:
    - Tạo thói quen
    - Phân loại thói quen
    - Nhật kí thói quen
    - Tạo screen Add habit như trên app, cho user có nhiều tùy chọn
    - Đầy đủ các chức năng thống kê thành tích
    - Một số tiện ích: Đồng hồ,..
    - Xử lí logic trên database
    - Các chức năng cơ bản trong setting như theme, custom tab bar, tag manager
      Phần bổ sung sau vấn đáp
    - Tự custom thêm unit, tag: user có thể thêm, xóa, sửa theo ý muốn
    - Hoàn thiện screen habit details đầy đủ chức năng như app gốc
    - Bổ sung filter theo tag ở màn hình Home
    - Chỉnh sửa thêm giao diện sao cho hợp lí dễ nhìn

- Một habit có thể thuộc nhiều tag khác nhau

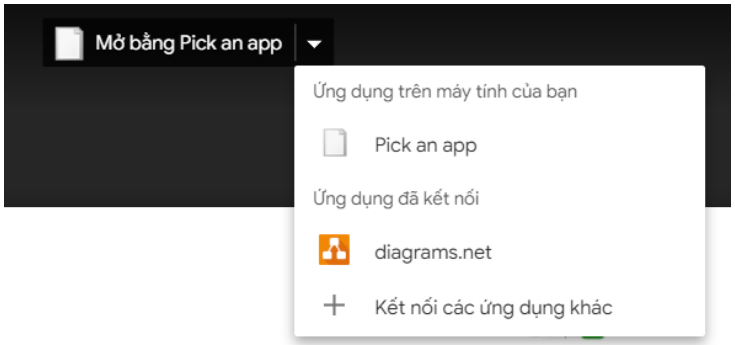
- Các chức năng chưa làm được:
  - Nhắc nhở
  - Sound: âm thanh
  - Vacation mode: Xét thời gian cho kì nghỉ để các thông báo nhắc nhỡ habit từng ngày sẽ không hiện thị thông báo
  - Safety lock: khóa app bằng mật khẩu ➔ cá nhân hóa
  - Chức năng group bạn bè
  - Chưa có chức năng xuất báo cáo report (premium chưa được trải nghiệm)
  - Chức năng đổi ngôn ngữ khác (đã có làm nhưng chưa thay đổi ngôn ngữ được)

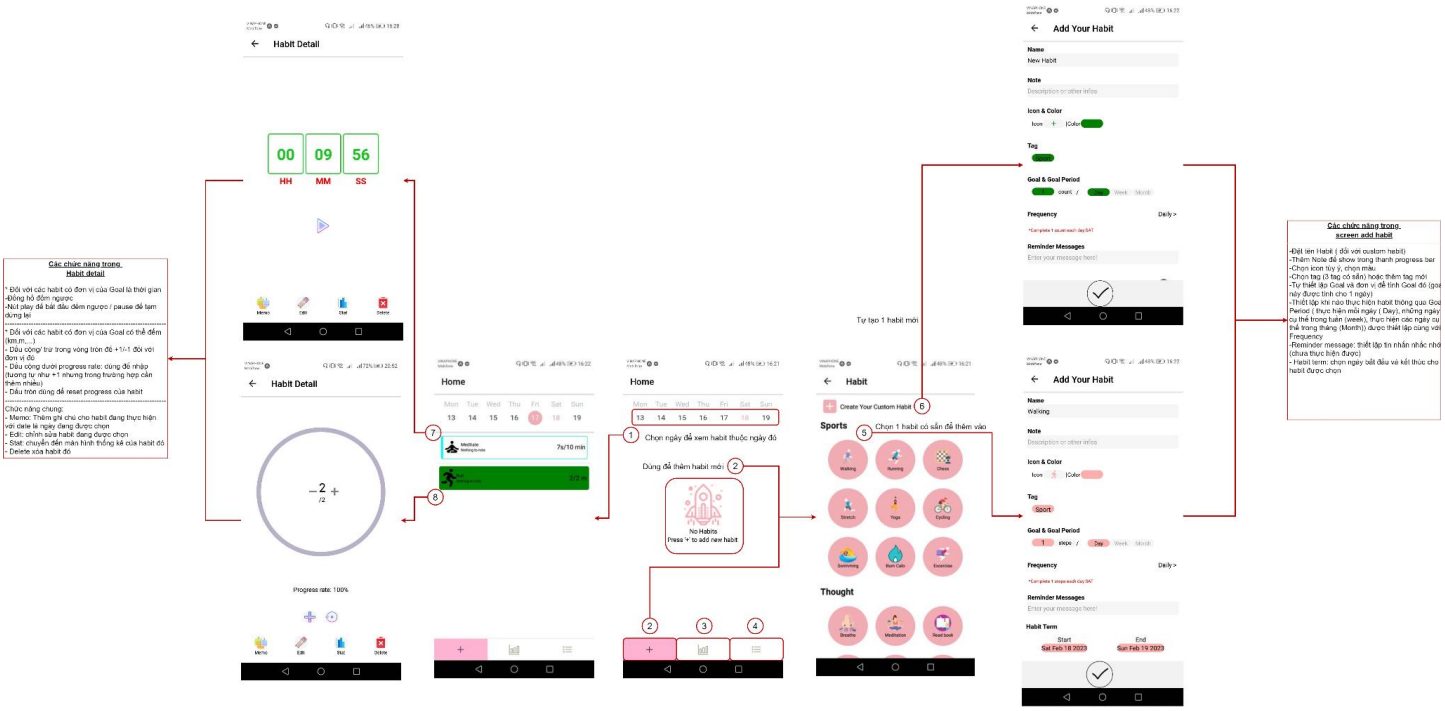
CHƯƠNG 2: BÁO CÁO ĐỒ ÁN

Sơ đồ tổng quát

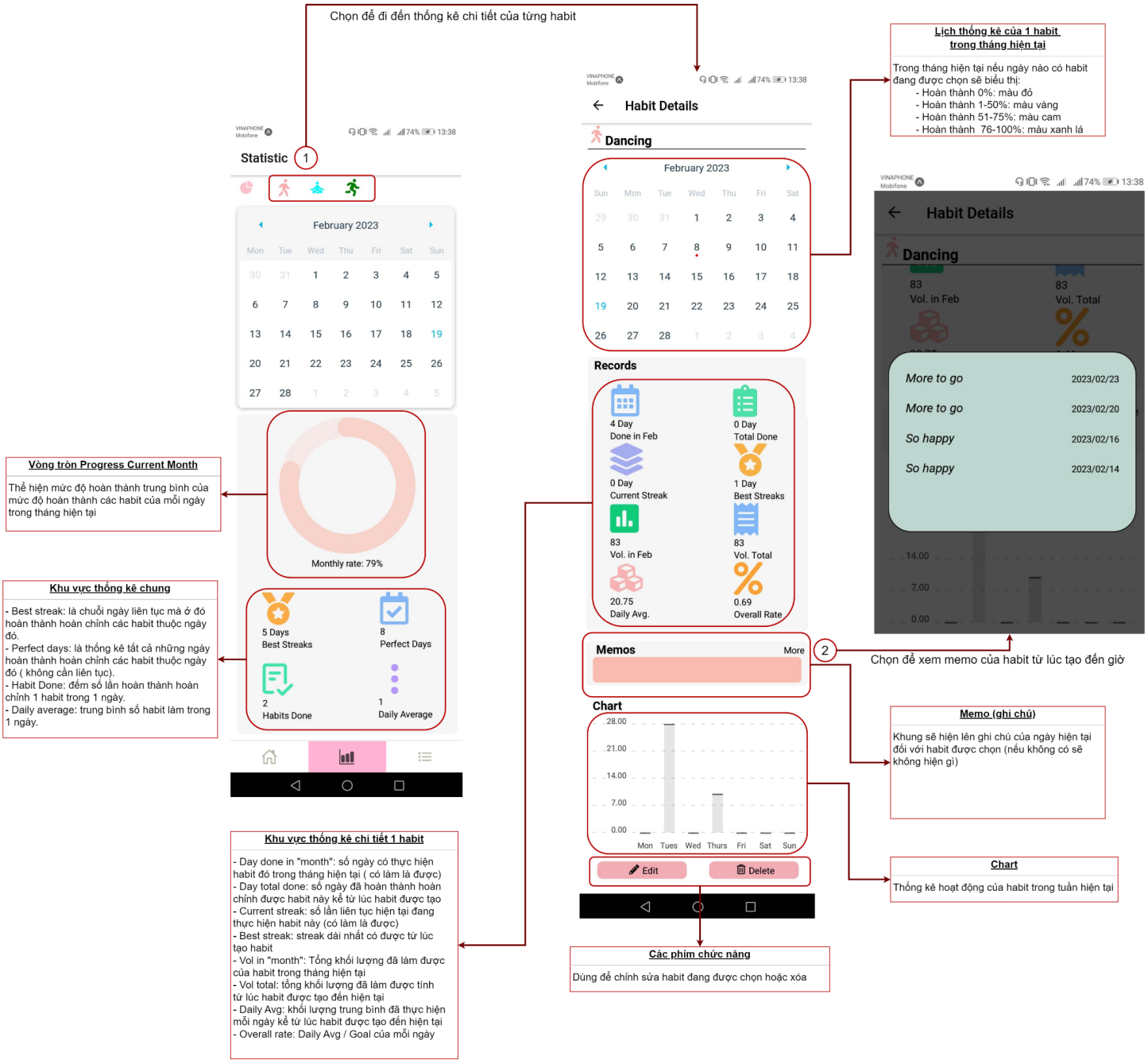
Sơ đồ tổng quát của ứng dụng được chúng em vẽ trên web diagram theo link sau: https://drive.google.com/file/d/17zZKbQhn5igcbZdyrtODFhVNNmhfx4dx/view?usp=sharing

(Để xem đầy đủ bằng diagram: Truy cập vào link => Chọn pick an app => Chọn diagram.net

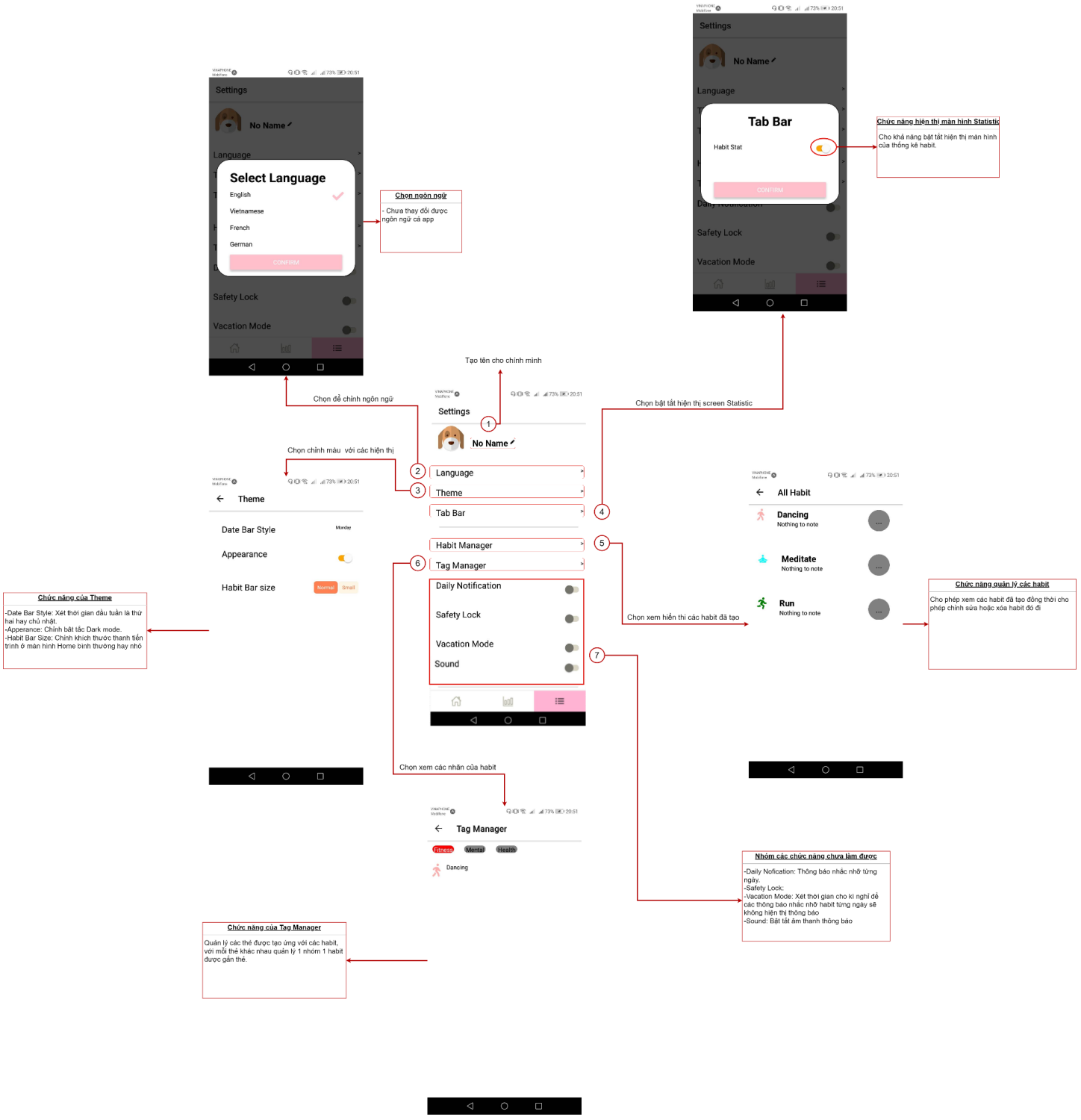




Hình 1. Màn hình add habit (2)



Hình 2. Màn hình statistic (3)



Hình 3. Màn hình setting (4)

Với:

+ Trang đầu : “Màn hình add habit (2)” : đây là màn hình chính và màn hình thêm habit mới.
+ Trang thứ 2: “Màn hình statistic (3)”: đây là màn hình thống kê

+ Trang thứ 3: “Màn hình setting (4)”: đây là màn hình cài đặt

Database

HABIT

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
id	TEXT		Mã habit
name	TEXT	PRIMARY KEY	Tên habit
note	TEXT		Mô tả habit
frequency	TEXT	NOT NULL	Tần suất thực hiện
color	TEXT	NOT NULL	Màu sắc
frequency type	TEXT	CHECK IN (Daily, Weekly, Monthly)	Loại tần suất (Mỗi ngày, mỗi tuần, mỗi tháng)
timeRange	TEXT	CHECK IN (Morning, Afternoon, Evening)	Khoảng thời gian thực hiện
reminderMessage	TEXT		Lời nhắc
showMemo	INTEGER	CHECK IN (0, 1)	Tắt mở memo khi hoàn thành habit
chartType	INTEGER	CHECK IN (0, 1)	Loại chart

habitStartDate	TEXT	NOT NULL	Ngày bắt đầu habit
habitEndDate	TEXT		Ngày kết thúc habit
goalNo	INTEGER	NOT NULL	Mục tiêu thực hiện
goalPeriod	TEXT	NOT NULL	Khoảng thời gian lặp mục tiêu
unitID	INTEGER		Mã đơn vị
Icon	TEXT		Tên Icon
iconFamily	TEXT		Tên Icon family
flag	INTEGER		Cờ để hiển thị dấu check trong màn hình habit mặc định
tag	TEXT		Tên tag

TAG

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
Id	INTEGER	AUTO INCREMENT, PRIMARY KEY	Khóa chính
Name	TEXT	NOT NULL, UNIQUE	Tên tag

HAVETAG

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả

habitName	TEXT	PRIMARY KEY	Tên habit
tagID	INTEGER	PRIMARY KEY	ID của tag

MEMO

Thuộc tính	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
habitName	TEXT	PRIMARY KEY	Tên habit
date	TEXT	PRIMARY KEY	Ngày viết memo
content	TEXT		Nội dung memo
progress	INTEGER		Tiến trình làm habit

REMINDER

Thuộc tính	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
habitName	TEXT	PRIMARY KEY	Tên habit
time	TEXT	PRIMARY KEY	Thời gian nhắc nhở

UNIT

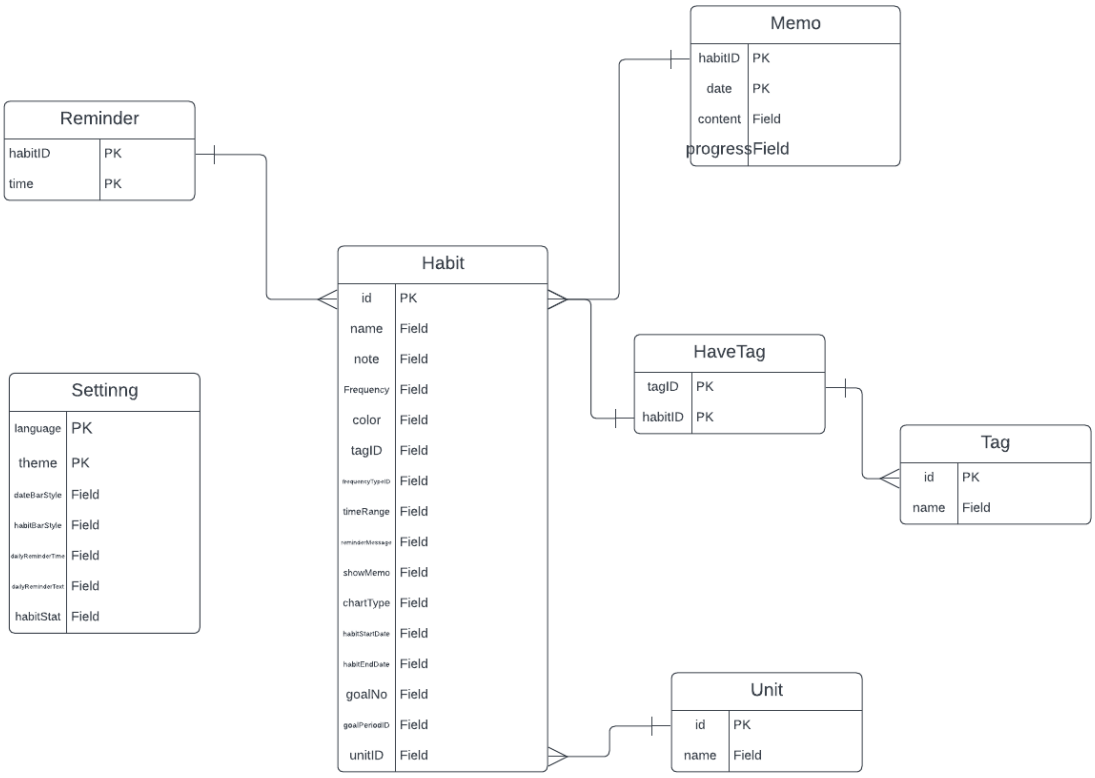
Thuộc tính	**Kiểu dữ liệu**	**Ràng buộc**	**Mô tả**
Id	INTEGER	PRIMARY KEY	Mã đơn vị

name	TEXT	NOT NULL	Tên đơn vị

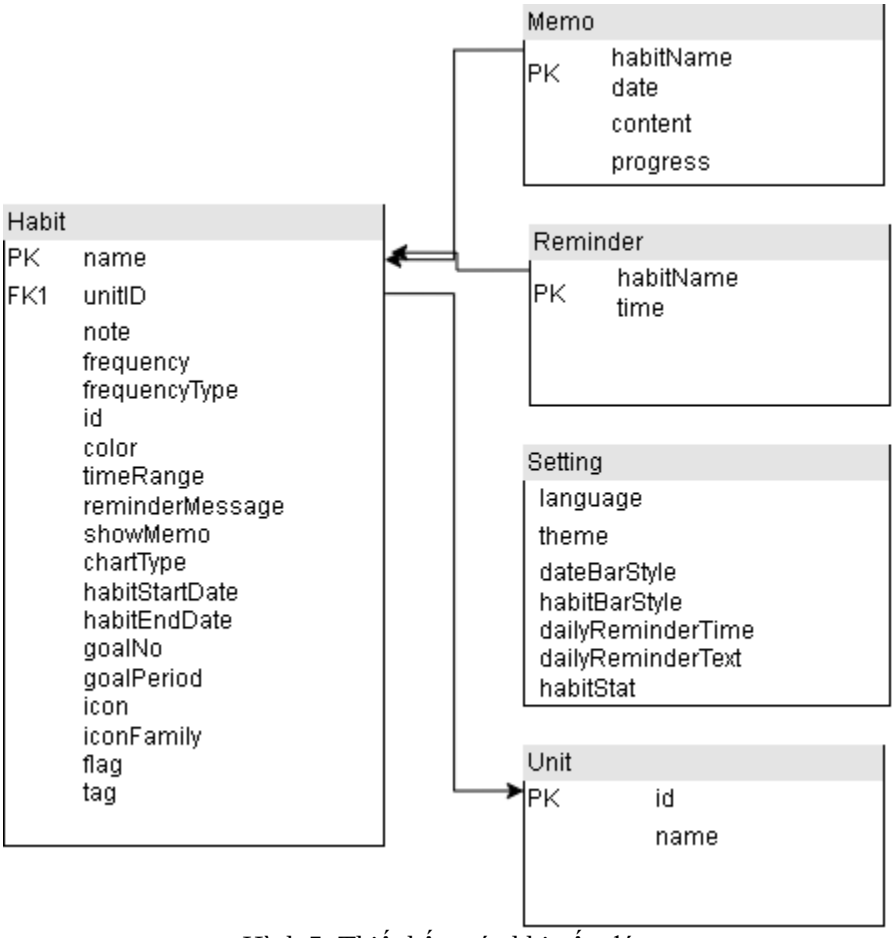
SETTING

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Mô tả
Key	TEXT		Khóa cho các trường setting
Value	TEXT		Gía trị cho các trường setting

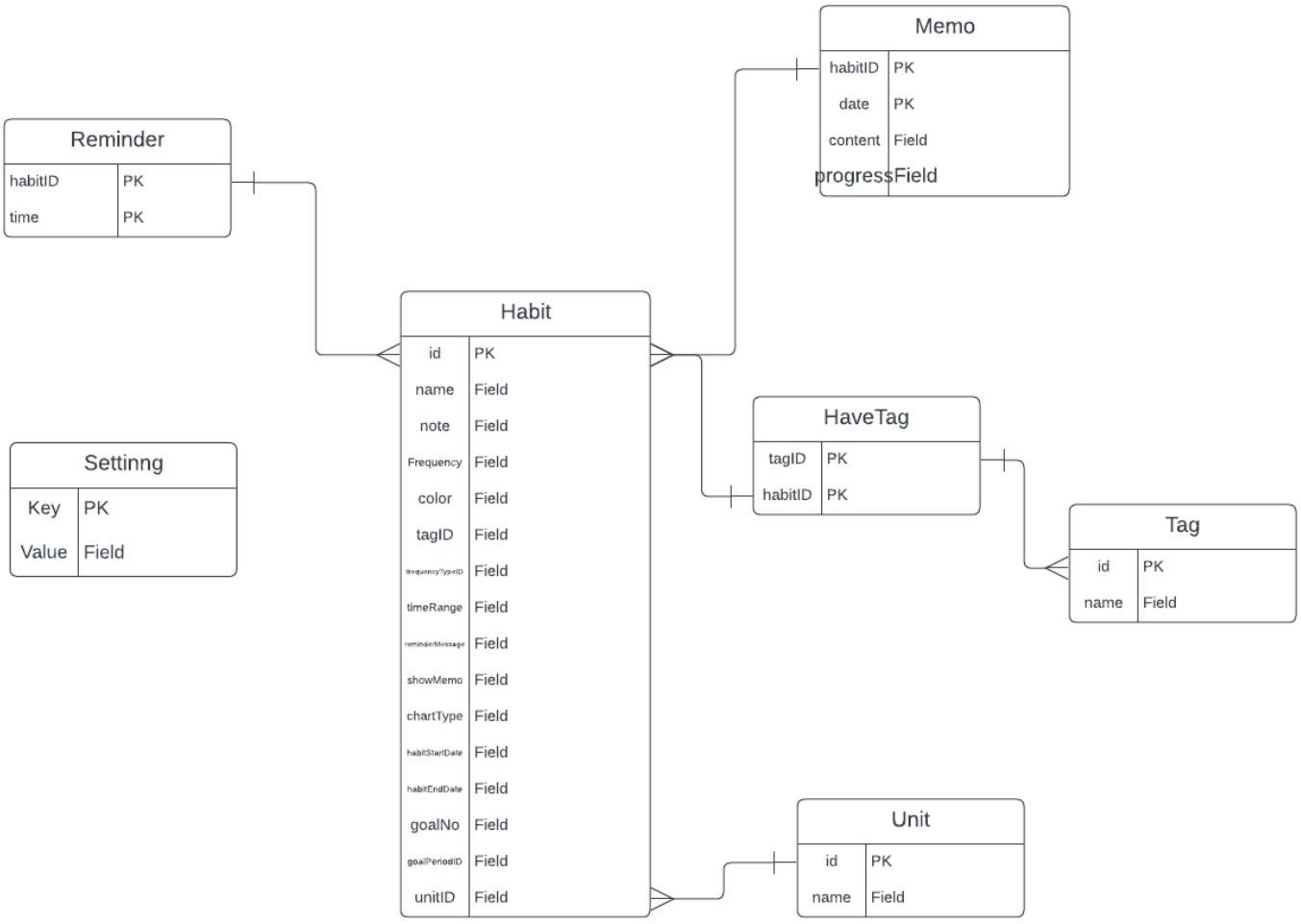
Sơ đồ thực thể



Hình 4. Thiết kế ban đầu



Hình 5. Thiết kế trước khi vấn đáp



Hình 6. Thiết kế sau khi vấn đáp

Chi tiết cách vận hành và tương tác với người dùng của các giao diện chức năng chính

Các chức năng khó khăn về mặt kỹ thuật lập trình:

Màn hình:

- Màn hình add habit (2):

  Custom lại countdown clock cho phù hợp với phiên bản react hiện tại

  Thanh progress bar để thể hiện tiến độ hoàn thành ở ngày được chọn

  Custom thanh tab bar khi ở Home sẽ là dấu "+" để thêm habit, khi qua các screen khác sẽ là icon Home để quay về màn hình chính

  Custom thanh progress hình tròn trong chi tiết của habit

  Ở màn hình add habit nếu đã chọn habit "A" rồi sẽ không cho chọn lại

  Không cho user điền progress vào ngày ở tương lai

Hiện habit theo ngày được chọn trong tuần

- Màn hình statistic (3):

  Custom progress circle chart thể hiện phần trăm hoàn thành của tháng hiện tại

  Xử lý database và global state để hiển thị các thông số tính toán

  Xử lý onPress cho các icon tương ứng với mỗi habit khi chọn sẽ dẫn đến thống kê tương ứng

  Màn hình con chi tiết của mỗi habit được chọn:

    Custom lịch để thể hiện mức độ hoàn thành của mỗi habit ở ngày có habit đó trong tháng hiện tại

    Xử lý database và global state để hiển thị các thông số tính toán

    Custom bar chart để hiển thị thông số đã hoàn thành habit trong tuần hiện tại

    Xử lý cho edit và delete habit

- Màn hình setting (4):

  Xử lý hiển thị modal class cho các chức năng

  Màn hình con của Theme:

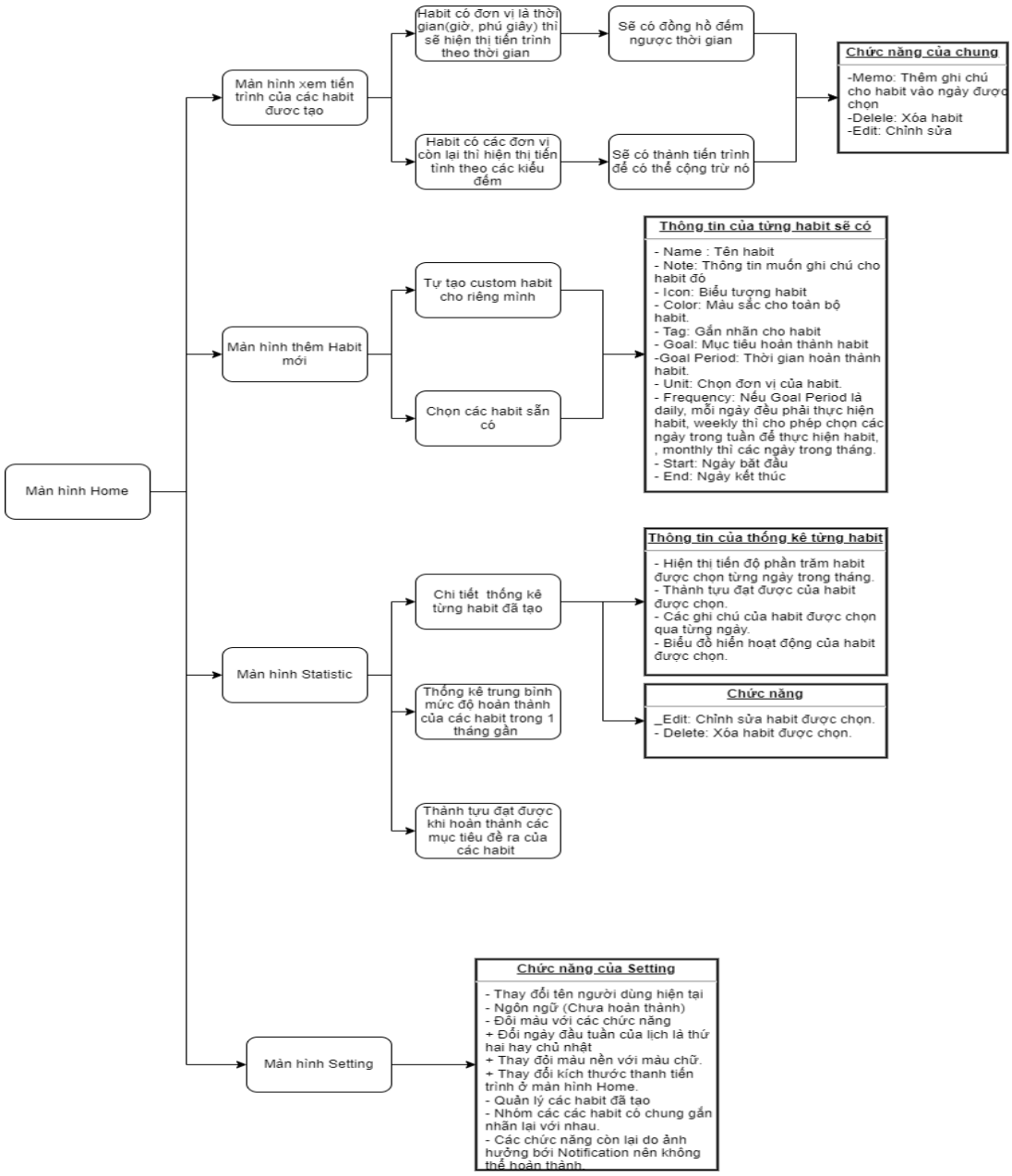
    Đổi ngày bắt đầu của một tuần: thứ 2/ CN

    Xử lý dark theme đổi kiểu tối toàn bộ app

    Kích thước thanh progress bar ở màn hình Home

  Xử lý Habit Manager: edit hoặc delete

  Xử lý Tag Manager: hiển thị các habit thuộc từng tag cụ thể được chọn



Hình 6. Sơ đồ tương tác với người dùng của các giao diện chức năng chính

CHƯƠNG 3: CÁC VẤN ĐỀ KỸ THUẬT VÀ ỨNG DỤNG

VẤN ĐỀ	HƯỚNG GIẢI QUYẾT
Các hàm trong một screen render lại nhiều lần và có dấu hiệu lặp vô tận ➔ Gây lag đơ app trong trường hợp với các màn hình có sử dụng nhiều câu query ➔ đơ lag	Sử dụng useEffect để hạn chế việc lặp vô tận. Kiểm tra lại tất cả các function, screen, các params, câu query được truyền qua các screen, để các thành phần đó vào useEffect

```
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
LOG  listHaveTag []
LOG  listTag []
```

```
useEffect(() => {
  calculateDayTotalDone(state.listHabit, state, dispatch);
  CountPerfectDay(state.listHabit, state, dispatch);
  CountPerfectStreak(state.listHabit, state, dispatch);
  CalculateOverallRate(state.listHabit, state, dispatch);
  CalculateDailyAverage(state, dispatch);
}, []); // 👈 empty dependencies array
```

Icon được sử dụng đa dạng do thư viện icon do react hỗ trợ rất da dạng, dẫn đến việc ở mỗi màn hình nếu có dùng icon thuộc các thư viện khác nhau phải import nhiều lần. Không sử dụng được ***map*** cho các icon này vì đòi hỏi thẻ khác nhau.

```
import Feather from 'react-native-vector-icons/Feather';
import Ionicons from 'react-native-vector-icons/Ionicons';
import ZocialIcon from 'react-native-vector-icons/Zocial';
import EntypoIcon from 'react-native-vector-icons/Entypo';
import Fontisto from 'react-native-vector-icons/Fontisto';
import EvilIcon from 'react-native-vector-icons/EvilIcons';
import FontAwesome from 'react-native-vector-icons/FontAwesome';
import AntDesign from 'react-native-vector-icons/AntDesign';
import OcticonIcon from 'react-native-vector-icons/Octicons';
import FontAwesome5 from 'react-native-vector-icons/FontAwesome5';
import FoundationIcon from 'react-native-vector-icons/Foundation';
import MaterialIcons from 'react-native-vector-icons/MaterialIcons';
import SimpleLineIcon from 'react-native-vector-icons/SimpleLineIcons';
import MaterialCommunityIcons from 'react-native-vector-icons/MaterialCommunity
```

Tự custom thẻ Icons, trong này chứa tất cả thư viện icon mà react hỗ trợ. Dùng các biến Icon Family, Name, color để lưu các thuộc tính để truyền vào thẻ Icons từ đó trả về icon mà mình mong muốn.

```
const Icons = ({type, ...props}) => {
  const getIcon = type => {
```

```
const FontIcon = getIcon(type);
    return <FontIcon {...props} />;
};
export default Icons;
```

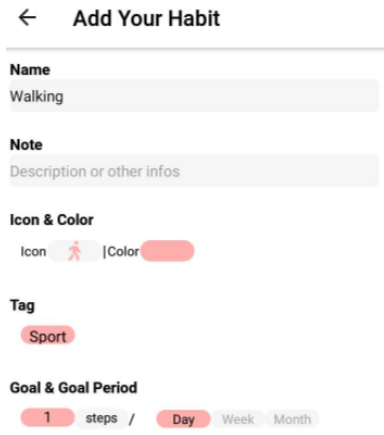
User chỉnh sửa progress của habit ở ngày trong tương lai dẫn đến lỗi ở các câu query trong phần statistic cụ thể đây là màn hình thống kê trong đó có các câu thống kê theo tháng hiện tại

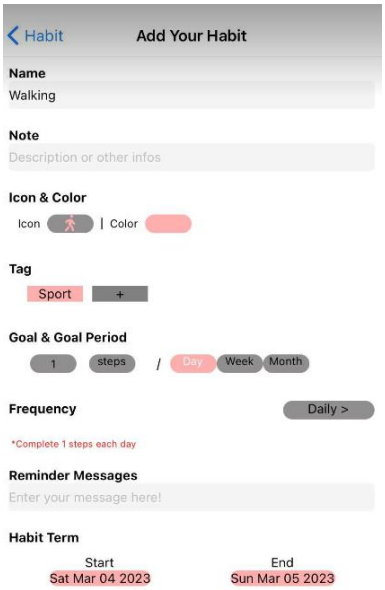
Hạn chế quyền user, không cho edit habit nếu ngày hiện tại < ngày được chọn

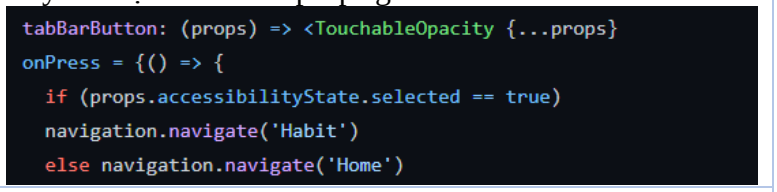
```
Alert.alert(
  'Warning',
  'You can not check for future date',
  [
    {text: 'OK', onPress: () => {navigation.goBack()}},
  ],
  {cancelable: false},
```

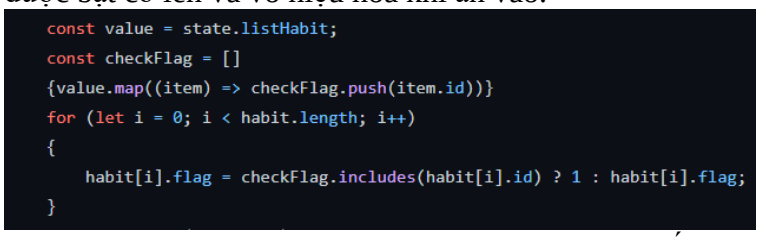
`` getProgressCurMonth(habit, state, dispatch) getDataOfCurWeek(habit, state, dispatch); calculateDayDoneInMonth(habit, state, dispatch); calculateDayTotalDone(habit, state, dispatch); calculateMonthlyVolumn(habit, state, dispatch); calculateTotalVolumn(habit, state, dispatch); calculateCurrentStreak(habit, state, dispatch); calculateBestStreak(habit, state, dispatch); ``` Các hàm thực hiện query này sẽ cho ra kết quả thống kê sai lệch	
Data không thể sử dụng ở tất cả các screen mà phải truyền qua route.params trung gian nhiều lần, database có nhiều bảng, thông tin được load lên cần tái sử dụng nhiều lần nên nếu khi nào dùng thì sử dụng query sẽ rất mất thời gian	Sử dụng useContext, useReducer và Provider để có thể sử dụng các biến global từ đó các screen có thể truy cập trực tiếp và tương tác không phải truyền qua trung gian nhiều lần ``` 439 440 export {globalState} 441 export default reducer; ``` Tham khảo nguồn sử dụng reducer cho javascript trên course của F8 ở youtube và thử nghiệm và sửa nhiều lần
Chưa tìm ra cách hiển thị được dữ liệu được khởi tạo trong thiết bị một các trực quan như trong SQL server, không kiểm soát được dữ liệu đã được lưu, nên mỗi khi git clone từ github về thường xuyên bị conflict với dữ liệu còn lưu lại trong máy hoặc chồng chéo data, gây ra nhiều khó khăn trong việc test các tính năng	Sau nhiều lần thử và để đảm bảo tính ổn định cho app tạo một file initdata.js để chứa dữ liệu, hàm refreshdata và initdata để có thể xóa dtb cũ và tạo database mới. Giúp kiểm soát được dữ liệu. ``` JS init_data.js × Store > JS init_data.js > habitInit 1 export const habitInit = "INSERT 2 chartType, habitStartDate, ha 3 ('Dancing', 'Nothing to note' 4 '30', 'Week', '2', 'walking', 5 ('Meditate', 'Nothing to note 6 '10', 'Day', '2', 'meditation 7 ('Run', 'Nothing to note', '2 8 '2', 'Day', '9', 'running', ' 9 10 export const memoInit = "INSERT I ```
Có một số operator SQLite không hỗ trợ được so với SQL bình thường như operator IN, … ``` const FindWeeklyAndDailyHabit = () => { db.transaction(tx => { tx.executeSql('SELECT * \ FROM Habit\ WHERE frequencyType IN ("Weekly", "Daily")', [habit.name], (txObj, resultSet) => { console.log("Resultset: ", resultSet.rows._array) }, (txObj, error) => console.log(error)); }) } ```	Tìm kiếm và tự viết lại các function theo Js sao cho phù hợp và thỏa với nhu cầu tìm kiếm ``` const FindWeeklyAndDailyHabit = () => { db.transaction(tx => { tx.executeSql('SELECT * \ FROM Habit', [habit.name], (txObj, resultSet) => { let res = [] for (let item in resultSet.rows._array) if (item.frequencyType == 'Daily' || frequencyType == 'Weekly') res.push(item) console.log("Resultset: ", res) }, (txObj, error) => console.log(error)); }) } ```

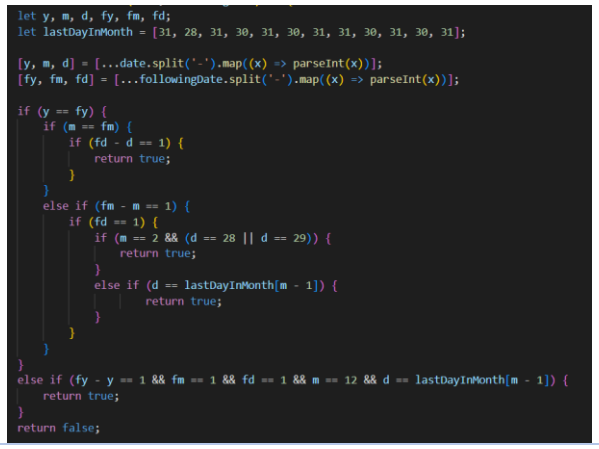
Khi sử dụng các thư viện build sẵn,một số thư viện cũ chưa được cập nhật ứng với bản react-native hiện tại hoặc một số thư viện lại cài một thư viện khác với phiên bản hiện tại mà thư viện ta đang dùng gây ra đụng độ. Cụ thể ở đây là 'react-native-countdown' thì nó có những cú pháp mà phiên bản react hiện tại tụi em dùng không còn hỗ trợ nữa (AppState.removeEventListener) `componentWillUnmount() { clearInterval(this.timer); AppState.removeEventListener('change', this._handleAppStateChange); }`	Tham khảo từ nhiều nguồn khác nhau thì cần vào tận thư viện gốc sửa, xóa hoặc thay đổi câu lệnh phù hợp với phiên bản hiện react-native hiện tại. `componentDidMount() { this.appStateSubscription = AppState.addEventListener('change', this._handleAppStateChange); }` `componentWillUnmount() { clearInterval(this.timer); this.appStateSubscription.remove(); }`
Lỗi RTC View do chưa có dữ từ database lên lên để truyền vào biến để tính toán và hiển thị ra màn hình. Từ đó không sử dụng được progressBar công cụ thể hiện mức độ hoàn thành của một habit nào đó `<TouchableOpacity style={{ padding: 5 }} key={value.id} onPress={() => navigation.navigate('HabitDetail', {habit: value, checkShow: checkShow})}> <Progress.Bar progress={handleMath()} width={null} height={state.habitBarSize?50:35} color={value.color}> {/* (value.progress)/(value.goalNo) */} </Progress.Bar> <View style={{ flex: 1, position: 'absolute', flexDirection: 'row', alignItems: 'center', top:9, left:10 }}>`	Có 2 trường hợp xảy ra hoặc là habit ngày hôm đó chưa có dữ liệu thì sẽ truyền cho habit đó một số dữ liệu init để đảm bảo có thể thực hiện tính toán bình thường. Một trường hợp khác là nếu database quá lớn, nhiều trường dữ liệu thì sẽ mất tương đối thời gian để có thể load hoàn chỉnh ➔ để giải quyết thì tạo 1 màn hình chờ khác trước khi vào thẳng home
Mong muốn khi chọn để popup ra một modal ngay dưới vị trí bấm thì nó lại xuất hiện ở giữa màn hình.	Sử dụng thư viện react-native-select-dropdown để modal có thể xuất hiện theo đúng như ý muốn `<SelectDropdown data={option} // defaultButtonText={<View style={{AlignItems:'center'}}> // <MaterialCommunityIcons name="dots-horizontal-circle-outline // </View> // } defaultButtonText={<Text style={{alignSelf:'flex-start',color: currentTh buttonStyle={{ width: 50, height: 50, borderRadius: 25, backgroundColor: 'gray'}} //style={{ width: 100, height: 100}} dropdownStyle={{ marginLeft:'-13%', width: 180, height: 100, backgroundColor: currentTheme.backgroundCol }} rowTextStyle={{ color: currentTheme.color }} onSelect={(selectedItem, index) => { if (selectedItem == option[0]) navigation .navigate('EditHabit', { Habit: habit, }) else if (selectedItem == option[1]) { handleDelHablit(habit.id, habit.name) } }}`
Lập trình chạy giao diện trên expo-ios nó hiển thị khác trên expo-android. Vì trong nhóm có 2 thành viên dùng android, 2 bạn còn lại dùng ios gây ra	Thay đổi giao diện dựa trên 1 trong 2 hệ điều hành điện thoại, nhóm phải nhiều lần họp thảo luận và thống nhất khi code nên tốn nhiều thời gian công sức, phải fix các lỗi

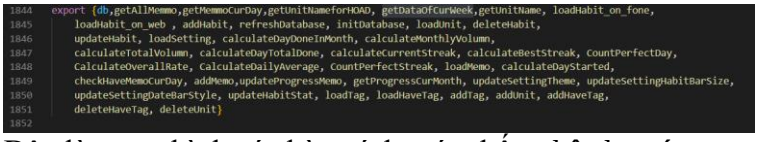
<table>
<tr>
<td>nhiều khó khăn trong quá trình thiết kế giao diện cũng như sử dụng các thư viện hay dùng các function. VD:
 

Đây là giao diện với android các button được thiết kế tròn bo góc
 

Đây là giao diện với IOS các button được thiết kế hoàn toàn khác so với ý định bên android</td>
<td>để các bạn trong nhóm có thể cùng nhau thực hiện nhằm đảm bảo tiến độ app</td>
</tr>
<tr>
<td>Thiết kế navigation giữa các screen sử dụng TabBar Navigation khiến cho việc chuyển trang gặp khó khăn do mặc định TabBar chỉ điều hướng tới một số trang đã thiết lập sẵn.</td>
<td>Custom lại hàm điều hướng trang của TabBar Navigation, qua đó cho phép điều hướng tới trang khác khi Icon của TabBar thay đổi. Cần thêm câu lệnh điều kiện và phải truyền được tất cả các props giữa các screen
 
</td>
</tr>
<tr>
<td>Thiết kế các Button chọn Frequency gặp nhiều khó khăn do số lượng các Button nhiều.</td>
<td>Dùng mảng lưu giá trị của các Button, sau đó dùng hàm map để ánh xạ từng giá trị có trong mảng ra 1 Button cụ thể.</td>
</tr>
</table>

	 Bên cạnh đó phải đảm bảo được tính logic ví dụ khi user chọn day thì mặc định là daily, chọn week thì sẽ có thể chọn cố định một số ngày trong tuần, chọn month thì có thể chọn những ngày cố định trong tháng
Khi cài thư viện được xây dựng sẵn mà trong đó nó có cài thêm một thư viện với biên bản thấp hơn phiên bản mà thư viện đó ta đã cài gây xung đột phiên bản với nhau cụ thể thư viện lỗi là @react-native-community/datetimepicker	Một là xóa cái thư viện ta đã cài với phiên bản cao nhất hoặc là vào thẳng thư viện có cài thư viện đó và nâng cấp nó cho bằng phiên bản ta đã cài. Xung đột vs cái datetime của cái thư viện react-native-calendars nhưng lỗi này chỉ xuất hiện bên IOS, android thì không nhưng do nhóm có 2 bạn sử dụng IOS nên cũng cần khắc phục Theo hướng dẫn từ một số bài trên Github thì sẽ xóa cái file datetime bên thư viện react-native-calendars ra khỏi node module
Thiết kế các Button trong Habit sao cho Habit đã chọn rồi thì sẽ đánh đánh dấu tích xanh.	Đặt thêm biến flag cho mỗi Habit, Habit đã được chọn sẽ được bật cờ lên và vô hiệu hóa khi ấn vào.  Tạo mảng check flag và cũng phải lưu mảng này xuống database để đảm bảo những lần tiếp theo khi mở app lên vẫn đảm bảo được những ràng buộc như mong muốn
Xây dựng mà hình Home hiện thị các habit đã tạo theo ngày và tag mà người dùng muốn hiển thị	Phải tạo ra 4 bảng: 1 bảng habit lưu lại ngày bắt đầu kết thúc, ngày muốn hiện thị nó theo hằng ngày hay thứ trong tuần hoặc ngày trong tháng 1 bảng progress dữ trên dữ liệu của của habit sẽ lưu lại những ngày ứng vs nhưng thông tin trên của 1 habit 1 bảng tag lưu lại những tag đã tạo 1 bảng haveTag là bảng trung gian để kết hợp tag vs bảng habitSau đó muốn hiện thị ra home theo cái yêu cầu trên thì ta phải filter từ 4 bảng trên

Dữ liệu lưu dưới database được lưu dưới dạng chuỗi, do đó không thể xử lý như một mảng thông thường. Khi cần truy xuất dữ liệu phải xử lí chuỗi sao cho lấy đúng thông tin cần lấy ví dụ định dạng ngày tháng,..	Phải xử lý chuỗi được lấy lên từ database thông qua các hàm như strip, include… để có dữ liệu dùng trong các màn hình. 
Thanh progress bar là một trong những tính năng nổi bật của app tuy nhiên thư viện hỗ trợ rất ít hoặc không đầy đủ, cần phải custom lại theo đúng yêu cầu của app đồng thời thể hiện được đúng theo những ý nghĩa như app gốc	Sử dụng thư viện react-native-progress và custom với thông số tùy chọn
Custom progress circle chart để hiển thị phần trăm hiện tại đã hoàn thành trong tháng, dùng các câu query để lấy thông tin lên truyền vào reducer từ đó custom data và chartConfig cho các biểu đồ	Dùng react-native-chart-kit và custom các thông số về data và chart config, các câu lệnh sqlite cần chỉnh sửa nhiều lần và chắt lọc những thông tin cần thiết  

Custom lịch để thể hiện mức độ hoàn thành của mỗi habit ở ngày có habit đó trong tháng hiện tại 1 cách trực quan nhất	Xây dựng function CustomCalendar với thẻ Calendar được truyền các tham số thích hợp như markingType, markingDates  const getMarked = () => { let marked = {}; if (state.listProDate.length != 0) for(let i = 0; i< state.listPro.length; i++){ let temp = state.listPro[i]/habit.goalNo; // console.log(typeof state.listProDate[i]) let tempDate = state.listProDate[i].split('/').join('-'); if (temp == 0) { marked[tempDate] = { dots: [none] }; } else if (temp <= 0.5) { marked[tempDate] = { dots: [half] }; } else if (temp <= .75) { marked[tempDate] = { dots: [near] }; } else if (temp <= 1) { marked[tempDate] = { dots: [done] }; }
Ở màn hình statistic phải thực hiện rất nhiều thống kê về tổng quan các habit như tính tổng số ngày, tính chuỗi, rate đã hoàn thành, mức độ trung bình 1 ngày,.. với mỗi habit cụ thể cũng phải thực hiện nhiều câu query truy xuất thông tin trung bình với mỗi habit cần 8 – 10 câu	Phải viết nhiều câu query khác nhau, xử lí với dữ liệu tương đối khó khăn vì đây là những habit khi chỉnh sửa sẽ có ảnh hưởng không chỉ ngày hiện tại mà còn ảnh hưởng đến dữ liệu lưu trong quá khứ hoặc tương lai do có các funtion thống kê tính toán nên phải đảm bảo tính logic cực kì khó khăn  export {db,getAllMemmo,getMemmoCurDay,getUnitNameforHOAD, getDataOfCurWeek,getUnitName, loadHabit_on_fone, loadHabit_on_web , addHabit, refreshDatabase, initDatabase, loadUnit, deleteHabit, updateHabit, loadSetting, calculateDayDoneInMonth, calculateMonthlyVolumn, calculateTotalVolumn, calculateDayTotalDone, calculateCurrentStreak, calculateBestStreak, CountPerfectDay, CalculateOverallRate, CalculateDailyAverage, CountPerfectStreak, loadMemo, calculateDayStarted, checkHaveMemoCurDay, addMemo,updateProgressMemo, getProgressCurMonth, updateSettingTheme, updateSettingHabitBarSize, updateSettingDateBarstyle, updateHabitStat, loadTag, loadHaveTag, addTag, addUnit, addHaveTag, deleteHaveTag, deleteUnit} Đây là trung bình các hàm tính toán thống kê cho các habit đã được viết

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

CHỨC NĂNG CHƯA HOÀN THIỆN	HƯỚNG GIẢI QUYẾT
Chức năng Reminder, notification, Vacation Mode, Sound, chưa làm được	Chuyển sang CSDL mới Firebase
Countdown clock không thể reset được	Tìm hiểu cách sử dụng useRef trong node modules
Chỉ mới hỗ trợ biểu đồ thể hiện quá trình là biểu đồ cột	Thêm các dạng biểu đồ khác (đường) để user có thể dễ quan sát
Vòng tròn thể thể progress của một ngày cụ thể đã thực hiện được bao nhiêu phần trăm trên calendar	Chỉ mới là các dấu chấm thể hiện tương đối. Tìm hiểu thêm các thư viện hỗ trợ
Thay đổi ngôn ngữ	Thiết kế các screen mới tương đương phù hợp với ngôn ngữ được chọn
Custom cho thanh progress bar ở màn hình home	Cho thêm các chức năng để user tự custom thanh bar tùy thích

Khi hoàn thành 1 ngày chỉ tiêu, thì hiện thông báo memo khi được chọn	Thêm hiệu ứng cho screen habit detail
Custom vòng tròn progress trên Calendar	Tìm hiểu thêm về thư viện Calendar để custom tương ứng

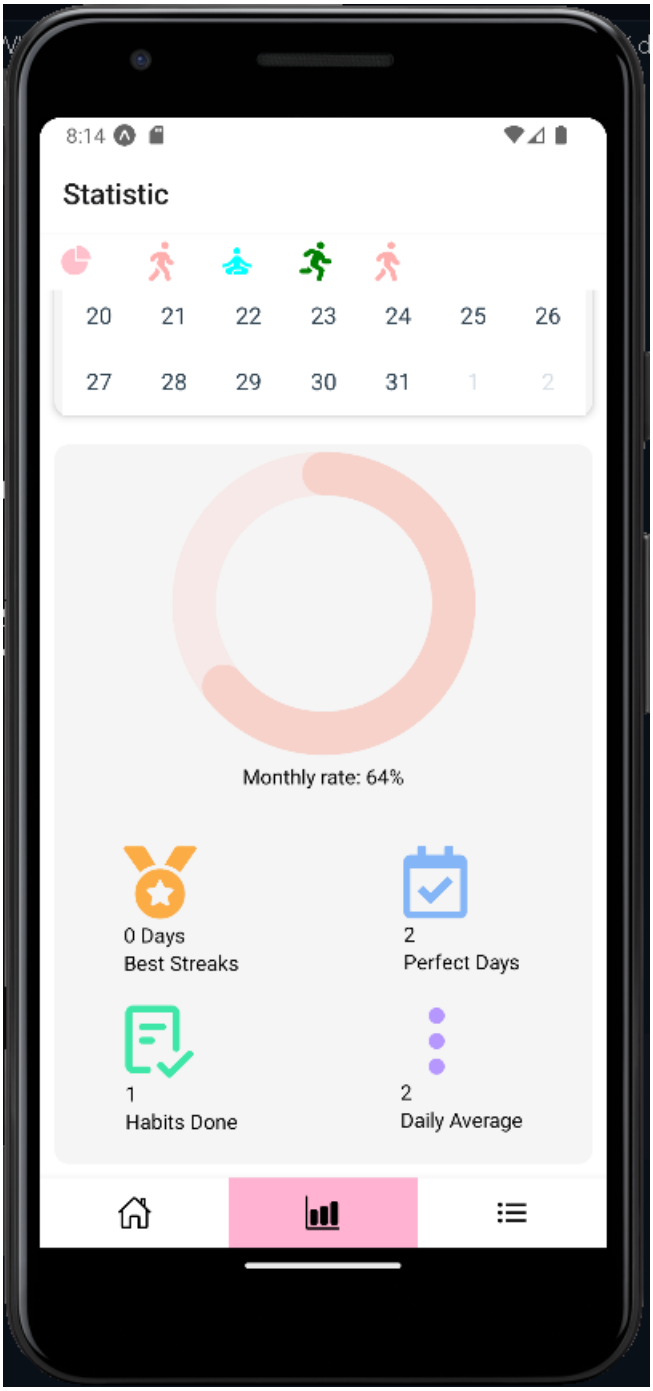
CHƯƠNG 5: BỔ SUNG SAU VẤN ĐÁP

Bổ sung lại chương 1 (Phần so sánh với app trên store)

Sửa lại bảng setting từ nhiều cột 1 hàng sang 2 cột key, value (tại đây)

Bổ sung lại chương 3 (Bổ sung chi tiết hơn, thêm cách dẫn chứng cụ thể)

Bổ sung giải thích phần thông kê:



Monthly rate: hàm trong database calculateOverallRate

Bước 1: Lấy data của tháng cần xét từ database Group theo ngày, Count(*)

Bước 2: Khởi tạo biến rates để lưu lại các tiến độ hàng ngày

Bước 3: Với từng này kiểm tra xem ngày đó có habit nào không

- Nếu không thì cho ngày đó hoàn thành 100%, thêm vào rates
- Nếu có thì tính tiến độ của ngày đó, Nếu ngày đó có 5 hoạt động mà hoàn thành được 3 hoạt động thì rate của ngày đó sẽ bằng 60%

Bước 4: Lấy trung bình các giá trị của rate lại. Trả về giá trị đó

Best streak: hàm trong dtb CountPerfectStreak

Bước 1: Lấy data của tháng cần xét từ database Group theo ngày, Count(*)

Bước 2: Khởi tạo biến bestStreak, count, date, followingDate

Bước 3: Với từng ngày, So sánh số lượng habit làm được trong ngày đó với số habit có trong ngày đó

Nếu bằng nhau thì chúng ta sẽ xét xem chuỗi ngày có bị gãy không bằng cách xem xét date và followingDate

- Nếu chuỗi được giữ thì count += 1, so sánh với bestStreak. Nếu BestStreak nhỏ hơn count thì gán count cho BestStreak.
- Nếu không thì count = 1

Bước 4: Trả về giá trị BestStreak

Perfect day: hàm trong database.js CountPerfectDay

Bước 1: Lấy data của tháng cần xét từ database Group theo ngày, Count(*)

Bước 2: Khởi tạo biến count

Bước 3: Với từng ngày, So sánh số lượng habit làm được trong ngày đó với số habit có trong ngày đó

Nếu bằng nhau thì count += 1

Bước 4: Trả về giá trị count

Habit done: hàm trong database.js CountDayTotalDone

Bước 1: Đếm những habit đã hoàn thành rồi trong database

Bước 2: Trả về giá trị đếm được

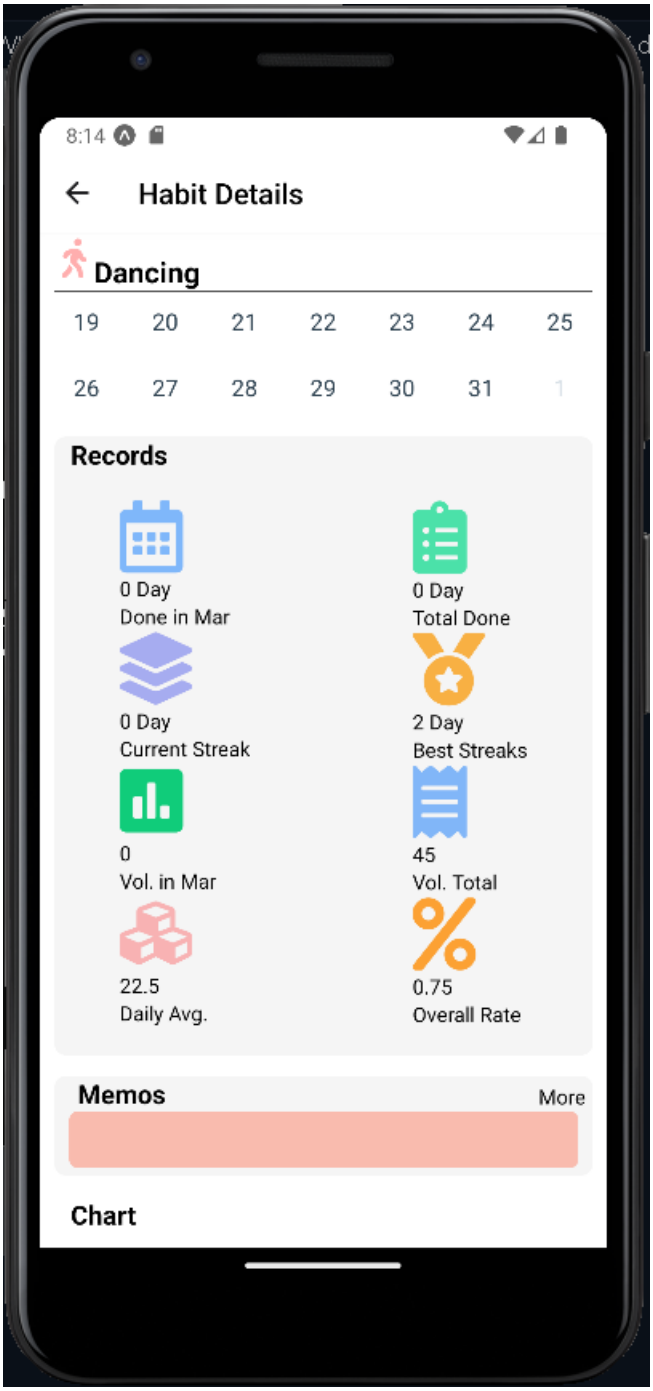
DailyAverage: hàm trong database.js tên là CalculateDailyAverage

Bước 1: Lấy data của tháng cần xét từ database Group theo ngày, Count(*)

Bước 2: Khởi tạo biến counts

Bước 3: Với từng ngày, Đếm số lượng habit thục hiện, thêm vào biến counts

Bước 4: Tính trung bình biến counts, trả về giá trị đó.



Done in <Month>: hàm trong database.js là CalculateDayDoneInMonth

Bước 1: Lấy data của habit cần xét trong tháng hiện tại từ database Group theo ngày, Count(*)

Bước 2: Trả về COUNT(*)

Day Total Done: hàm trong database.js CountDayTotalDone

Bước 1: Tìm kiếm ngày hoàn thành habit đó trong database và đếm chúng.

Bước 2: Trả về giá trị đếm được

Current Streak: Hàm trong database.js là CalculateCurrentStreak và CurrentStreak

Bước 1: Chọn những ngày có thực hiện trong Memo của habit cần thống kê, sắp xếp theo thứ tự giảm dần

Bước 2: Nếu kết quả là rỗng thì trả về 0

Bước 3: Nếu ngày gần nhất khác ngày hôm nay thì trả về 0

Bước 4: Khởi tạo count = 1

Bước 5: Xét những ngày trước đó, nếu bị gãy chuỗi thì trả về count ngay

- Nếu không thì count += 1

Bước 6: Trả về giá trị count

Best Streak: Hàm trong database.js là CalculateBestStreak và BestStreak

Bước 1: Chọn những ngày có thực hiện trong Memo của habit cần thống kê, sắp xếp theo thứ tự giảm dần

Bước 2: Nếu kết quả là rỗng thì trả về 0

Bước 3: Khởi tạo biến bestStreak, count,

Bước 4: Với từng ngày, khởi tạo date, followingDate. Xem xét cả hai biến đó,

- Nếu chuỗi không gãy thì count += 1, so sánh với bestStreak. Nếu BestStreak nhỏ hơn count thì gán count cho BestStreak.
- Nếu không thì count = 1

Bước 5: Trả về bestStreak

Vol in <Month>: hàm trong database.js là CalculateMonthlyVolumn

Bước 1: Lấy data của habit cần xét trong tháng hiện tại từ database Group theo ngày, Sum(progress)

Bước 2: Trả về Sum(progress)

Vol in Total: hàm trong database.js là CalculateTotalVolumn

Bước 1: Lấy data của habit cần xét từ database Group theo ngày, Sum(progress)

Bước 2: Trả về Sum(progress)

Daily Average: hàm trong database.js là CalculateDayStarted

Bước 1: Đếm số ngày thực hiện habit cần xét trong database

Bước 2: Lấy TotalVolumn chia cho số ngày đếm được

Bước 3: Trả về giá trị đã tính

Overall rate:

Bước 1: Lấy Daily Average chia cho goalNo của habit đó và trả về kết quả