GitHub Link: <https://github.com/VirtuFit/virtufitapp>

Note1: The same read me is in github.

Note2: After installing the app we will explain how the users should use the app.

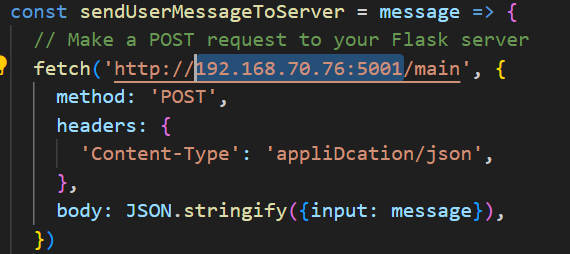
# How to Install App:

Step 1: Install the two python files for chatbots from repositry: <https://github.com/VirtuFit/VirtuFit-Chatbot.git>

Step 2: Add the api keys, as they can't be uploaded to github.

Step 3: To connect flask, download flask and the required packages from pip install. After installing everything, open the the chatbot files from cmd using 'python '. After opening them. Change the address from Chatbot.js and StartUpChatbot.js to match your ip address that is found in the cmd that runs flask.

Example(image):

[](https://private-user-images.githubusercontent.com/153627116/289927973-0da94e5a-9eb3-487d-9276-acb87e1d8b96.png?jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoicmF3LmdpdGh1YnVzZXJjb250ZW50LmNvbSIsImtleSI6ImtleTEiLCJleHAiOjE3MDI0NTM0NTQsIm5iZiI6MTcwMjQ1MzE1NCwicGF0aCI6Ii8xNTM2MjcxMTYvMjg5OTI3OTczLTBkYTk0ZTVhLTllYjMtNDg3ZC05Mjc2LWFjYjg3ZTFkOGI5Ni5wbmc_WC1BbXotQWxnb3JpdGhtPUFXUzQtSE1BQy1TSEEyNTYmWC1BbXotQ3JlZGVudGlhbD1BS0lBSVdOSllBWDRDU1ZFSDUzQSUyRjIwMjMxMjEzJTJGdXMtZWFzdC0xJTJGczMlMkZhd3M0X3JlcXVlc3QmWC1BbXotRGF0ZT0yMDIzMTIxM1QwNzM5MTRaJlgtQW16LUV4cGlyZXM9MzAwJlgtQW16LVNpZ25hdHVyZT03MDMyNzQzZGRlYWI5MGEwMjY3Y2I3MzIyOTYzZDNmNTNmYmJmNmEyYjJhYzgwZDdiNjQxZjgwZTlkYzljNjdhJlgtQW16LVNpZ25lZEhlYWRlcnM9aG9zdCZhY3Rvcl9pZD0wJmtleV9pZD0wJnJlcG9faWQ9MCJ9.Z32UwXl_PHdzS_9btvKQKLPcNoksX-UQRFmX22xFT6A)

Step 4: clone the current repositry, which is for running the react native application.

Step 5: Open the clone of the current repositry in visual studio code.

Step 6: Open a new terminal and write npm install.

Step 7: After running npm install and getting node\_modules folder, go to node\_modules/@react-navigation/drawer/src/views/DrawerView.tsx and change it as its outdated and causes a lot of issues.

new DrawerView.tsx:

import { getHeaderTitle, Header, SafeAreaProviderCompat, Screen, } from '@react-navigation/elements'; import { DrawerActions, DrawerNavigationState, DrawerStatus, ParamListBase, useTheme, } from '@react-navigation/native'; import \* as React from 'react'; import { I18nManager, Platform, StyleSheet, View } from 'react-native'; import \* as Reanimated from 'react-native-reanimated'; import { useSafeAreaFrame } from 'react-native-safe-area-context';

import type { DrawerContentComponentProps, DrawerDescriptorMap, DrawerHeaderProps, DrawerNavigationConfig, DrawerNavigationHelpers, DrawerNavigationProp, DrawerProps, } from '../types'; import { addCancelListener } from '../utils/addCancelListener'; import DrawerPositionContext from '../utils/DrawerPositionContext'; import DrawerStatusContext from '../utils/DrawerStatusContext'; import getDrawerStatusFromState from '../utils/getDrawerStatusFromState'; import DrawerContent from './DrawerContent'; import DrawerToggleButton from './DrawerToggleButton'; import { GestureHandlerRootView } from './GestureHandler'; import { MaybeScreen, MaybeScreenContainer } from './ScreenFallback'; import Drawer from './modern/Drawer';

type Props = DrawerNavigationConfig & { defaultStatus: DrawerStatus; state: DrawerNavigationState; navigation: DrawerNavigationHelpers; descriptors: DrawerDescriptorMap; };

const getDefaultDrawerWidth = ({ height, width, }: { height: number; width: number; }) => { const smallerAxisSize = Math.min(height, width); const isLandscape = width > height; const isTablet = smallerAxisSize >= 600; const appBarHeight = Platform.OS === 'ios' ? (isLandscape ? 32 : 44) : 56; const maxWidth = isTablet ? 320 : 280;

return Math.min(smallerAxisSize - appBarHeight, maxWidth); };

const GestureHandlerWrapper = GestureHandlerRootView ?? View;

function DrawerViewBase({ state, navigation, descriptors, defaultStatus, drawerContent = (props: DrawerContentComponentProps) => ( <DrawerContent {...props} /> ), detachInactiveScreens = Platform.OS === 'web' || Platform.OS === 'android' || Platform.OS === 'ios', }: Props) { const legacyImplemenationNotAvailable = require('react-native-reanimated').abs === undefined;

const focusedRouteKey = state.routes[state.index].key; const { drawerHideStatusBarOnOpen = false, drawerPosition = I18nManager.getConstants().isRTL ? 'right' : 'left', drawerStatusBarAnimation = 'slide', drawerStyle, drawerType = Platform.select({ ios: 'slide', default: 'front' }), gestureHandlerProps, keyboardDismissMode = 'on-drag', overlayColor = 'rgba(0, 0, 0, 0.5)', swipeEdgeWidth = 32, swipeEnabled = Platform.OS !== 'web' && Platform.OS !== 'windows' && Platform.OS !== 'macos', swipeMinDistance = 60, overlayAccessibilityLabel, } = descriptors[focusedRouteKey].options;

const [loaded, setLoaded] = React.useState([focusedRouteKey]);

if (!loaded.includes(focusedRouteKey)) { setLoaded([...loaded, focusedRouteKey]); }

const dimensions = useSafeAreaFrame();

const { colors } = useTheme();

const drawerStatus = getDrawerStatusFromState(state);

const handleDrawerOpen = React.useCallback(() => { navigation.dispatch({ ...DrawerActions.openDrawer(), target: state.key, }); }, [navigation, state.key]);

const handleDrawerClose = React.useCallback(() => { navigation.dispatch({ ...DrawerActions.closeDrawer(), target: state.key, }); }, [navigation, state.key]);

React.useEffect(() => { if (drawerStatus === defaultStatus || drawerType === 'permanent') { return; }

const handleHardwareBack = () => {

if (!navigation.isFocused()) {

return false;

}

if (defaultStatus === 'open') {

handleDrawerOpen();

} else {

handleDrawerClose();

}

return true;

};

return addCancelListener(handleHardwareBack);

}, [ defaultStatus, drawerStatus, drawerType, handleDrawerClose, handleDrawerOpen, navigation, ]);

const renderDrawerContent = () => { return ( <DrawerPositionContext.Provider value={drawerPosition}> {drawerContent({ state: state, navigation: navigation, descriptors: descriptors, })} </DrawerPositionContext.Provider> ); };

const renderSceneContent = () => { return ( {state.routes.map((route, index) => { const descriptor = descriptors[route.key]; const { lazy = true, unmountOnBlur } = descriptor.options; const isFocused = state.index === index;

if (unmountOnBlur && !isFocused) {

return null;

}

if (lazy && !loaded.includes(route.key) && !isFocused) {

return null;

}

const {

freezeOnBlur,

header = ({ layout, options }: DrawerHeaderProps) => (

<Header

{...options}

layout={layout}

title={getHeaderTitle(options, route.name)}

headerLeft={

options.headerLeft ??

((props) => <DrawerToggleButton {...props} />)

}

/>

),

headerShown,

headerStatusBarHeight,

headerTransparent,

sceneContainerStyle,

} = descriptor.options;

return (

<MaybeScreen

key={route.key}

style={[StyleSheet.absoluteFill, { zIndex: isFocused ? 0 : -1 }]}

visible={isFocused}

enabled={detachInactiveScreens}

freezeOnBlur={freezeOnBlur}

>

<Screen

focused={isFocused}

route={descriptor.route}

navigation={descriptor.navigation}

headerShown={headerShown}

headerStatusBarHeight={headerStatusBarHeight}

headerTransparent={headerTransparent}

header={header({

layout: dimensions,

route: descriptor.route,

navigation:

descriptor.navigation as DrawerNavigationProp<ParamListBase>,

options: descriptor.options,

})}

style={sceneContainerStyle}

>

{descriptor.render()}

</Screen>

</MaybeScreen>

);

})}

</MaybeScreenContainer>

);

};

return ( <DrawerStatusContext.Provider value={drawerStatus}> <Drawer open={drawerStatus !== 'closed'} onOpen={handleDrawerOpen} onClose={handleDrawerClose} gestureHandlerProps={gestureHandlerProps} swipeEnabled={swipeEnabled} swipeEdgeWidth={swipeEdgeWidth} swipeVelocityThreshold={500} swipeDistanceThreshold={swipeMinDistance} hideStatusBarOnOpen={drawerHideStatusBarOnOpen} statusBarAnimation={drawerStatusBarAnimation} keyboardDismissMode={keyboardDismissMode} drawerType={drawerType} overlayAccessibilityLabel={overlayAccessibilityLabel} drawerPosition={drawerPosition} drawerStyle={[ { width: getDefaultDrawerWidth(dimensions), backgroundColor: colors.card, }, drawerType === 'permanent' && (drawerPosition === 'left' ? { borderRightColor: colors.border, borderRightWidth: StyleSheet.hairlineWidth, } : { borderLeftColor: colors.border, borderLeftWidth: StyleSheet.hairlineWidth, }), drawerStyle, ]} overlayStyle={{ backgroundColor: overlayColor }} renderDrawerContent={renderDrawerContent} renderSceneContent={renderSceneContent} dimensions={dimensions} /> </DrawerStatusContext.Provider> ); }

export default function DrawerView({ navigation, ...rest }: Props) { return ( <DrawerViewBase navigation={navigation} {...rest} /> ); }

const styles = StyleSheet.create({ content: { flex: 1, }, });

Step 8: Connect the device to an android emulator or a physical device.

--if physical device: make sure developers options and usb debugging are enabled. connect the phone to the device with usb. type 'adb devices' in the terminal and make sure its authenticated.

Step 9: Run npm start and press a to run on android.

# How to Use App:

Step 1: Click on Register and then create an account:

A screenshot of a login screen

Description automatically generated

A screenshot of a phone

Description automatically generated

A person and person with boxing gloves

Description automatically generatedStep 2: After logging in to the account choose a personal trainer and then click start.

Step 3: For new accounts the user will immediately converse with the chatbot for general information. This is a one time thing and will not happen when users login again.

A screenshot of a chat

Description automatically generated

Step4: Communicate with the chatbot. Once all 5 questions have been answered the user will be directed to the home page.

A screenshot of a cell phone

Description automatically generated

Step 5: Users can look at the workout schedule. Clicking on any of days will display the workout exercise if any is available at that day.

A screenshot of a phone

Description automatically generated

Step 6: Edit to the schedule days can be done by clicking the top right button on the workout schedule page.

A screenshot of a phone

Description automatically generated

Step 7: For starting workout go from the home page to the start workout page and then click on start workout. Inside the start workout page click play and then users can switch between exercise by clicking right and left.

A screenshot of a cell phone

Description automatically generated

A screenshot of a phone

Description automatically generated



A white wall with a white ceiling

Description automatically generatedStep 8: For time-based exercise make sure to click on start then go to the right position only then will the timer counter increase. If the correct position is not met timer will pause.

Step 9: From the home page users can go to workouts page where they can view various exercise available and how they are performed. During this page user will interact with a drop down to switch between exercise.

A screenshot of a cell phone

Description automatically generated

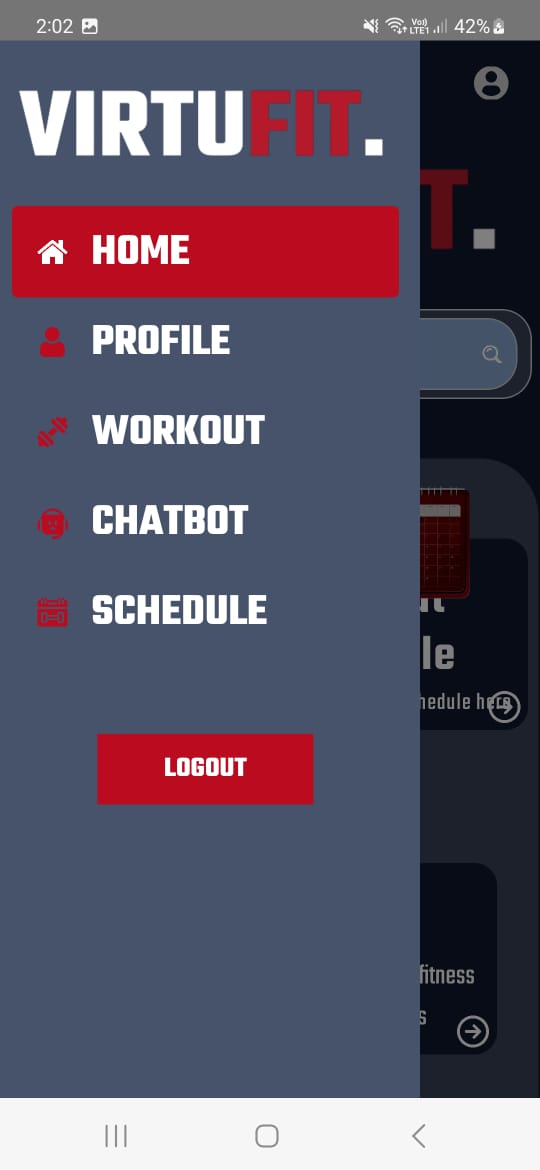
A screenshot of a body builder

Description automatically generated

Step 10: For quick navigation user can click on the side bar

A screenshot of a cell phone

Description automatically generated



Step 11: For any Changes click on profile page in top right of home page.

A screenshot of a cell phone

Description automatically generated

A screenshot of a phone

Description automatically generated

Step 12: In the profile page user can change their personal information.

A screenshot of a phone

Description automatically generated

A screenshot of a phone

Description automatically generated