Metmoi setup guide (Expo version)

Metmoi is React Native template application.

Using expo and typescript language to develop mobile app for both iOS and Android.

1. Install Node

Go to https://nodejs.org/en/download/, download latest LTS version then install it.

2. Watchman

Follow the Watchman installation guide to compile and install Watchman from source.

https://facebook.github.io/watchman/docs/install.html

> Watchman is a tool by Facebook for watching changes in the filesystem. It is highly recommended you install it for better performance and increased compatibility in certain edge cases (translation: you may be able to get by without installing this, but your mileage may vary; installing this now may save you from a headache later).

3. Expo CLI

Assuming that you have Node 12 LTS or greater installed, you can use npm to install the Expo CLI command line utility:

```
npm install -g expo-cli
```

4. Android development environment

Setting up your development environment can be somewhat tedious if you're new to Android development. If you're already familiar with Android development, there are a few things you may need to configure. In either case, please make sure to carefully follow the next few steps.

4.1. Install Android Studio

Download and install Android Studio. While on Android Studio installation wizard, make sure the boxes next to all of the following items are checked:

- Android SDK
- Android SDK Platform

- Android Virtual Device

Then, click "Next" to install all of these components.

> If the checkboxes are grayed out, you will have a chance to install these components later on.

Once setup has finalized and you're presented with the Welcome screen, proceed to the next step.

4.2. Install the Android SDK

Android Studio installs the latest Android SDK by default. Building a React Native app with native code, however, requires the Android 10 (Q) SDK in particular. Additional Android SDKs can be installed through the SDK Manager in Android Studio.

To do that, open Android Studio, click on "Configure" button and select "SDK Manager".

> The SDK Manager can also be found within the Android Studio "Preferences" dialog, under Appearance & Behavior → System Settings → Android SDK.

Select the "SDK Platforms" tab from within the SDK Manager, then check the box next to "Show Package Details" in the bottom right corner. Look for and expand the Android 10 (Q) entry, then make sure the following items are checked:

- Android SDK Platform 29
- Intel x86 Atom_64 System Image or Google APIs Intel x86 Atom System Image

Next, select the "SDK Tools" tab and check the box next to "Show Package Details" here as well. Look for and expand the "Android SDK Build-Tools" entry, then make sure that 29.0.2 is selected.

Finally, click "Apply" to download and install the Android SDK and related build tools.

4.3. Configure the ANDROID_HOME environment variable

The React Native tools require some environment variables to be set up in order to build apps with native code.

Add the following lines to your $HOME/.bash_profile or $HOME/.bashrc (if you are using zsh then ~/.zprofile or ~/.zshrc) config file:

```
export ANDROID_HOME=$HOME/Android/Sdk
export PATH=$PATH:$ANDROID_HOME/emulator
export PATH=$PATH:$ANDROID_HOME/tools
export PATH=$PATH:$ANDROID_HOME/tools/bin
export PATH=$PATH:$ANDROID_HOME/platform-tools
```

> .bash_profile is specific to bash. If you're using another shell, you will need to edit the appropriate shell-specific config file.

Type source $HOME/.bash_profile for bash or source $HOME/.zprofile to load the config into your current shell. Verify that ANDROID_HOME has been set by running echo $ANDROID_HOME and the appropriate directories have been added to your path by running echo $PATH.

> Please make sure you use the correct Android SDK path. You can find the actual location of the SDK in the Android Studio "Preferences" dialog, under Appearance & Behavior → System Settings → Android SDK.

5. iOS development environment

You will need Node, Watchman, the React Native command line interface, Xcode and CocoaPods.

While you can use any editor of your choice to develop your app, you will need to install Xcode in order to set up the necessary tooling to build your React Native app for iOS.

5.1. Xcode

The easiest way to install Xcode is via the Mac App Store. Installing Xcode will also install the iOS Simulator and all the necessary tools to build your iOS app.

If you have already installed Xcode on your system, make sure it is version 10 or newer.

5.2. Command Line Tools

You will also need to install the Xcode Command Line Tools. Open Xcode, then choose "Preferences..." from the Xcode menu. Go to the Locations panel and install the tools by selecting the most recent version in the Command Line Tools dropdown.

5.3. Installing an iOS Simulator in Xcode

To install a simulator, open Xcode > Preferences... and select the Components tab. Select a simulator with the corresponding version of iOS you wish to use.

6. Environment confirmation

In Expo template application, please setup environment follow Expo environment setup guide first.

https://docs.timistudio.dev/docs/environment-setup/expo-environment-setup

Then to confirm all setup is ok, please execute command below to verify.

```
$ yarn -v
1.22.17
$ node -v
v14.17.5
$ npm -v
```

```
    6.14.14
    $ expo --version
    5.0.3
```

If you are using MacOS and work with iOS, please check your XCode is latest version.

```
    $ /usr/bin/xcodebuild -version
    Xcode 13.2.1
    Build version 13C100
```

6. Run for development

```
cd <react-native-project-root>
yarn install
```

If you are using npm without yarn

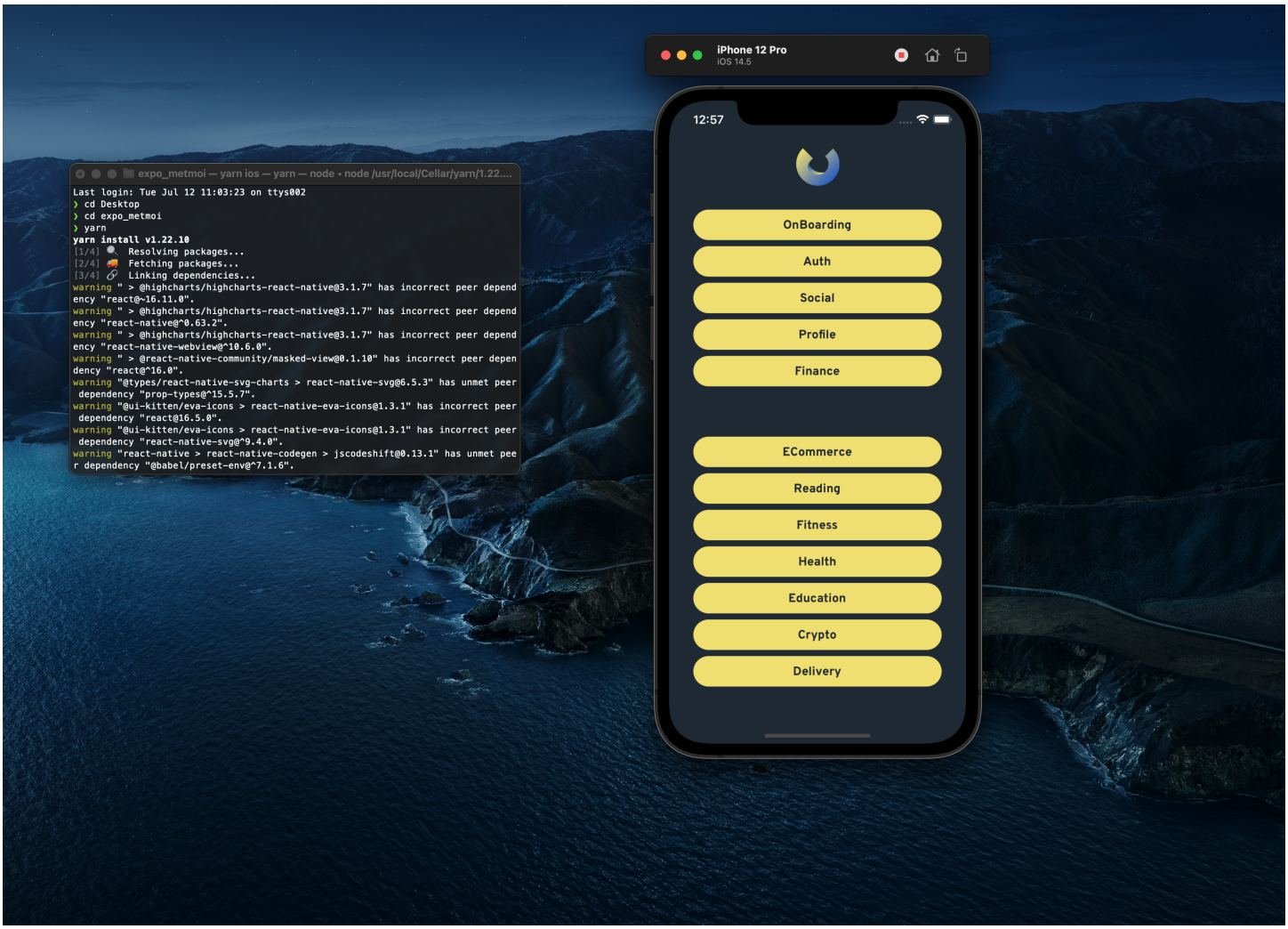
```
cd <react-native-project-root>
npm install
```

Finally start simulator with command below

```
yarn android
# or
yarn ios
```

If you are using npm without yarn

```
npm run android
# or
npm run ios
```


iPhone 12 Pro
iOS 14.5
12:57
OnBoarding
Auth
Social
Profile
Finance
ECommerce
Reading
Fitness
Health
Education
Crypto
Delivery
expo_metmoi — yarn ios — yarn — node ◂ node /usr/local/Cellar/yarn/1.22....
Last login: Tue Jul 12 11:03:23 on ttys002
❯ cd Desktop
❯ cd expo_metmoi
❯ yarn
yarn install v1.22.10
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
warning " > @highcharts/highcharts-react-native@3.1.7" has incorrect peer depend
ency "react@~16.11.0".
warning " > @highcharts/highcharts-react-native@3.1.7" has incorrect peer depend
ency "react-native@^0.63.2".
warning " > @highcharts/highcharts-react-native@3.1.7" has incorrect peer depend
ency "react-native-webview@^10.6.0".
warning " > @react-native-community/masked-view@0.1.10" has incorrect peer depen
dency "react@^16.0".
warning "@types/react-native-svg-charts > react-native-svg@6.5.3" has unmet peer
dependency "prop-types@^15.5.7".
warning "@ui-kitten/eva-icons > react-native-eva-icons@1.3.1" has incorrect peer
dependency "react@16.5.0".
warning "@ui-kitten/eva-icons > react-native-eva-icons@1.3.1" has incorrect peer
dependency "react-native-svg@^9.4.0".
warning "react-native > react-native-codegen > jscodeshift@0.13.1" has unmet pee
r dependency "@babel/preset-env@^7.1.6".

7. File structure

```
- assets/                                  # Image asset files
- components/                              # React Components
- constants/                               # Project constant files
- hooks/                                   # React hooks
- i18n/                                    # Internationalization files
- navigation/                              # React native navigation
- screens/                                 # Project screens
- utils/                                   # Helper files
- App.tsx
- AuthContext.tsx
- ThemeContext.tsx
- app.json
- index.js
- babel.config.js
- package.json                             # Node packages
- tsconfig.json                            # Typescript config
- highcharts-react-native.ts
- react-native-web-refresh-control.d.ts
```

8. Customization

Project UI template base on UI Kitten and Eva Design System

https://akveo.github.io/react-native-ui-kitten/docs/getting-started/what-is-ui-kitten#what-is-ui-kitten

1. Fonts

This how to add fonts to project you need to follow all these steps

https://fonts.google.com/

assets/fonts/

```
Montserrat-Regular.ttf
Overpass-Bold.ttf
...
```

hooks/useCachedResources.ts

```
"Overpass-Bold": require("../assets/fonts/Overpass-Bold.ttf"),
"Montserrat-Regular": require("../assets/fonts/Montserrat-Regular.ttf"),
...
```

then u need to import on file mapping

constants/theme/mapping.json

```
...
"text-font-family-Overpass-Regular": "Overpass-Regular",
"text-font-family-Montserrat-Regular": "Montserrat-Regular",
...
```

2. Icons

assets/icons/index.ts

```
website: require("./ic_website.png"),
leftArrow: require("./ic_left_arrow.png"),
rightArrow: require("./ic_right_arrow.png"),
...
```

then add to pack icon or you can create your own pack

assets/AssetIconsPack.tsx

```
const AssetIconsPack: IconPack<ImageProps | SvgProps> = {
  name: "assets", - *name package icons*
  icons: {
    website: createIcon(Icons.website),
    leftArrow: createIcon(Icons.leftArrow),
    rightArrow: createIcon(Icons.rightArrow),
  },
 };

 export default AssetIconsPack;
...
```

final step you need add to IconRegistry on App.tsx

App.tsx

```
import AssetIconsPack from "assets/AssetIconsPack";

<SafeAreaProvider>
  <ThemeContext.Provider value={{ theme, toggleTheme }}>
    <IconRegistry icons={[EvaIconsPack, AssetIconsPack]} />
    ...
  </ThemeContext.Provider>
</SafeAreaProvider>
```

3. Colors

Add Colors

https://colors.eva.design/

- contants/theme/appTheme.json

```
"color-basic-100": "#FFFFFF",
"color-basic-200": "#FAFAFA",
"color-basic-300": "#F6F6F6",
"color-basic-400": "#E0E0E0",
"color-basic-500": "#F0F0F0",
"color-basic-600": "#F5F7FA",
"color-basic-700": "#1F2933",
"color-basic-800": "#323F4B",
...
```

Theme Variables

- light.json

```
"background-basic-color-1": "$color-basic-700",
"background-basic-color-2": "$color-basic-800",
"background-basic-color-3": "$color-basic-700",
"text-basic-color": "$color-basic-600",
"text-placeholder-color": "$color-basic-1200",
...
```

- dark.json

```
"background-basic-color-1": "$color-basic-700",
"background-basic-color-2": "$color-basic-800",
"background-basic-color-3": "$color-basic-700",
"text-basic-color": "$color-basic-600",
```

```
    "text-placeholder-color": "$color-basic-1200",
    ...
```

Setup DarkMode

```
import React from 'react';

type AppTheme = {
  theme: 'light' | 'dark';
  toggleTheme: () => void;
};

export default React.createContext<AppTheme>({
  theme: 'light',
  toggleTheme: () => {},
});
```

- App.tsx

```
import ThemeContext from "./ThemeContext";

const [theme, setTheme] = React.useState<"light" | "dark">("dark");

React.useEffect(() => {
  AsyncStorage.getItem("theme").then((value) => {
    if (value === "light" || value === "dark") setTheme(value);
  });
}, []);

const toggleTheme = () => {
  const nextTheme = theme === "light" ? "dark" : "light";
  AsyncStorage.setItem("theme", nextTheme).then(() => {
    setTheme(nextTheme);
  });
};

<SafeAreaProvider>
  <ThemeContext.Provider value={{ theme, toggleTheme }}>
    ...
  </ThemeContext.Provider>
</SafeAreaProvider>
```

Use Theme

```
import { useTheme } from "@ui-kitten/components";

const theme = useTheme();
```

```
{ backgroundColor: theme["color-basic-1300"] },
{ backgroundColor: theme["background-basic-color-2"] },
```

Switch Theme

- create file hooks/useAppTheme.ts

```
import React from 'react';

import ThemeContext from '../ThemeContext';

export default () => {
  const appTheme = React.useContext(ThemeContext);
  return appTheme;
};
```

```
import useAppTheme from 'hooks/useAppTheme';

const { theme, toggleTheme } = useAppTheme();
```

4. Components

Default components UI Kitten

https://raw.githubusercontent.com/eva-design/eva/master/packages/eva/mapping.json

Basic components config on file constants/theme/mapping.json

1. Text

https://akveo.github.io/react-native-ui-kitten/docs/components/text/overview#text

We create our own text , for ease of use and use in different places

Add Category and Status

```
- Add name category and status on MyTextProps

category?:
    | "h6"
    | "title-1"
    | "title-2"
    ...
```

```
    status?:
        | EvaStatus
        | "placeholder"
        | "white"
        | "black"
        ...
```

- Config category and status on file constants/theme/mapping.json
- Color value on file appTheme.json,light.json or dark.json

```
    "strict": {
        "text-h6-font-size": 12,
        "text-h6-font-weight": "700",
        "text-h6-font-family": "$text-font-family-FPro-bold",
        "text-title-1-font-size": 32,
        "text-title-1-font-weight": "700",
        "text-title-1-font-family": "$text-font-family",
        "text-title-2-font-size": 28,
        "text-title-2-font-weight": "700",
        "text-title-2-font-family": "$text-font-family",
        ...
    },
    "components": {
      "Text": {
          "meta": {
            "variantGroups": {
              "category": {
                "h6": {
                  "default": false
                },
                "title1": {
                  "default": false
                },
                "title2": {
                  "default": false
                },
                ...
              },
              "status": {
                "black": {
                  "default": false
                },
                "white": {
                  "default": false
                },
                "placeholder": {
                  "default": false
                },
                ...
              }
            }
          },
```

```
      "appearances": {
        "default": {
          "mapping": {
            "color": "text-basic-color"
          },
          "variantGroups": {
            "category": {
              "h6": {
                "fontSize": "text-h6-font-size",
                "fontWeight": "text-h6-font-weight",
                "fontFamily": "text-h6-font-family"
              },

              "title1": {
                "fontSize": "text-title-1-font-size",
                "fontWeight": "text-title-1-font-weight",
                "fontFamily": "text-title-1-font-family"
              },
              "title2": {
                "fontSize": "text-title-2-font-size",
                "fontWeight": "text-title-2-font-weight",
                "fontFamily": "text-title-2-font-family"
              },
              ...
            },
            "status": {
              "placeholder": {
                "color": "text-placeholder-color"
              },
              "white": {
                "color": "color-basic-100"
              },
              "black": {
                "color": "color-basic-700"
              },
            }
          }
        },
      }
    }
```

Use Text

```
import Text from "components/Text";

<Text category="title2" status="placeholder">Metmoi</Text>
```

2. Layout

- Layout is the most abstract component on top of which all other UI Kitten components are built. By default, it renders a `View` element

```
"Layout": {
      "meta": {
        "scope": "mobile",
        "parameters": {
          "backgroundColor": {
            "type": "string"
          }
        },
        "appearances": {
          "default": {
            "default": true
          }
        },
        "variantGroups": {
          "level": {
            "5": {
              "default": true
            },
            "6": {
              "default": false
            },
            "7": {
              "default": false
            },
            "8": {
              "default": false
            }
          }
        }
      },
      "appearances": {
        "default": {
          "variantGroups": {
            "level": {
              "5": {
                "backgroundColor": "$background-basic-color-5"
              },
              "6": {
                "backgroundColor": "$background-basic-color-6"
              },
              "7": {
                "backgroundColor": "$background-basic-color-7"
              },
              "8": {
                "backgroundColor": "$background-basic-color-8"
              }
            }
          }
        }
```

```
          }
        },
```

3. Container

- Content is `Layout` with flex to 1 and use safe area insets. It renders a `View` element.

```
import Container from "components/Container";

<Container>
  <Content>
    <Text>Metmoi</Text>
  </Content>
</Container>
```

4. Content

- Content is ScrollView with helpful style shorthand. It renders a `ScrollView` element.

```
import Content from "components/Content";

<Content>
  <Text>Metmoi</Text>
</Content>
```

SUPPORT INFORMATION

Thank for your purchase, feel free to contact with us if you have any trouble when install application.

- Email: admin@timistudio.dev
- Facebook official fanpage: https://www.facebook.com/timistudio.dev