Introduction

I searched and searched but could not find it..!! I was busy looking for a react native package that could help with creating a multi-step formik and yup forms but i could not find one. Once i was convinced i could not find one, i sought to create it and make it available to others so that others after me could have their lives easier. So i created this library out of need, should you find it useful, that will put a smile on my face...

Installation

```
npm i @mana/simple-react-native-formik-wizard --save
```

Usage

The package uses its own input components, also included in the package. The availed input components are: the comprehensive list of available input components are listed below:

- **AppForm** - the form component that wraps all other input components Props are:
    1. *initialValues* - the initial values of the form, passed internally by the FormikStepper component.
    2. *validationSchema* - Yup validationSchema passed internally by the FormikStepper Component.
    3. *onSubmit* - The function to execute when next/save button is clicked, supplied internally by the FormikStepper component.
- **AppFormPicker** - single select picker component. Its props are:
    1. *name* * - the name of the variable that holds selected option, exactly as is in initialValues array.
    2. *label* * - the prompt describing what the user is exped to do.
    3. *items* * - the items to display on the picker. should be in the format `{label:x, value:y}` where label is the data shown to user and value is the data return on selection.
    4. *icon* - the name of the icon to show next to the picker *(optional, uses MaterialIcons from react-native-vector-icons)*.
    5. *callback* - a callback to get the value of selected option for use like to show or hide other components *(optional)*
- **AppFormNumberInput** - number input component. Its props are:
    1. *name* * - the name of the variable that holds selected option, exactly as is in initialValues array.
    2. *label* * - the prompt describing what the user is exped to do.
- **AppFormField** - simple text input component, wraps around react native's `TextInput` component. All the props for `TextInput` component can be passed to it. Specific props required are:
    1. *name* * - the name of the variable that holds typed data, exactly as is in initialValues array.
    2. *label* * - the prompt describing what the user is exped to do.
- **AppSubmitButtonSmall** - a small submit button component used to move to the next step or submit the whole form. Its props are:
    1. *title* - the title of the button
    2. *backgroundColor* - string representing a background color eg. '#eee' or 'green' *(optional)*.
    3. *textColor* - text color string *(optional)*.
    4. *textSize* - text size as a number *(optional)*;
- **AppFormCheckBoxes** - component for multi-select checkboxes. Its props are:

    1. *name* * - the name of the variable that holds selected options, exactly as is in initialValues array.
    2. *label* * - the prompt describing what the user is exped to do.
    3. *items* * - the items to display on the picker. should be in the format `{label:x, value:y}` where label is the data shown to user and value is the data return on selection.
    4. *boxColor* - color for the box on each checkbox *(optional)*
    5. *checkedColor* - color string for the checkmark when checkbox is selected. *(optional)*
    6. *unCheckedColor* - color string for when the checkbox is not selected. *(optional)*
    7. *boxSize* - the size of each size of the checkbox box, default is 30 density pixels. *(optional)*
    8. *labelPosition* - determines whether to place label before or after each checkbox box. Its should be either `start` or `end`.
- **AppFormRadio** - component for radiobuttons. Its props are:
    1. *name* * - the name of the variable that holds selected option, exactly as is in initialValues array.
    2. *label* * - the prompt describing what the user is exped to do.
    3. *items* * - the items to display on the picker. should be in the format `{label:x, value:y}` where label is the data shown to user and value is the data return on selection.
- **AppFormSwitch** - the wrapper component for a custom made switch. Its props are:
    1. *name* * - the name of the variable that holds selected options, exactly as is in initialValues array.
    2. *label* * - the prompt describing what the user is exped to do.
- **AppSubmitButton** - A submit button component that takes 100% width. its props are the same as the **AppSubmitButtonSmall.**
- **AppFormSingleCheckBox** - the custom checkbox component wrapper. Its props are:
    1. *name* * - the name of the variable that holds the value, exactly as is in initialValues array.
    2. *label* * - the prompt describing what the user is exped to do.
    3. All other props on **AppFormCheckBoxes**

Steps to take - below is an example

1. Create a parent component that will hold the data collected from the form. The component should also have a state variable eg. `mergedValues` and its setter function, say, `setMergedValues`.
2. Create an initalValues array of objects, each array element representing initialValues of a corresponding step.
3. Create an array of validationSchema objects, each element representing validation schema for a corresponding step.
4. Create the steps components, the ones that will be executed for each step in the multi-step form.
5. Create an object with keys 0...n where n is `number-of-steps - 1`, and values which are ReactNode references for each step.
6. Create an array of strings of discriptions/labels of each step (these descriptions are like labels to each step). The number of labels should tally with the number of steps.
7. create an onSubmit method that receives the final values as an object.

Example

- create parent component and create a state variable and its setter.

```
export default function Example(){
 const [mergedValues, setMergedValues] = useState({});
```

```
        ...
    }
```

- Create an initalValues array of objects and validationSchema array:

```
const initialValues = [
  {
    fullName: "", // text field
    gender: "", // picker component
    category: 0, // Number input component
    isItTrue: false, // switch component
  },
  {
    progLang: [], // multiselect checkboxes
    hobby: "", //radio
  },
];

const validationSchema = [
  yup.object().shape({
    fullName: yup
      .string()
      .required("required")
      .test(
        "",
        "please enter fullname",
        (name: any) => name && name.split(" ").length > 1
      ),
    gender: yup.string().required("required"),
    category: yup.number().min(1, "Number to be positive"),
  }),
  yup.object().shape({
    progLang: yup.array().length(2, "Array must have at least 2 items"),
  }),
];
```

- create step components using the provided input component wrappers (buttons, textfeld, radio buttons, etc.), the step components should have exactly the following signature:

```
  import {
  FormikStepper, // main stepper component
  AppForm, // the form component that wraps all other input components
  AppFormPicker, // picker component
  AppFormNumberInput, // number input component
  AppFormField,  // simple text input component
  AppSubmitButtonSmall, // a small submit button component
  AppFormCheckBoxes, // component for multi-select checkboxes
  AppFormRadio, // compinent for checkboxes
  ...
```

```
} from "@mana/simple-react-native-formik-wizard";


const Step1 = ({ initValues, validationSchema, onNextStep, onBack }) => {
    const genderOptions = [
        { label: "Male", value: "0" },
        { label: "Female", value: "1" },
        { label: "Other", value: "2" },
    ];
    return (
        <AppForm
        initialValues={initValues}
        validationSchema={validationSchema}
        onSubmit={onNextStep}
        >
        <AppFormField name={"fullName"} label={"Your full name"} />
        <AppFormPicker
            name={"gender"}
            items={genderOptions}
            label={"select gender"}
            icon="none"
        />

        <AppFormSwitch name={"isItTrue"} label={"Accept Terms"} />
        <AppFormNumberInput name={"category"} label={"Select category number"}
/>

        <View
            style={{ width: "100%", alignItems: "flex-end", paddingHorizontal:
10 }}
        >
            <AppSubmitButtonSmall title={"Next"} /> // you can warp it inside a
view to position it
        </View>
        </AppForm>
    );
};
```

- and then step 2:

```
const Step2 = ({ initValues, validationSchema, onNextStep, onBack }) => {
  const progLangOptions = [
    { label: "Javascript", value: "0" },
    { label: "Typescript", value: "1" },
    { label: "Java", value: "2" },
    { label: "VB.Net", value: "3" },
    { label: "C#", value: "4" },
    { label: "C++", value: "5" },
  ];
  const hobbiesOptions = [
    { label: "Music", value: "0" },
    { label: "Sports", value: "1" },
```

```
  ];
  return (
    <AppForm
      initialValues={initValues}
      validationSchema={validationSchema}
      onSubmit={onNextStep}
    >
      <AppFormCheckBoxes
        items={progLangOptions}
        label={"Select favorite Language"}
        name={"progLang"}
        callback={null}
      />
      <AppFormRadio
        items={hobbiesOptions}
        name={"hobby"}
        label={"Select hobby"}
        row="column"
      />

      <View
        style={{
          width: "100%",
          marginHorizontal: 30,
          flexDirection: "row",
          justifyContent: "space-around",
        }}
      >
        <AppButtonSmall title={"Back"} onPress={onBack} />
        <AppSubmitButtonSmall title={"Save"} />
      </View>
    </AppForm>
  );
};
```

- create an object with ReactNode values:

```
const steps = { 0: Step1, 1: Step2 };
```

- create an array of string labels:

```
const stepLabels = ["Step 1", "Step 2"];
```

- create an onSubmit method that receives the final values as an object:

```
const submit = (values: []) => {
  console.log(values);
```

```
  };
```

- implement the return method of the parent component:

```
return (
  <FormikStepper
    steps={steps}
    stepLabels={stepLabels}
    initialValues={initialValues}
    validationSchema={validationSchema}
    mergedValues={mergedValues}
    setMergedValues={setMergedValues}
    onSubmit={submit}
  />
);
```

Full example

```
// more imports as necessary
import {
  FormikStepper, // main stepper component
  AppForm, // the form component that wraps all other input components
  AppFormPicker, // picker component
  AppFormNumberInput, // number input component
  AppFormField,  // simple text input component
  AppSubmitButtonSmall, // a small submit button component
  AppFormCheckBoxes, // component for multi-select checkboxes
  AppFormRadio, // compinent for checkboxes
  ...
  } from "@mana/simple-react-native-formik-wizard";


export default function Example() {
  const [mergedValues, setMergedValues] = useState({});

  const submit = (values: []) => {
    console.log(values);
  };

  const initialValues = [
    {
      fullName: "", // text field
      gender: "", // picker component
      category: 0, // Number input component
      isItTrue: false, // switch component
    },
    {
      progLang: [], // multiselect checkboxes
      hobby: "", //radio
```

```
    },
  ];

  const validationSchema = [
    yup.object().shape({
      fullName: yup
        .string()
        .required("required")
        .test(
          "",
          "please enter fullname",
          (name: any) => name && name.split(" ").length > 1
        ),
      gender: yup.string().required("required"),
      category: yup.number().min(1, "Number to be positive"),
    }),
    yup.object().shape({
      progLang: yup.array().length(2, "Array must have at least 2 items"),
    }),
  ];

  const steps = { 0: Step1, 1: Step2 };

  const stepLabels = ["Step 1", "Step 2"];

  return (
    <FormikStepper
      steps={steps}
      stepLabels={stepLabels}
      initialValues={initialValues}
      validationSchema={validationSchema}
      mergedValues={mergedValues}
      setMergedValues={setMergedValues}
      onSubmit={submit}
    />
  );
}

// Now create first step
const Step1 = ({ initValues, validationSchema, onNextStep, onBack }) => {
  const genderOptions = [
    { label: "Male", value: "0" },
    { label: "Female", value: "1" },
    { label: "Other", value: "2" },
  ];
  return (
    <AppForm
      initialValues={initValues}
      validationSchema={validationSchema}
      onSubmit={onNextStep}
    >
      <AppFormField name={"fullName"} label={"your full name"} />
      <AppFormPicker
        name={"gender"}
```

```
          items={genderOptions}
          label={"select gender"}
          icon="none"
        />

        <AppFormSwitch name={"isItTrue"} label={"Accept Terms"} />
        <AppFormNumberInput name={"category"} label={"Select category number"} />

        <View
          style={{ width: "100%", alignItems: "flex-end", paddingHorizontal: 10 }}
        >
          <AppSubmitButtonSmall title={"Next"} />
        </View>
      </AppForm>
    );
};

// Now create second step
const Step2 = ({ initValues, validationSchema, onNextStep, onBack }) => {
  const progLangOptions = [
    { label: "Javascript", value: "0" },
    { label: "Typescript", value: "1" },
    { label: "Java", value: "2" },
    { label: "VB.Net", value: "3" },
    { label: "C#", value: "4" },
    { label: "C++", value: "5" },
  ];
  const hobbiesOptions = [
    { label: "Music", value: "0" },
    { label: "Sports", value: "1" },
  ];
  return (
    <AppForm
      initialValues={initValues}
      validationSchema={validationSchema}
      onSubmit={onNextStep}
    >
      <AppFormCheckBoxes
        items={progLangOptions}
        label={"Select favorite Language"}
        name={"progLang"}
        callback={null}
      />
      <AppFormRadio
        items={hobbiesOptions}
        name={"hobby"}
        label={"Select hobby"}
        row="column"
      />

      <View
        style={{
          width: "100%",
          marginHorizontal: 30,
```

```
            flexDirection: "row",
            justifyContent: "space-around",
          }}
        >
          <AppButtonSmall title={"Back"} onPress={onBack} />
          <AppSubmitButtonSmall title={"Save"} />
        </View>
      </AppForm>
    );
};

const styles = StyleSheet.create({});
```