Hands-on practical test

Date and duration of practical task: (6.12.2020) 3 hours + (9.12.2020) 6 hours + (10.12.2020) 7 hours + (11.12.2020) 5 hours + (13.12.2020) 4 hours = 25 hours

Section-1: Reviewing mobile app idea

App Idea

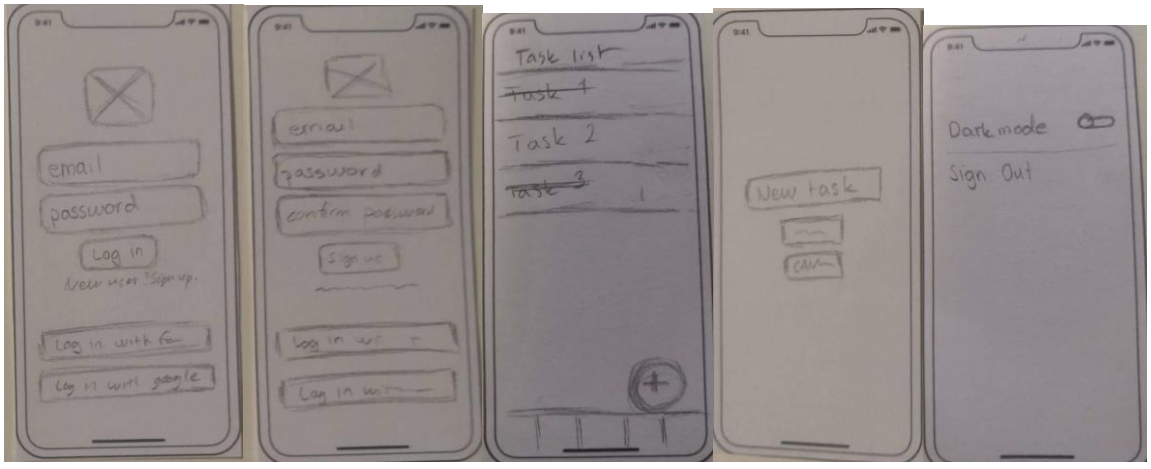
My idea is to make a far better Task app than I did in the last module. Modifications I will be making:

- Option of signing up and logging in with email, google account or facebook account
- Connected to firebase
- Task list will look more modern and bigger
- Add –button's location and design different
- Deleting task is done by a long press
- Tapping the task will add a line over the task to indicate it is done
- Bottom tab navigation
- Settings which include dark mode option and log out function
- Color theme will be more comfortable and more compatible

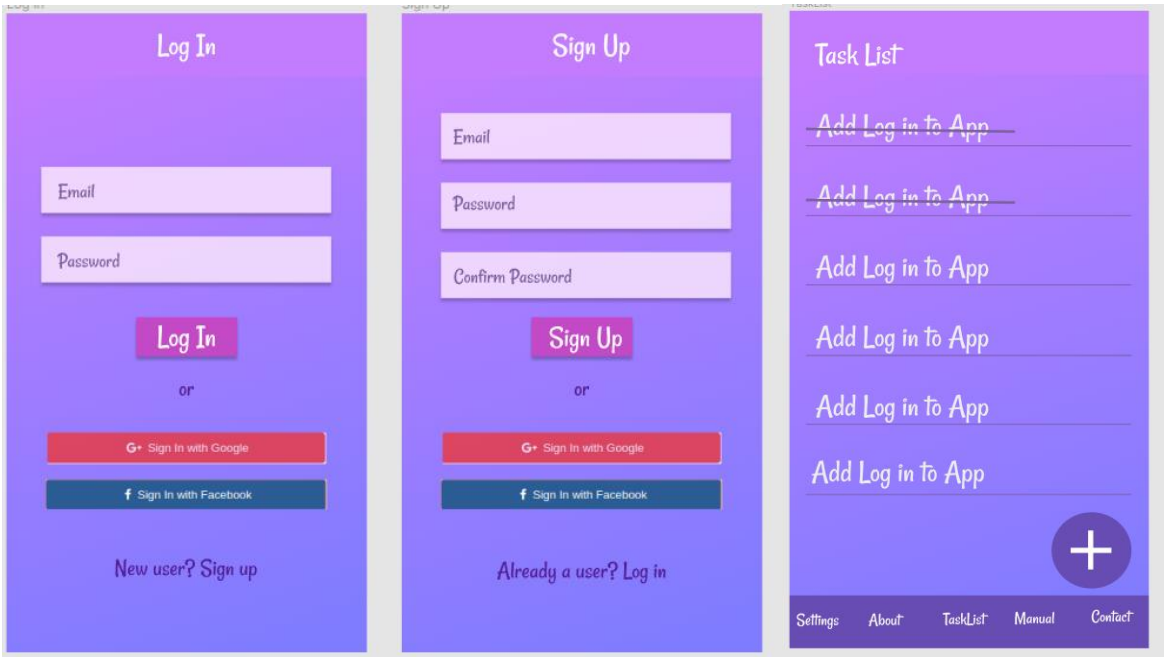
This kind of an app has all the needed functions to list all tasks and to keep track which tasks has been done. My goal is to make an app that is used multiple times a day and easy to use. Future versions of the app will include different themes that can be bought to make the app more personalized.

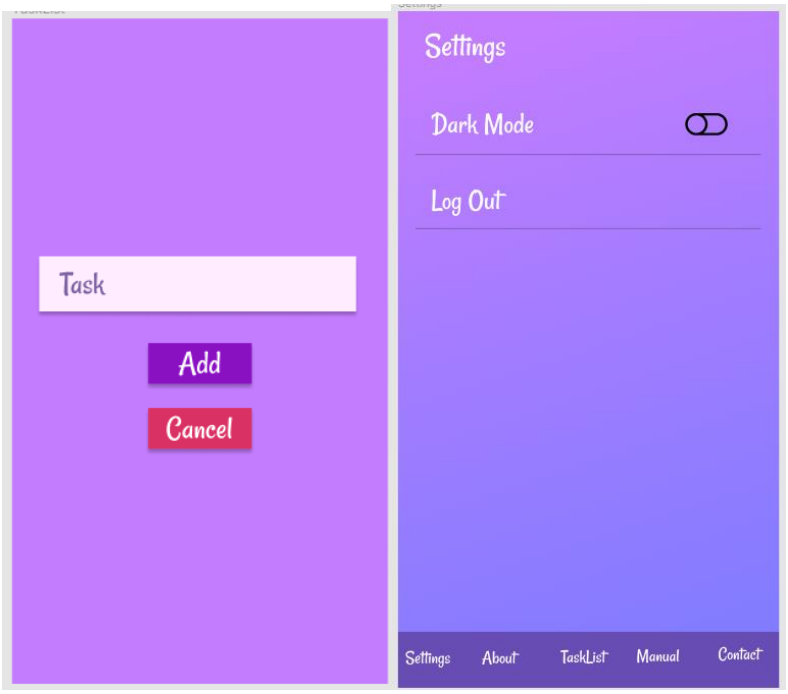
Section-2: Designing the Mobile App

First, I sketched out my idea on paper. I used the previous app as a base since I knew I could do that.



And after that I went and made a prototype of it with figma.





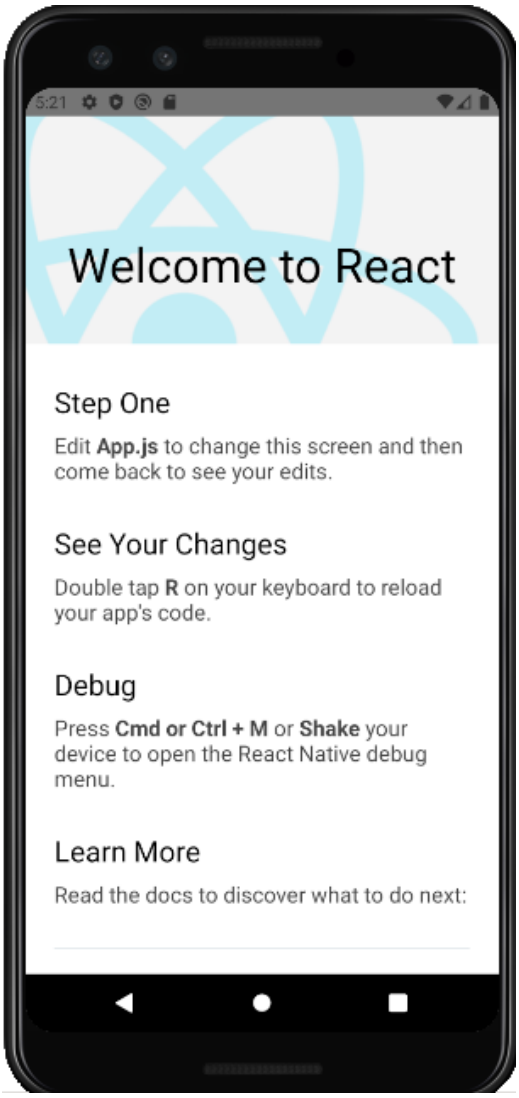
Section-3: Mobile App Development

I need to remember to add "About" and "Contact Information" pages to the app. I will add these to be accessed from the side navigation. All this is added because it will affect the given points and are things I did not think about, but are good to have in the app.

Step-by-step manual on how to make this app:

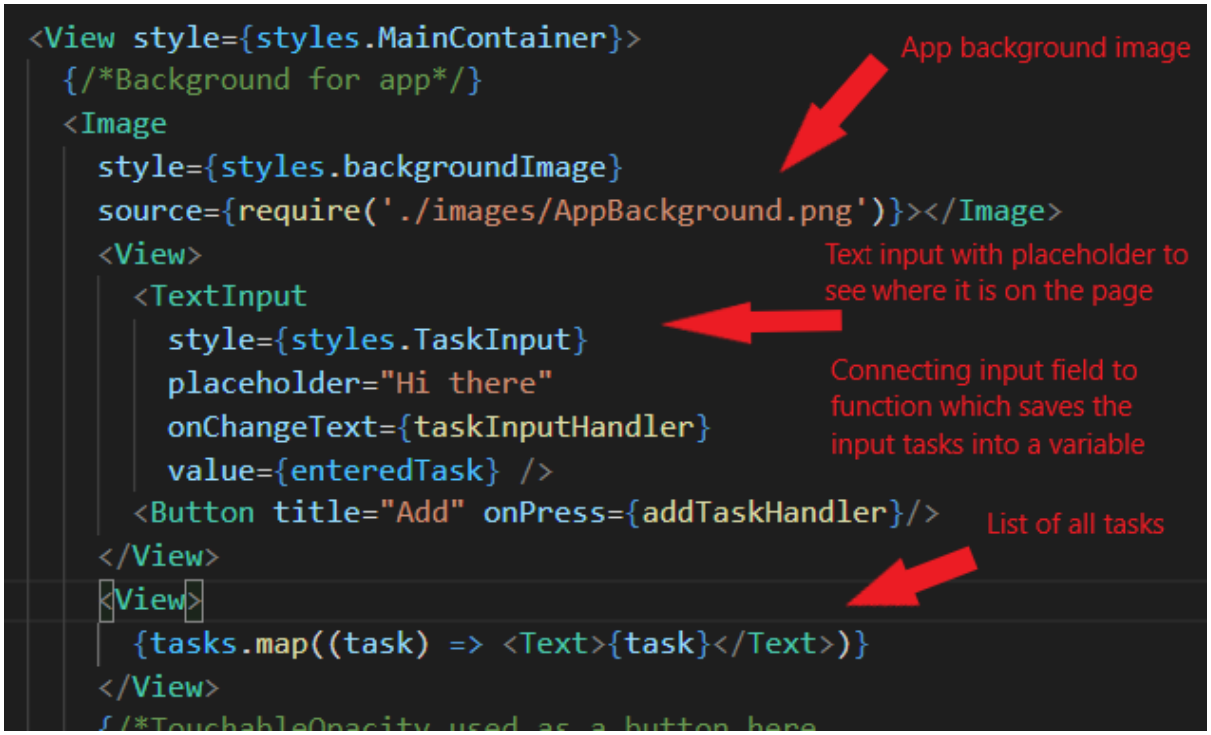
Step 1) Create the project

Open the command prompt and go the desired folder and run ***npx react-native init ProjecName .*** My project is called Task List. After this I opened the project into an emulator with commands ***npx react-native start*** and ***npx react-native run-android*** to see if it works.



Step 2) Making the task list

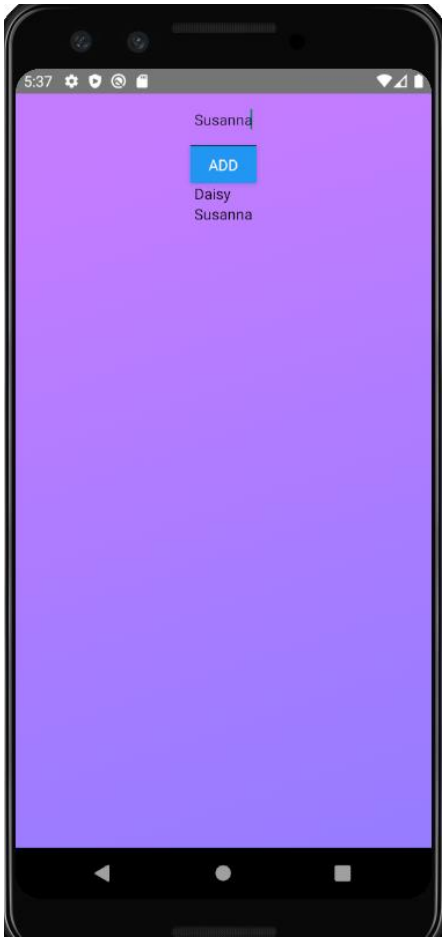
I started by adding a background image I made before, it is a png. After this I added a textinput, where the user can write the task and an add button that will add the input task into a list.



Here are the functions I am using

```
const App: () => React$Node = () => {
  // creating variables to save what user input
  const [enteredTask, setEnteredTask] = useState('');
  // empty array for tasks
  const [tasks, setTasks] = useState([]);

  // function to save the entered task into variable
  const taskInputHandler = (enteredTask) => {
    setEnteredTask(enteredTask);
  };

  // function that executed when add button is pressed
  const addTaskHandler = () => {
    setTasks(currentTasks => [...tasks, enteredTask]);
  };
```

This is how the app looks like at the moment, I have some styling properties which this might look a little different is someone would follow my code.



After seeing how my background looks like on the emulator, I might change it to be a more obvious gradient later.

Next, I am going to style the list how I have planned. I also changed it into a Flatlist and gave each task their own randomized id.

Here is the styling. I also added a google font to the project.

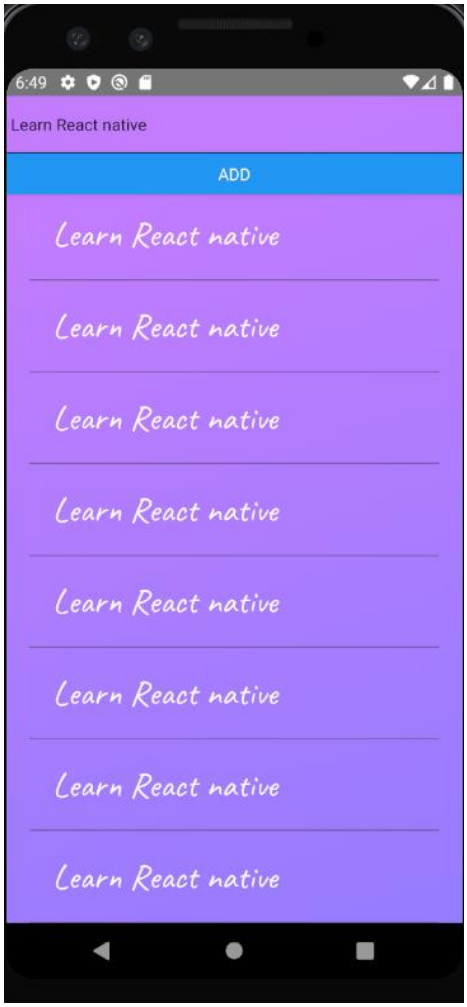
```
list: {
  padding: 10,
  borderBottomColor: "#7A5AA1",
  borderBottomWidth: 1,
  marginLeft: 20,
  marginRight: 20,
  padding: 20
},
listText: {
  fontSize: 30,
  fontFamily: "Caveat-Regular",
  color: "white"
```

```
    <Button title="Add" onPress={addTaskHandler}/>
  </View>
  <FlatList
  keyExtractor={(item, index) => item.id}
    data={tasks}
    renderItem={itemData => (
    <View style={styles.list}>
      <Text style={styles.listText}>{itemData.item.value}</Text>
    </View>
    )}
  />
```

Modified the addTaskHandler function a little bit

```
const addTaskHandler = () => {
  setTasks(currentTasks => [
  ...tasks,
  {id: Math.random().toString(), value: enteredTask }
  ]);
};
```

This is how it looks after.



Now that I can add tasks to a list and the list looks like I want, I am going to cut the code into components so it looks cleaner. All the code this far has been put into App.js.

I created a folder called components and added two files into it: TaskItem.js and TaskInput.js

TaskItem.js will take care of the tasks and putting them into a list. It is almost the same code as in what was done in App.js . The only differences that done is the routing because it is different. The app still works the same as before.

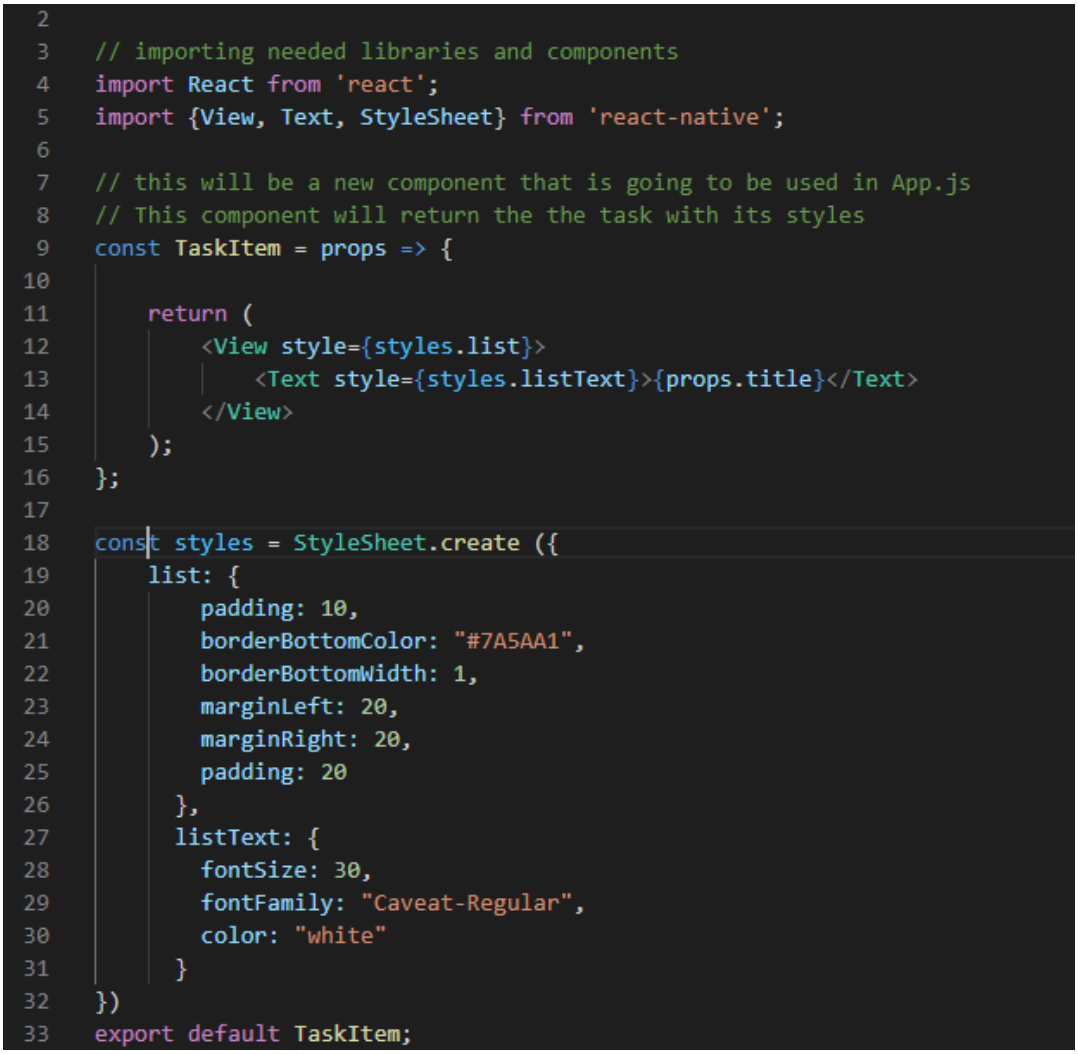
```
2
3   // importing needed libraries and components
4   import React from 'react';
5   import {View, Text, StyleSheet} from 'react-native';
6
7   // this will be a new component that is going to be used in App.js
8   // This component will return the the task with its styles
9   const TaskItem = props => {
10
11      return (
12          <View style={styles.list}>
13              <Text style={styles.listText}>{props.title}</Text>
14          </View>
15      );
16  };
17
18  const styles = StyleSheet.create ({
19      list: {
20          padding: 10,
21          borderBottomColor: "#7A5AA1",
22          borderBottomWidth: 1,
23          marginLeft: 20,
24          marginRight: 20,
25          padding: 20
26        },
27        listText: {
28          fontSize: 30,
29          fontFamily: "Caveat-Regular",
30          color: "white"
31        }
32  })
33  export default TaskItem;
```

The same is done to TaskInput.js. The 3 files should look like this now.

App.js

```
// importing components
import React, {useState} from 'react';
import { StyleSheet, View, Text, TextInput, Button, TouchableOpacity, Image, FlatList } from 'react-native';
// importing custom components
import TaskItem from './components/TaskItem';
import TaskInput from './components/TaskInput';

const App: () => React$Node = () => {
  // empty array for tasks
  const [tasks, setTasks] = useState([]);

  // function that executed when add button is pressed
  const addTaskHandler = taskTitle => {
    setTasks(currentTasks => [
    ...tasks,
    {id: Math.random().toString(), value: taskTitle }
    ]);
  };

  return (
    <>
    <View style={styles.MainContainer}>
      {/*Background for app*/}
      <Image
        style={styles.backgroundImage}
        source={require('./images/AppBackground.png')}></Image>
        <TaskInput onAddTask={addTaskHandler}/>
        <FlatList
        keyExtractor={(item, index) => item.id}
          data={tasks}
          renderItem={itemData => <TaskItem title={itemData.item.value}/>}
        />
        {/*TouchableOpacity used as a button here
        <TouchableOpacity
          style={styles.roundAddButton}>
            <Image
              style={styles.addButtonImage}
              source={require('./images/plusIconWhite.png')}></Image>
        </TouchableOpacity> */}
    </View>
    </>
  );
};

// Styling on page and images
const styles = StyleSheet.create({
  MainContainer: {
    flex: 1,
  },
  backgroundImage: {
```

TaskItem.js



```
2
3   // importing needed libraries and components
4   import React from 'react';
5   import {View, Text, StyleSheet} from 'react-native';
6
7   // this will be a new component that is going to be used in App.js
8   // This component will return the the task with its styles
9   const TaskItem = props => {
10
11      return (
12          <View style={styles.list}>
13              <Text style={styles.listText}>{props.title}</Text>
14          </View>
15      );
16  };
17
18  const styles = StyleSheet.create ({
19      list: {
20          padding: 10,
21          borderBottomColor: "#7A5AA1",
22          borderBottomWidth: 1,
23          marginLeft: 20,
24          marginRight: 20,
25          padding: 20
26        },
27        listText: {
28          fontSize: 30,
29          fontFamily: "Caveat-Regular",
30          color: "white"
31        }
32  })
33  export default TaskItem;
```

TaskInput.js

```
// importing needed libraries and components
import React from 'react';
import {View, Text, StyleSheet} from 'react-native';

// this will be a new component that is going to be used in App.js
// This component will return the the task with its styles
const TaskItem = props => {

    return (
        <View style={styles.list}>
            <Text style={styles.listText}>{props.title}</Text>
        </View>
    );
};

const styles = StyleSheet.create ({
    list: {
        padding: 10,
        borderBottomColor: "#7A5AA1",
        borderBottomWidth: 1,
        marginLeft: 20,
        marginRight: 20,
        padding: 20
      },
      listText: {
        fontSize: 30,
        fontFamily: "Caveat-Regular",
        color: "white"
      }
})
export default TaskItem;
```

Next I will be adding a delete function.

I wanted to use swipeable deletion, but I could not figure out how to add it to my project. So I will be doing the delete function by pressing the task for a little longer.

TaskItem.js

```
    return (
        <TouchableOpacity onLongPress={props.onDelete}>
            <View style={styles.list}>
                <Text style={styles.listText}>{props.title}</Text>
            </View>
        </TouchableOpacity>
    );
};
```

App.js

```
29    // function which removes task
30    // removes only when item is pressed for longer - tapping does not delete it
31    const removeTaskHandler = taskId => {
32      setTasks(currentTasks => {
33            return currentTasks.filter((task) => task.id !== taskId);
34      })}
35
36
37    return (
38      <>
39      <View style={styles.MainContainer}>
40        {/*Background for app*/}
41        <Image
42          style={styles.backgroundImage}
43          source={require('./images/AppBackground.png')}></Image>
44          <TaskInput visible={isAddMode} onAddTask={addTaskHandler} onCancel={cancelAddTaskHandler}/>
45          <FlatList
46          keyExtractor={(item, index) => item.id}
47            data={tasks}
48            renderItem={itemData => <TaskItem onDelete={removeTaskHandler.bind(this, itemData.item.id)} title={itemData.item.value}/>}
49          />
50          {/*TouchableOpacity used as a button here*/}
51          {/*Setting setIsAddMode to true to open modal where task can be input*/}
52          <TouchableOpacity
53            onPress={() => setIsAddMode(true)}
54            style={styles.roundAddButton}>
55              <Image
56                style={styles.addButtonImage}
57                source={require('./images/plusIconWhite.png')}></Image>
58          </TouchableOpacity>
59      </View>
60      </>
61    );
62  };
```

After this I decide that pressing a button on the list, will open a new window where the task can be input. This is done by using Modal.

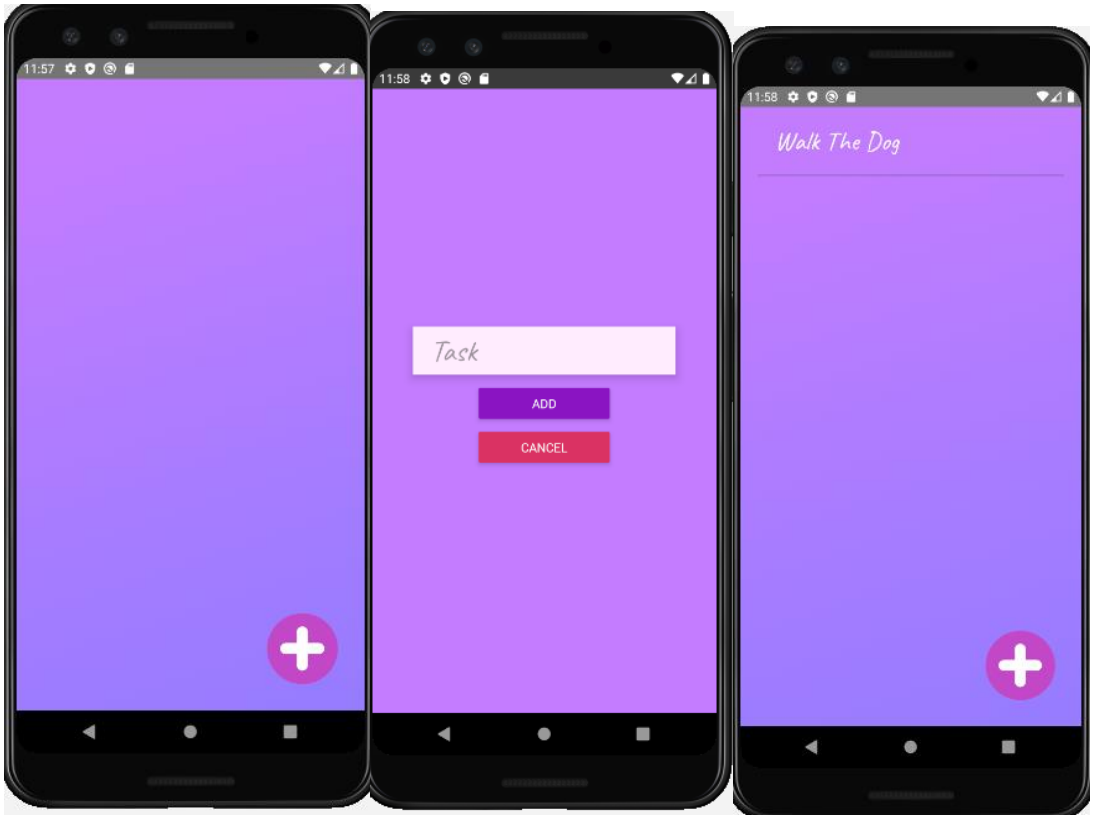
The code in TaskInput.js must be wrapped in the Modal tag. I also added an animationtype of slide, which makes the new page slide up onto the screen.

```
    return (
      <Modal visible={props.visible} animationType="slide">
          <View style={styles.inputSite}>
              {/*Input field for the task to be input*/}
              <TextInput
              style={styles.TaskInput}
              placeholder= "Task"
              onChangeText={taskInputHandler}
              value={enteredTask} />
              {/*Button which starts the fucntion addTaskhandler*/}
              <View style={styles.buttonContainer}>
                  <View style= {styles.button}>
                      <Button title="Add" onPress={addTaskHandler} color="#8A11C2" />
                  </View>
                  <View style= {styles.button}>
                      <Button title="Cancel" onPress={props.onCancel} color="#DA3164" />
                  </View>
              </View>
          </View>
      </Modal>

    );
};
```

At this point I also added a cancel button.

```
        onChangeText={taskInputHandler}
        value={enteredTask} />
        {/*Button which starts the fucntion addTaskhandler*/}
        <View style={styles.buttonContainer}>
            <View style= {styles.button}>
                <Button title="Add" onPress={addTaskHandler} color="#8A11C2" />
            </View>
            <View style= {styles.button}>
                <Button title="Cancel" onPress={props.onCancel} color="#DA3164" />
            </View>
        </View>
    </View>
```

This is how the app looks like at the moment. When the plus sign is pressed, it open a new page in which the task can be input. After pressing add, it will go back to the task list lpage and show the added task.



Step-3) Navigation of app

My navigation in the app will use tab navigation. It will have the pages: task list, settings, about, manual and contact.

To use navigation, some things need to be added to the project with commands:

- npm install @react-navigation/native
- npm install react-native-reanimated react-native-gesture-handler react-native-screens react-native-safe-area-context @react-native-community/masked-view
- npm install @react-navigation/bottom-tabs

I will be making all pages and their content in App.js but they can me also made into different files and imported as components.

I made a function for each page.

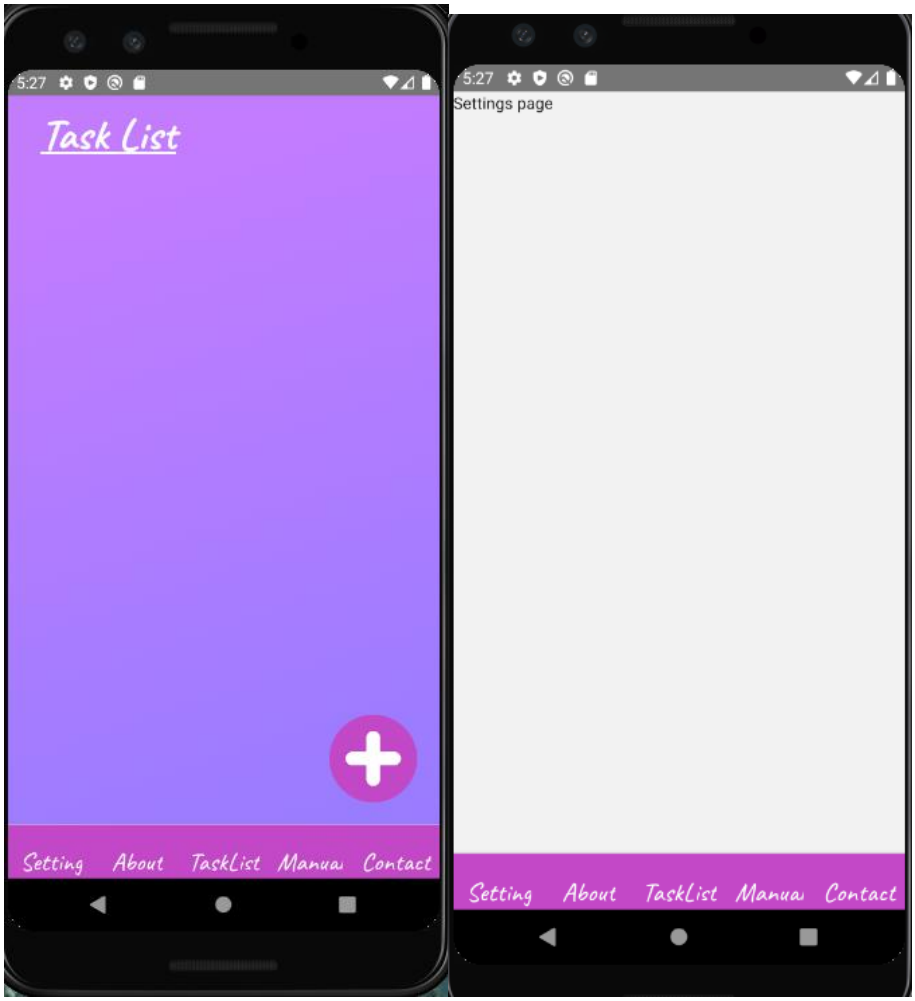
```
// Task page
function TaskList() {···
}
// Setting page
function Settings() {
  return (
    <View>
      <Text>Settings page</Text>
    </View>
  )
}
// About page
function About(){
  return (
    <View>
      <Text>About page</Text>
    </View>
  )
}
function Manual(){
  return (
    <View>
      <Text>Manual Page</Text>
    </View>
  )
}
function Contact(){
  return(
    <View>
      <Text>Contact page</Text>
    </View>
  )
}
```

I moved the code from const App into function TaskList.

```
// Task page
function TaskList() {

  // empty array for tasks
  const [tasks, setTasks] = useState([]);
  // a state for if a task is being added or not
  const [isAddMode, setIsAddMode] = useState(false);

  // function that executed when add button is pressed
  const addTaskHandler = taskTitle => {
    setTasks(currentTasks => [
    ...tasks,
    {id: Math.random().toString(), value: taskTitle }
    ]);
    setIsAddMode(false);
  };
  // function for cancel button
  const cancelAddTaskHandler = () => {
    setIsAddMode(false);
  }

  // function which removes task
  // removes only when item is pressed for longer - tapping does not delete it
  const removeTaskHandler = taskId => {
    setTasks(currentTasks => {
          return currentTasks.filter((task) => task.id !== taskId);
    })}
  return (
    <>
    <View style={styles.MainContainer}>
      {/*Background for app*/}
      <Image
        style={styles.backgroundImage}
        source={require('./images/AppBackground.png')}></Image>
        <Text style={styles.header}>Task List</Text>
        <TaskInput visible={isAddMode} onAddTask={addTaskHandler} onCancel={cancelAddTaskHandler}/>
        <FlatList
        keyExtractor={(item, index) => item.id}
          data={tasks}
          renderItem={itemData => <TaskItem  onDelete={removeTaskHandler.bind(this, itemData.item.id)} title=
        />
        {/*TouchableOpacity used as a button here*/}
        {/*Setting setIsAddMode to true to open modal where task can be input*/}
        <TouchableOpacity
          onPress={() => setIsAddMode(true)}
          style={styles.roundAddButton}>
            <Image
              style={styles.addButtonImage}
```

I made the navigation into the const App function as seen below

```
const App: () => React$Node = () => {


  return (
    <NavigationContainer>
      <Tab.Navigator
        initialRouteName="TaskList"
        tabBarOptions={{
          tabStyle: {
            backgroundColor: "#c349c7"
          },
          labelStyle: {
            fontSize: 23,
            color: "white",
            fontFamily: "Caveat-Regular",
          }
        }}
        >
        <Tab.Screen name="Setting" component={Settings}/>
        <Tab.Screen name="About" component={About}/>
        <Tab.Screen name="TaskList" component={TaskList}/>
        <Tab.Screen name="Manual" component={Manual}/>
        <Tab.Screen name="Contact" component={Contact}/>
      </Tab.Navigator>
    </NavigationContainer>
  )
}
```



Navigation in app works now, so I can work on the pages content.

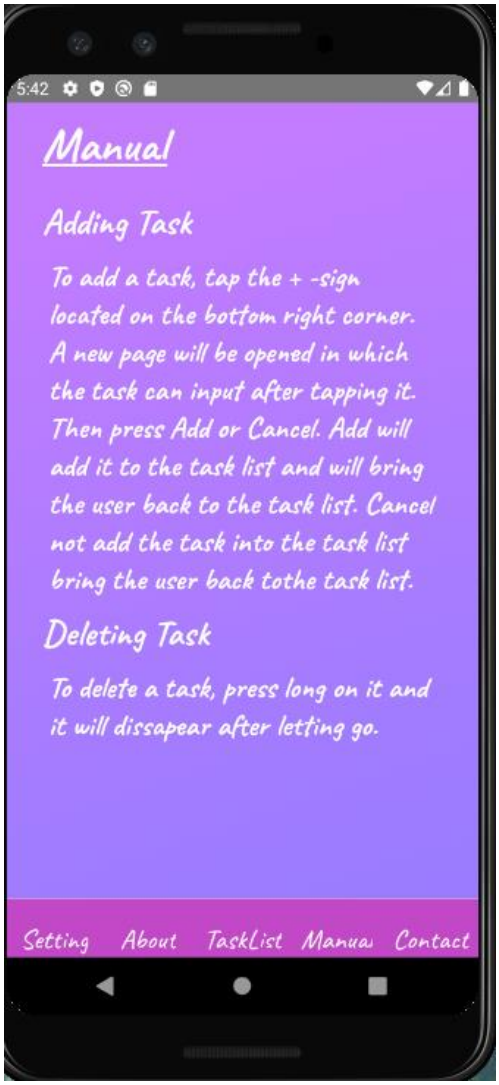
Step-4) Content of pages

All pages use the same StyleSheet so be careful which names are used.

First, I am going to write the manual.

```
function Manual(){
  return (
    <View style={styles.MainContainer}>
    {/*Background for app*/}
    <Image
      style={styles.backgroundImage}
      source={require('./images/AppBackground.png')}></Image>
      <View>
        <Text style={styles.header}>Manual</Text>
        <Text style={styles.header2}>Adding Task</Text>
        <Text style={styles.text}>To add a task, tap the + -sign located on the bottom right corner.
          A new page will be opened in which the task can input after tapping it. Then press Add or Cancel. Add
          to the task list and will bring the user back to the task list. Cancel not add the task into the task
        <Text style={styles.header2}>Deleting Task</Text>
        <Text style={styles.text}>To delete a task, press long on it and it will dissapear after letting go.</T
      </View>
    </View>
  )
}
```

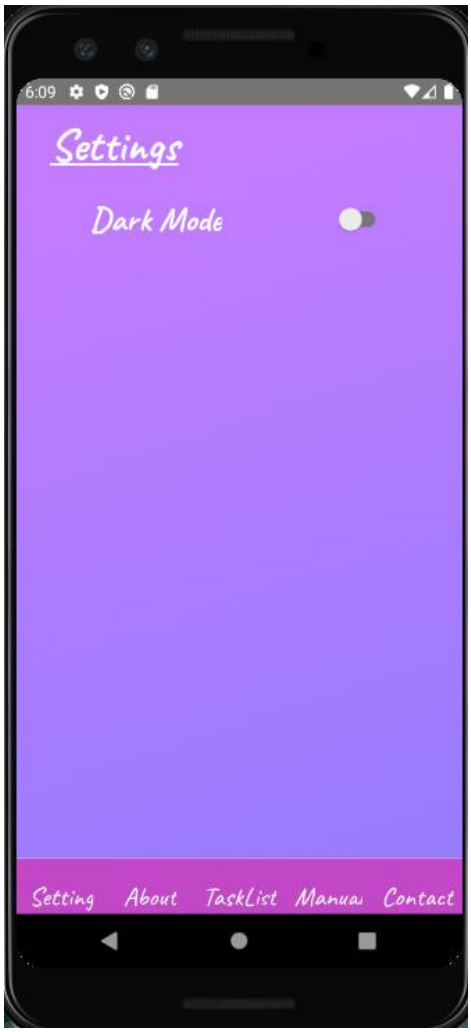
```
},
header: {
  flexDirection: "row",
  color: "white",
  fontFamily: "Caveat-Bold",
  fontSize: 40,
  paddingVertical: 10,
  paddingHorizontal: 30,
  textDecorationLine: "underline"
},
header2: {
  flexDirection: "row",
  color: "white",
  fontFamily: "Caveat-Bold",
  fontSize: 30,
  paddingVertical: 10,
  paddingHorizontal: 30
},
text: {
  flexDirection: "row",
  color: "white",
  fontFamily: "Caveat-Bold",
  fontSize: 25,
  paddingHorizontal: 35
},
```



I will be looking into changing the font to be easier to read. If the text would exceed over the seen page, one must remember to add a scrollview or a flatlist because scrolling is by default not allowed.

Next, I will be doing the settings page which will at this point only have a dark mode option.

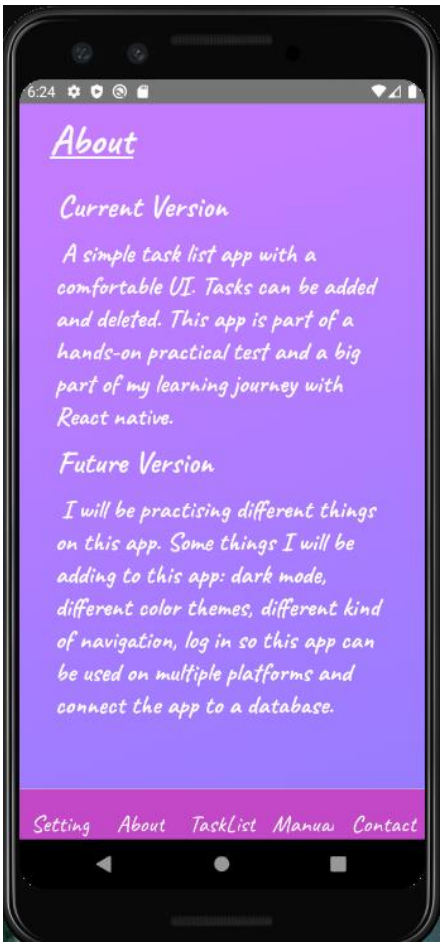
```
// Setting page
function Settings() {
  return (
    <View style={styles.MainContainer}>
    {/*Background for app*/}
    <Image
      style={styles.backgroundImage}
      source={require('./images/AppBackground.png')}></Image>
      <View>
        <Text style={styles.header}>Settings</Text>
        <View style={styles.setting}>
          <Text style={styles.settingstext}>Dark Mode</Text>
          <Switch
            trackColor= {{ false: "#767577", true: "#81b0ff"}} />
        </View>
      </View>
    </View>
  )
}
```



This dark mode is not yet functional.

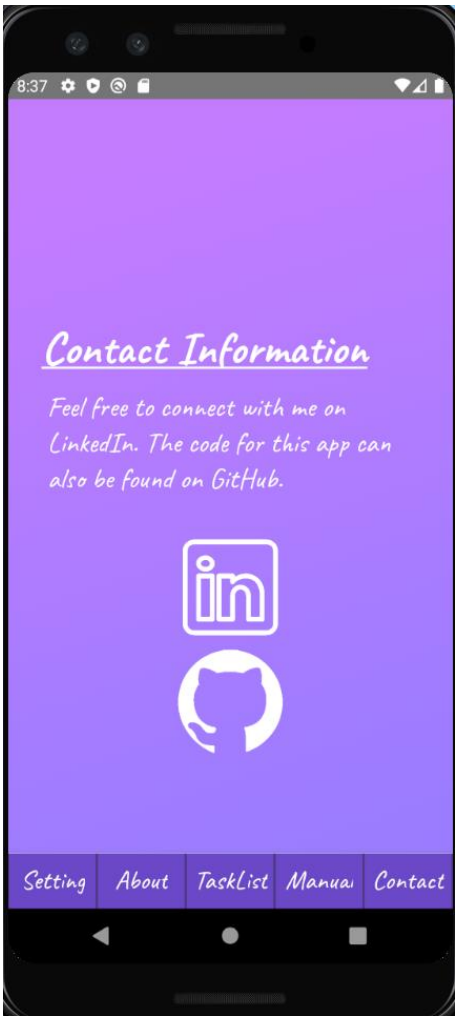
Next I will be doing the “about page. In the about page, I will write about why I did the app and what I will be doing with it in the future.

```
function About(){
  return (
    <View style={styles.MainContainer}>
    {/*Background for app*/}
    <Image
      style={styles.backgroundImage}
      source={require('./images/AppBackground.png')}></Image>
      <View>
        <Text style={styles.header}>About</Text>
        <Text style={styles.header2}> Current Version</Text>
        <Text style={styles.text}> A simple task list app with a comfortable UI. Tasks can be added and deleted.
          This app is part of a hands-on practical test and a big part of my learning journey with React native.
        <Text style={styles.header2}> Future Version</Text>
        <Text style={styles.text}> I will be practising different things on this app. Some things I will be addi
        log in so this app can be used on multiple platforms and connect the app to a database.</Text>
      </View>
    </View>
  )
}
```



The last page is “contact” in which I will link to my social accounts like github and linkedIn.

```
function Contact(){
  return(
    <View style={styles.MainContainerContact}>
    {/*Background for app*/}
    <Image
      style={styles.backgroundImage}
      source={require('./images/AppBackground.png')}></Image>
      <View>
        <Text style={styles.header}>Contact Information</Text>
        <Text style={styles.text}>Feel free to connect with me on LinkedIn.
        The code for this app can also be found on GitHub.</Text>
        <View  style={styles.socials}>
          <TouchableOpacity style={styles.social2} onPress={() => Linking.openURL("https://www.linkedin.com/in/susanna-h%C3%A4m%C3%A4l%C3%A4inen-81557616b/")}>
            <Image style={styles.social} source={require('./images/linkedinwhite.png')} />
          </TouchableOpacity>
          <TouchableOpacity style={styles.social2} onPress={() => Linking.openURL("https://github.com/Lesuz/TaskListApp")}>
            <Image style={styles.social} source={require('./images/githubwhite.png')} />
          </TouchableOpacity>
        </View>
      </View>
    </View>
  )
}
```



The icons can be clicked and they will bring the user to the repository of this app and to my linkedin profile.

This is as far as I got with this app without it breaking.

Section-4) Publishing and evaluation

This app is not ready to publish. I would need to add some kind of storage before publishing it, so the tasks don’t disappear after shutting the app down. I do not think a log in is required for a working task list app.

Evaluation:

I feel like I learned some basic but not all in this study unit. I learned and designed own app by benchmarking how other similar apps looks like. I added a simple tab navigation because a tab navigation is easy to use and I added a short manual for the app if the user is unsure on how to use it. Even though some aspects of this app are similar to the labwork done previously, most of the code is not similar and I added new pages and navigation. I feel like I now somewhat understand the navigation of a react native app even though I still need some help with it.