

# Simple Note Application

**Student:** WASD Wijesundara

**Registration Number:** 2018ICTS96

**Date:** 12/12/2024

**Github –**

## Introduction

Simple note application that store notes using an array and React Native and Expo used to develop and its includes pages of Login page, Registration page, Home page, Add Note page, View Note page.

## Project Requirements

### Login Page

- A user-friendly login interface where users can enter their username and password.
- Use form validation techniques to ensure proper input.
- Redirect users to the Home Page upon successful login.

### Registration Page

- Allow new users to sign up by providing the necessary details.
- Validate form inputs and navigate back to the Login Page after registration.

### Home Page

- Display a welcome message with the user's name after a successful login.
- Show a list of notes specific to the logged-in user with options to view notes.
- Provide a button to add a new note.

### Add Note Page

- Allow users to create and save new notes with a title and content.

### View Note Page

- Display the full content of a selected note.

# Development Process

## Initial Setup

Creating the project using `npx create-expo-app@latest`

```
Windows PowerShell
npm warn deprecated sudo-prompt@9.1.1: Package no longer supported. Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated domexception@4.0.0: Use your platform's native DOMException instead
npm warn deprecated sudo-prompt@8.2.5: Package no longer supported. Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated @xmldom/xmldom@0.7.13: this version is no longer supported, please update to at least 0.8.*

added 1178 packages, and audited 1179 packages in 4m

105 packages are looking for funding
  run `npm fund` for details

5 low severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

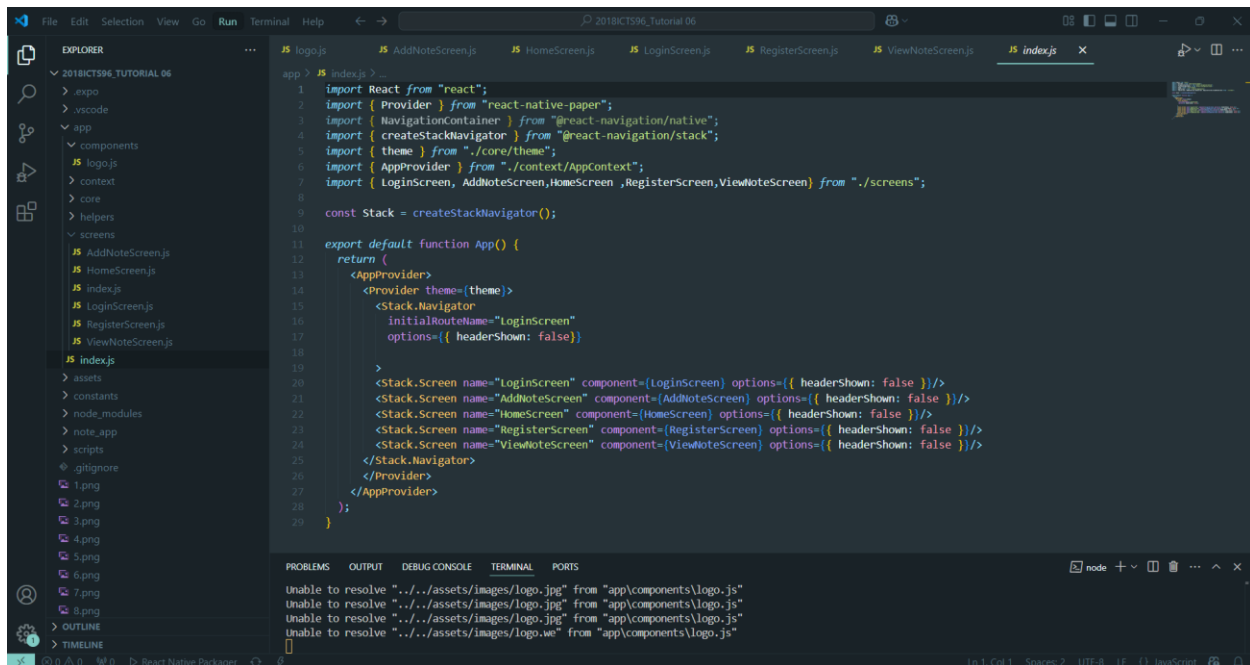
Run `npm audit` for details.

✔ Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

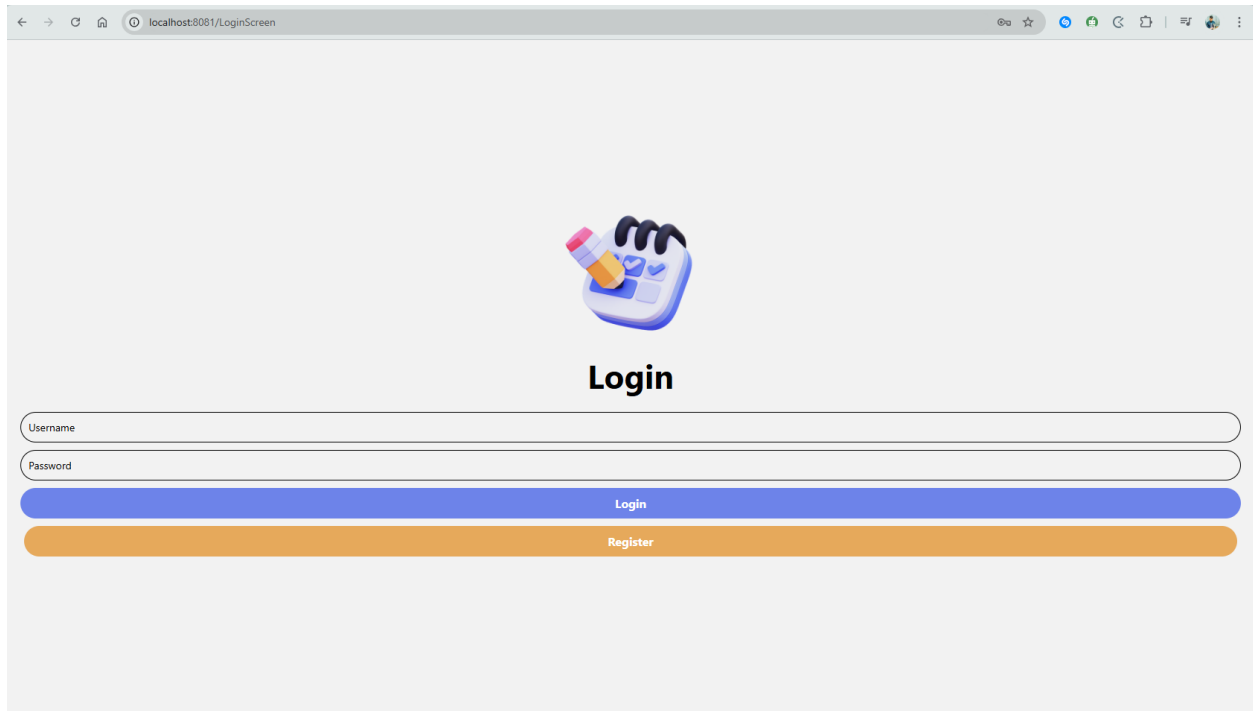
- cd Tutorial06
- npm run android
- npm run ios # you need to use macOS to build the iOS project - use the Expo app if you need to do iOS development without a Mac
- npm run web
PS E:\4 Year\Second Semester\TICT4242 MAD (P)>
```

## File Structure




# Application Screenshots

## Login Page



A screenshot of a web browser displaying the login page. The browser's address bar shows 'localhost:8081/LoginScreen'. The page has a light gray background. In the center, there is a 3D illustration of a hand with a blue sleeve holding a yellow and orange card. Below the illustration, the word 'Login' is written in a bold, black font. Underneath, there are two input fields: 'Username' and 'Password'. Below these fields are two buttons: a blue 'Login' button and an orange 'Register' button.

localhost:8081/LoginScreen



**Login**

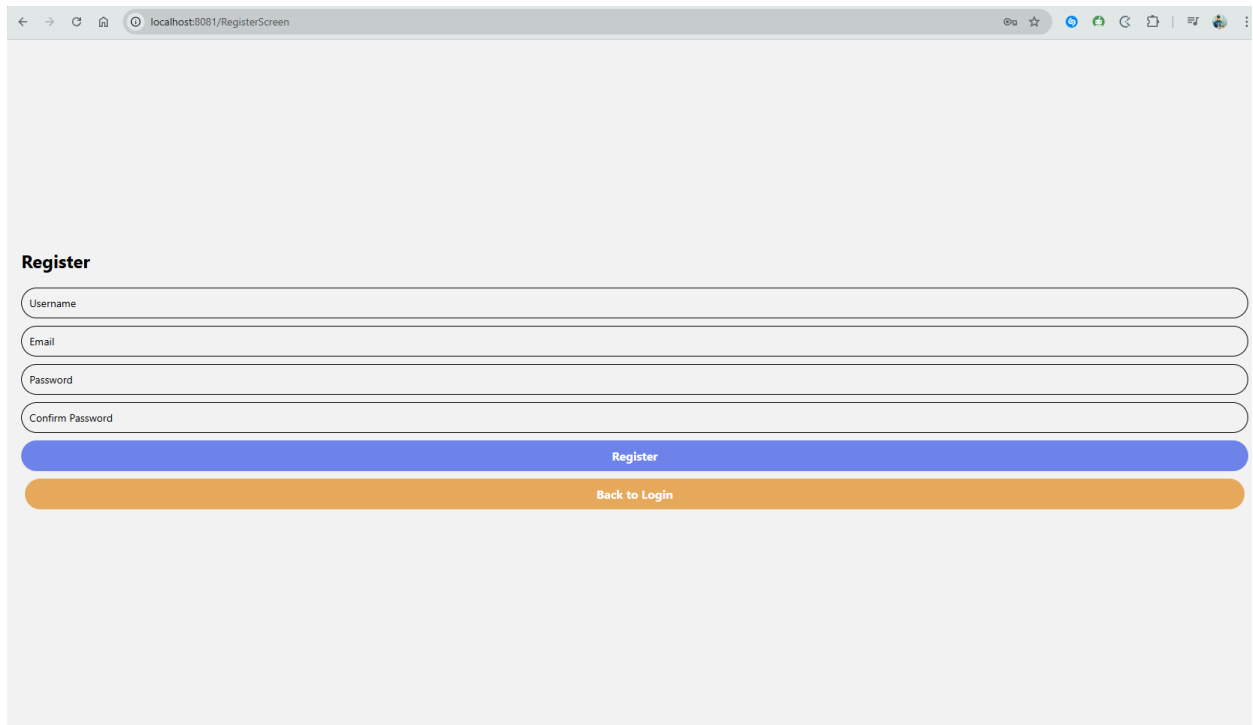
Username

Password

Login

Register

## Registration Page



A screenshot of a web browser displaying the registration page. The browser's address bar shows 'localhost:8081/RegisterScreen'. The page has a light gray background. The word 'Register' is written in a bold, black font. Below it, there are four input fields: 'Username', 'Email', 'Password', and 'Confirm Password'. At the bottom, there are two buttons: a blue 'Register' button and an orange 'Back to Login' button.

localhost:8081/RegisterScreen

**Register**

Username

Email

Password

Confirm Password

Register

Back to Login

## Home Page

← → ↺ 🏠

localhost:8081/HomeScreen

🔍 ☆ 🌐 🌱 ↶ 🗂 📄 ⬇ 🧑

Welcome, Guest!

Rugby Practise

Wends day @ 4.30

Delete

Mobile Application Development

Tutorial 6

Delete

E-Commerce

ICAE 02 on friday at 4.30pm

Delete

Fitness Training

Sunday 5.30am

Delete

Add Note

Log Out

## Add Note Page

← → ↺ 🏠

localhost:8081/AddNoteScreen

🔍 ☆ 🌐 🌱 ↶ 🗂 📄 ⬇ 🧑

Add Note

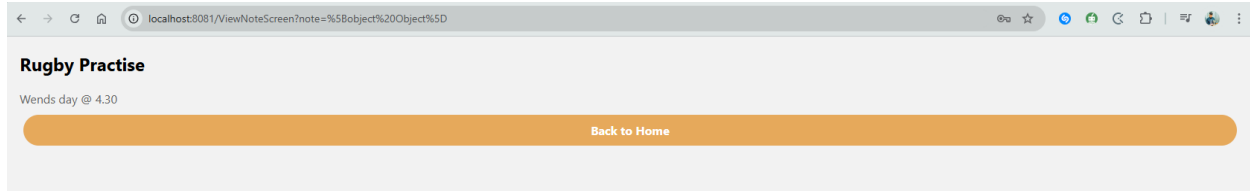
Title

Content

Save Note

Back to Home

## View Note Page



## Code Sections

**Context API Implementation** - Context API is used to manage application state for users and notes.

```
import React, { createContext, useState, useContext } from "react";

const AppContext = createContext();

export const AppProvider = ({ children }) => {
  const [users, setUsers] = useState([
    { username: "pasan@123", email: "john@example.com", password: "12345",
      name : "Pasan Madhuranga" },
  ]);
  const [user, setUser] = useState(null);
  const [notes, setNotes] = useState([]);
  const login = (username, password) => {
    const existingUser = users.find((u) => u.username === username &&
    u.password === password);
    if (existingUser) {
      setUser(existingUser);
      return true;
    }
    return false;
  };

  const register = (newUser) => {
    const exists = users.some((u) => u.username === newUser.username ||
    u.email === newUser.email);
    if (!exists) {
      setUsers((prev) => [...prev, newUser]);
      return true;
    }
    return false;
  };
};
```

```

const logout = () => setUser(null)
const addNote = (note) => {
  setNotes((prevNotes) => [...prevNotes, note]);
};
const removeNote = (noteId) => {
  setNotes((prevNotes) => prevNotes.filter((note) => note.id !== noteId));
};

return (
  <AppContext.Provider
    value={{
      user,
      login,
      logout,
      register,
      addNote,
      removeNote,
      notes,
      setUser,
    }}>
    {children}
  </AppContext.Provider>
);
};

export const useAppContext = () => useContext(AppContext);

```

**Form Validation Techniques** - form validation ensures input integrity.

```

export const validateLogin = ({ username, password }) => {
  const errors = {};
  if (!username.trim()) errors.username = "Username is required.";
  if (!password.trim()) errors.password = "Password is required.";
  return errors;
};

export const validateRegister = ({ username, email, password, confirmPassword })
=> {
  const errors = {};
  if (!username.trim()) errors.username = "Username is required.";

```

```

    if (!email.trim() || !email.includes("@")) errors.email = "Valid email is required.";
    if (!password.trim()) errors.password = "Password is required.";
    if (password !== confirmPassword) errors.confirmPassword = "Passwords must match.";
    return errors;
  };

export const validateNote = ({ title, content }) => {
  const errors = {};
  if (!title.trim()) errors.title = "Title is required.";
  if (!content.trim()) errors.content = "Content is required.";
  return errors;
};

```

## Note Management

### Add Note

```

import React, { useState } from "react";
import { View, Text, TextInput, Button, StyleSheet, TouchableOpacity } from "react-native";
import { useAppContext } from "../context/AppContext";
import { validateNote } from "../helpers/validation";

export default function AddNoteScreen({ navigation }) {
  const { addNote } = useAppContext() || {};
  if (!addNote) {
    console.log("addNote function is not available");
  }

  console.log("addNote:", addNote);
  const [form, setForm] = useState({ title: "", content: "" });
  const [errors, setErrors] = useState({});

  const handleSubmit = () => {
    const validationErrors = validateNote(form);
    if (Object.keys(validationErrors).length > 0) {
      setErrors(validationErrors);
      return;
    }
  }

```

```

addNote({
  id: Date.now(),
  ...form,
});
navigation.goBack();
};

return (
  <View style={styles.container}>
    <Text style={styles.title}>Add Note</Text>
    <TextInput
      style={styles.input}
      placeholder="Title"
      value={form.title}
      onChangeText={(text) => setForm({ ...form, title: text })}
    />
    {errors.title && <Text style={styles.error}>{errors.title}</Text>}

    <TextInput
      style={[styles.input, styles.textArea]}
      placeholder="Content"
      value={form.content}
      multiline
      numberOfLines={5}
      onChangeText={(text) => setForm({ ...form, content: text })}
    />
    {errors.content && <Text style={styles.error}>{errors.content}</Text>}

    <View style={styles.btnwrap}>
      <TouchableOpacity
        style={styles.customButton1}
        onPress={handleSubmit}
      >
        <Text style={styles.buttonText}>Save Note</Text>
      </TouchableOpacity>
    </View>

    <View style={styles.btnwrap}>
      <TouchableOpacity
        style={styles.customButton2}
        onPress={() => navigation.navigate("HomeScreen")}
      >
        <Text style={styles.buttonText}>Back to Home</Text>
      </TouchableOpacity>
    </View>
  </View>
);

```



```

    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: 20 },
  title: { fontSize: 24, fontWeight: "bold", marginBottom: 20 },
  input: { borderWidth: 1, padding: 10, marginBottom: 10, borderRadius: 5 },
  textArea: { height: 100, textAlignVertical: "top" },
  error: { color: "red", marginBottom: 10 },
  btnwrap: {marginBottom: 10,marginTop: 10,},
  buttonText: {
    color: "white",
    fontSize: 16,
    fontWeight: "bold",
  },
  customButton1: {
    backgroundColor: "#83ab6c",
    paddingVertical: 10,
    paddingHorizontal: 20,
    borderRadius: 30,
    alignItems: "center",
  },
  customButton2: {
    backgroundColor: "gray",
    paddingVertical: 10,
    paddingHorizontal: 20,
    borderRadius: 30,
    alignItems: "center",
    flex: 1,
    marginHorizontal: 5,
  },
});

```

## View Note

```

import React from "react";
import { View, Text, StyleSheet, TouchableOpacity } from "react-native";

export default function ViewNoteScreen({ route, navigation }) {
  const { note } = route.params;

```

```

if (!note) {
  return (
    <View style={styles.container}>
      <Text>No note found.</Text>
    </View>
  );
}

return (
  <View style={styles.container}>
    <Text style={styles.title}>{note.title}</Text>
    <Text style={styles.content}>{note.content}</Text>
    <View style={styles.btnwrap}>
      <TouchableOpacity
        style={styles.customButton2}
        onPress={() => navigation.navigate("HomeScreen")}
      >
        <Text style={styles.buttonText}>Back to Home</Text>
      </TouchableOpacity>
    </View>
  </View>
);
}

const styles = StyleSheet.create({
  container: { flex: 1, padding: 20 },
  title: { fontSize: 24, fontWeight: "bold", marginBottom: 20 },
  content: { fontSize: 16, color: "#666", lineHeight: 22 },
  btnwrap: {marginTop: 10},
  customButton2: {
    backgroundColor: "gray",
    paddingVertical: 10,
    paddingHorizontal: 20,
    borderRadius: 30,
    alignItems: "center",
    flex: 1,
    marginHorizontal: 5,
  },
  buttonText: {
    color: "white",
    fontSize: 16,
    fontWeight: "bold",
  },
});

```

## Navigation

```
export { default as HomeScreen } from './HomeScreen'
export { default as AddNoteScreen } from './AddNoteScreen'
export { default as LoginScreen } from './LoginScreen'
export { default as RegisterScreen } from './RegisterScreen'
export { default as ViewNoteScreen } from './ViewNoteScreen'
import React from "react";
import { Provider } from "react-native-paper";
import { NavigationContainer } from "@react-navigation/native";
import { createStackNavigator } from "@react-navigation/stack";
import { theme } from "../core/theme";
import { AppProvider } from "../context/AppContext";
import { LoginScreen, AddNoteScreen, HomeScreen, RegisterScreen, ViewNoteScreen }
from "../screens";

const Stack = createStackNavigator();

export default function App() {
  return (
    <AppProvider>
      <Provider theme={theme}>
        <Stack.Navigator
          initialRouteName="LoginScreen"
          options={{ headerShown: false }}
        >
          <Stack.Screen name="LoginScreen" component={LoginScreen} options={{
headerShown: false }}/>
          <Stack.Screen name="AddNoteScreen" component={AddNoteScreen} options={{
headerShown: false }}/>
          <Stack.Screen name="HomeScreen" component={HomeScreen} options={{
headerShown: false }}/>
          <Stack.Screen name="RegisterScreen" component={RegisterScreen} options={{
headerShown: false }}/>
          <Stack.Screen name="ViewNoteScreen" component={ViewNoteScreen} options={{
headerShown: false }}/>
        </Stack.Navigator>
      </Provider>
    </AppProvider>
  );
}
```

## Features

- Navigation
- Form Validation
- Context API
- Array-based Data Management

## Application Screens

- **Login Page** - Include validation errors and successful login behavior.
- **Registration Page** - Include validation errors and successful registration behavior.
- **Home Page** - Include notes list, add note button, and logout option.
- **Add Note Page** - Include form validation and note-saving behavior.
- **View Note Page** - Include full content of a selected note.

## References

[Pinterestin.pinterest.com](https://pinterestin.pinterest.com)

[Stack Overflowstackoverflow.com](https://stackoverflow.com)

[Mediummattclaffey.medium.com](https://mediummattclaffey.medium.com)

<https://legacy.reactjs.org/docs/context.html>