****

**การพัฒนาเว็ปแอปพลิเคชั่น**

**เรื่อง การพัฒนาโมบายแอปพลิเคชันการจัดการเวลา Chronobreak**

**จัดทำโดย**

**นางสาว เอสเธอร์ เหล่าสันติพลวุฒิ 003**

**นาย ปฏิวัฒน์ กฤษฏิ์สุภารัตน์ 031**

**นาย อังคสิทธิ์ การุณชาติ 035**

**นาย ทรรศนัย สวยล้ำ 037**

**เสนอ**

**ผศ.ดร. เสถียร จันทร์ปลา**

**รหัสวิชา CSD3201**

**ภาคเรียนที่ 2**

**คณะวิทยาศาสตร์และเทคโนโลยี**

**มหาวิทยาลัยราชภัฏสวนสุนันทา 2567**

**บทนำ**

ในยุคดิจิทัลปัจจุบัน สมาร์ทโฟนกลายเป็นอุปกรณ์ที่ขาดไม่ได้ในชีวิตประจำวันของผู้คนการจัดการเวลาอย่างมีประสิทธิภาพจึงเป็นสิ่งสำคัญ แอปพลิเคชันมือถือที่ช่วยในการจัดการเวลาได้รับความนิยมอย่างแพร่หลายเนื่องจากสามารถช่วยให้ผู้ใช้วางแผนและติดตามกิจกรรมต่าง ๆ ได้อย่างสะดวกสบายแม้ว่าแอปพลิเคชันจัดการเวลาจะมีอยู่มากมาย แต่หลายแอปพลิเคชันยังขาดความยืดหยุ่นและความสามารถในการปรับแต่งให้ตรงกับความต้องการเฉพาะของผู้ใช้ นอกจากนี้ การซิงโครไนซ์ข้อมูลระหว่างอุปกรณ์หรือการสำรองข้อมูลยังเป็นปัญหาที่พบได้บ่อย ผู้ใช้คาดหวังว่าแอปพลิเคชันจะสามารถตอบสนองต่อความต้องการเฉพาะของตนได้ แต่ในความเป็นจริง แอปพลิเคชันที่มีอยู่ยังไม่สามารถตอบสนองได้อย่างเต็มที่

โครงงานนี้มีวัตถุประสงค์ในการพัฒนาแอปพลิเคชันมือถือที่ชื่อว่า "Chronobreak" โดยใช้ React Native ซึ่งเป็นเฟรมเวิร์กที่ช่วยในการพัฒนาแอปพลิเคชันข้ามแพลตฟอร์ม ทำให้สามารถใช้งานได้ทั้งบนระบบปฏิบัติการ iOS และ Android นอกจากนี้ ยังใช้ฐานข้อมูล MySQL สำหรับการจัดเก็บและจัดการข้อมูลของผู้ใช้อย่างมีประสิทธิภาพ การผสานรวมเทคโนโลยีเหล่านี้จะช่วยให้แอปพลิเคชันมีความยืดหยุ่น ปรับแต่งได้ตามความต้องการของผู้ใช้ และสามารถซิงโครไนซ์ข้อมูลระหว่างอุปกรณ์ได้อย่างราบรื่น

การพัฒนาแอปพลิเคชัน "Chronobreak" อ้างอิงจากทฤษฎีการพัฒนาซอฟต์แวร์แบบ Agile ซึ่งเน้นการพัฒนาเป็นขั้นตอน มีการทดสอบและปรับปรุงอย่างต่อเนื่อง เพื่อให้ได้ผลิตภัณฑ์ที่ตรงกับความต้องการของผู้ใช้มากที่สุด นอกจากนี้ ยังใช้แนวคิดของการออกแบบที่มุ่งเน้นผู้ใช้ (User-Centered Design) เพื่อให้แน่ใจว่าแอปพลิเคชันมีอินเทอร์เฟซที่ใช้งานง่าย และตอบสนองต่อประสบการณ์ของผู้ใช้อย่างแท้จริง

**วัตถุประสงค์ของการวิจัยและความสำคัญของการวิจัย**

วัตถุประสงค์หลักของโครงงานนี้คือการพัฒนาแอปพลิเคชันจัดการเวลาที่มีความยืดหยุ่น ปรับแต่งได้ตามความต้องการของผู้ใช้ และสามารถซิงโครไนซ์ข้อมูลระหว่างอุปกรณ์ได้อย่างมีประสิทธิภาพ การวิจัยนี้มีความสำคัญเนื่องจากจะช่วยเติมเต็มช่องว่างที่มีอยู่ในแอปพลิเคชันจัดการเวลาปัจจุบัน และมอบประสบการณ์การใช้งานที่ดียิ่งขึ้นให้กับผู้ใช้

**อ้างอิง**

* คุณากร กล้าอาษา และพัชระ กันทา. (2563). พัฒนาโมบายแอปพลิเคชันก้าวไกลทูเดย์. วารสารวิชาการ "การประยุกต์ใช้เทคโนโลยีสารสนเทศ", 1(1), 1-10 [e-research.siam.edu](https://e-research.siam.edu/kb/mobile-application-development/?utm_source=chatgpt.com)
* บัญชา ปะสีละเตสัง. (2563). พัฒนา Application ด้วย React และ React Native. กรุงเทพฯ: ซีเอ็ดยูเคชั่น [m.se-ed.com](https://m.se-ed.com/Detail/%E0%B8%9E%E0%B8%B1%E0%B8%92%E0%B8%99%E0%B8%B2-Application-%E0%B8%94%E0%B9%89%E0%B8%A7%E0%B8%A2-React-%E0%B9%81%E0%B8%A5%E0%B8%B0-React-Native/9786160843220?utm_source=chatgpt.com)
* Hajiheydari, N., & Ashkani, M. (2018). Mobile application user behavior in the developing countries: a survey in Iran. Information Systems, 77, 22–33 [ph02.tci-thaijo.org](https://ph02.tci-thaijo.org/index.php/tsujournal/article/download/240094/164669?utm_source=chatgpt.com)
* Nuanmeesri, S. (2019). Mobile application for the purpose of marketing, product distribution and location-based logistics for elderly farmers. Applied Computing and Informatics, 1–20 https://doi.org/10.1016/j.aci.2019.11.001. [ph02.tci-thaijo.org](https://ph02.tci-thaijo.org/index.php/tsujournal/article/download/240094/164669?utm_source=chatgpt.com)

**จุดประสงค์**

* ช่วยให้ผู้ใช้งานสามารถบริหารและจัดการเวลาได้อย่างมีประสิทธิภาพ
* เพื่อพัฒนาโมบายแอปพลิเคชันการจัดการเวลา
* เพิ่มความสะดวกในการดูเวลาและเปรียบเทียบเขตเวลาโลก (Time Zone)
* ลดข้อผิดพลาดที่อาจเกิดจากการคำนวณเวลาข้ามเขต

**การวิเคราะห์ความต้องการ (Requirement Analysis)**

เป้าหมายของแอป

* ช่วยให้ผู้ใช้สามารถ จัดการเวลา ได้อย่างมีประสิทธิภาพ
* เพิ่ม ประสิทธิภาพในการจัดการเวลา ของผู้ใช้
* รองรับการทำงาน ข้ามแพลตฟอร์ม (iOS และ Android)
* อินเทอร์เฟซใช้งานง่ายตามหลัก User-Centered Design

**ฟังก์ชันที่จำเป็นของแอปพลิเคชัน**

การจัดการเวลาและกิจกรรม

* สร้าง แก้ไข และลบกิจกรรม
* ดูเวลาโลก
* แปลงเวลาของเขตเวลาโลก
* นับเวลาถอยหลัง
* กำหนดเวลาแจ้งเตือน

การเชื่อมต่อกับฐานข้อมูล

* บันทึกข้อมูลกิจกรรมและเวลาของผู้ใช้ลงใน MySQL

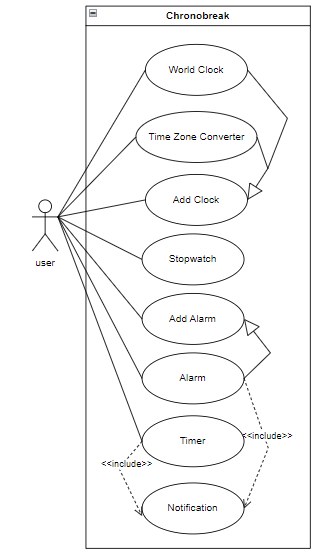
การแจ้งเตือนและเตือนความจำ

* ใช้ Push Notification เพื่อแจ้งเตือนกิจกรรม
* รองรับการตั้งค่าการแจ้งเตือนแบบปรับแต่งได้

ระบบวิเคราะห์พฤติกรรมการใช้เวลา

* สร้างรายงานการใช้เวลา
* แสดงข้อมูลสถิติการใช้เวลาในแต่ละหมวดหมู่

**Use Case Diagram**

****

**คำอธิบายภาพรวมของระบบ**

Frontend:

* พัฒนาโดยใช้ React Native รองรับทั้ง iOS และ Android
* ออกแบบ UI/UX ให้ใช้งานง่ายและตอบสนองได้ดี

Backend:

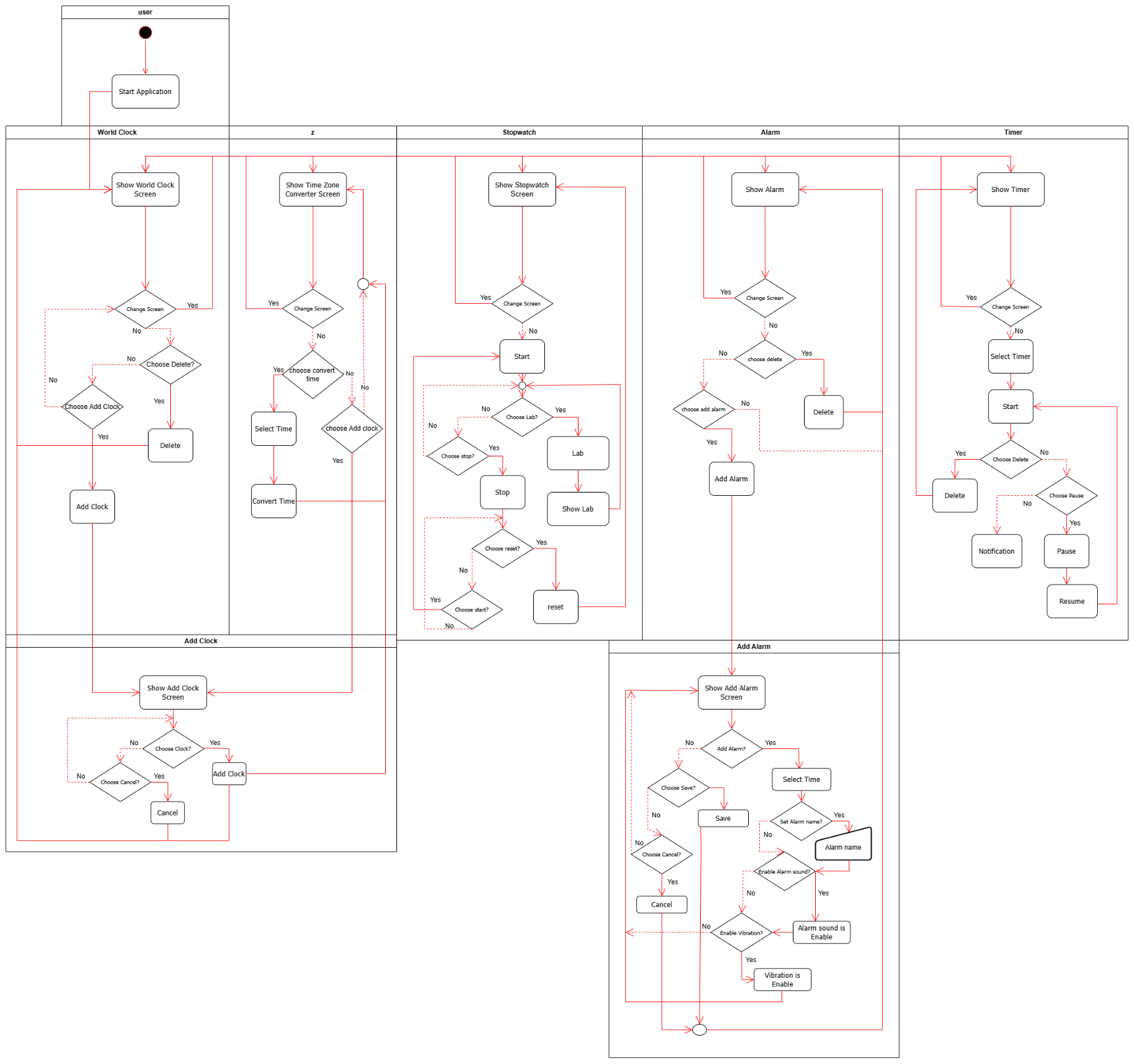
* ใช้ Node.js และ Express เป็นเซิร์ฟเวอร์ API
* จัดการการเชื่อมต่อกับฐานข้อมูล MySQL

Database:

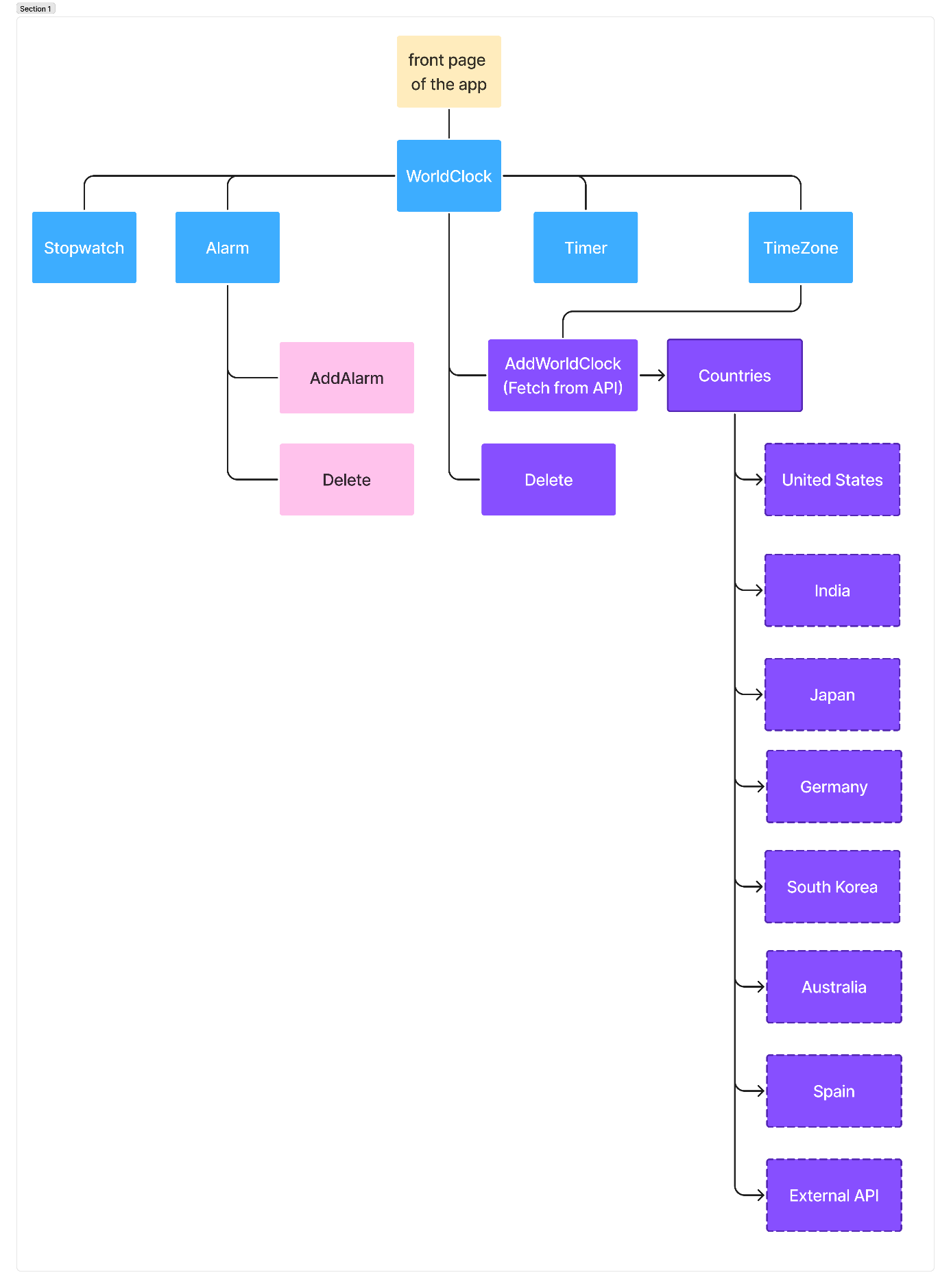
* ฐานข้อมูล MySQL ใช้จัดเก็บข้อมูลผู้ใช้และกิจกรรม
* มีโครงสร้างฐานข้อมูลที่รองรับการซิงโครไนซ์ข้อมูล

การทำงานโดยรวมของระบบ:

1. ผู้ใช้สามารถตั้งเวลานับถ่อยหลัง จับเวลา และตั้งนาฬิกาปลุกและการแจ้งเตือน
2. ผู้ใช้สามารถดูนาฬิกาโลก
3. ผู้ใช้สามารถเปลี่ยนเขตเวลา
4. ระบบแจ้งเตือนเตือนความจำกิจกรรมให้ผู้ใช้

**รายละเอียดของแต่ละฟังก์ชัน**

**ออกแบบโครงสร้างระบบโดยรวม**

****

**ออกแบบส่วนติดต่อผู้ใช้ (UI/UX Design)**

* Wireframe และ Mockup สำหรับหน้าต่าง ๆ
  + World Clock

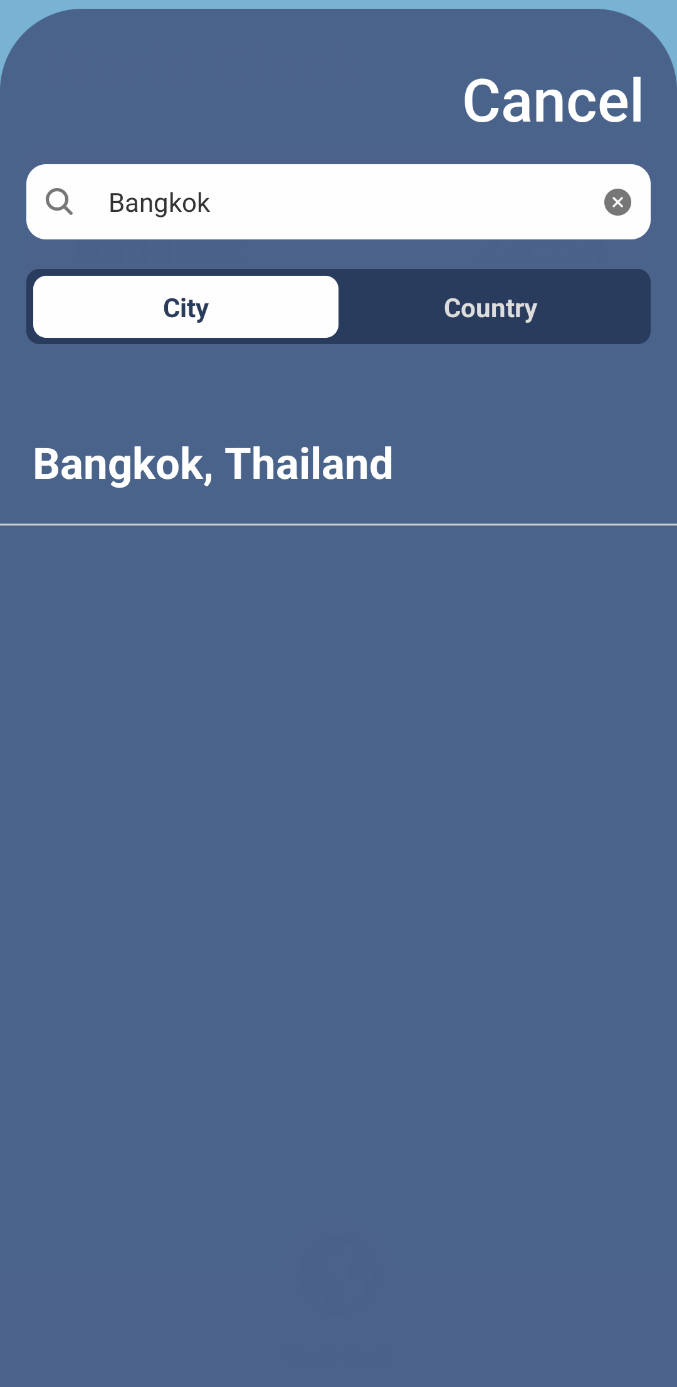
A screenshot of a phone

AI-generated content may be incorrect.

* + Add Clock



* + Search Add Clock

* + Delete Clock

A screenshot of a computer screen

AI-generated content may be incorrect.

* + Time Zone Converter

A screenshot of a phone

AI-generated content may be incorrect.

* + Timer

A screenshot of a phone

AI-generated content may be incorrect.

* + Start Timer, Resume Timer

A screenshot of a phone

AI-generated content may be incorrect.

* + Pause Timer, Delete Timer

A screenshot of a clock

AI-generated content may be incorrect.

* + Stopwatch, reset

A screenshot of a timer

AI-generated content may be incorrect.

* + Start/Stop

A screenshot of a phone

AI-generated content may be incorrect.

* + Record Lap

A screenshot of a timer

AI-generated content may be incorrect.

* + Alarm

A blue square with white dots

AI-generated content may be incorrect.

* + Add Alarm

A screenshot of a phone

AI-generated content may be incorrect.

* + Alarm Card

A blue screen with white text

AI-generated content may be incorrect.

* + Delete Alarm

A blue screen with red and white text

AI-generated content may be incorrect.

**ออกแบบโครงสร้างฐานข้อมูล (Database Design)**

ER Diagram:



Database Tables:

* **cities**

|  |  |
| --- | --- |
| id (PK) | integer |
| city | string |
| country | string |
| code | string |

Example:

* **cities**

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **city** | **country** | **code** |
| 1 | Bangkok | Thailand | TH |
| 2 | Tokyo | Japan | JP |
| 3 | London | United Kingdom | GB |

**การพัฒนาโปรแกรม**

|  |  |
| --- | --- |
| **ไฟล์** | **การทำงาน** |
| App.js | หน้าแรก |
| TimeUtils.js | ไฟล์จัดการเวลา |
| AddAlarm.js | หน้าจอตั้งนาฬิกาปลุก |
| AddClock.js | หน้าจอเพิ่มเวลาของเมืองที่จะดู |
| Alarm.js | หน้าจอนาฬิกาปลุก |
| Stopwatch.js | หน้าจอจับเวลา |
| Timer.js | หน้าจอตั้งเวลานับถอยหลัง |
| WorldClock.js | หน้าจอดูเวลาโลก |
| TimeZoneConverter.js | หน้าจอแปลงเวลา |
| AlarmNavigator.js | stack navigator ของ Alarm เชื่อมต่อกับ AddAlarm |
| MainNavigator.js | tab navigator เป็น navigator หลัก |
| TimeZoneNavigator.js | stack navigator ของ TimeZoneConverter เชื่อมต่อกับ AddClock |
| WorldClockNavigator.js | stack navigator ของ WorldClock เชื่อต่อกับ AddClock |
| AlarmCard.js | component card ใช้ใน Alarm |
| ClockCard.js | component card ใช้ใน WorldClock |
| SelectTimer.js | เลือกเวลาของ Timer |
| TimeSelector.js | เลือกเวลาของ TimeZoneConverter และ Alarm |
| Color.js | ค่าคงที่ของสี |

App.js

1. import { StatusBar } from "expo-status-bar";

2. import { NavigationContainer } from "@react-navigation/native";

3. import MainNavigator from "./navigator/MainNavigator";

4. // import { View } from 'react-native';

5.

6. // import styles from './styles/styles';

7.

8. export default function App() {

9. return (

10. <NavigationContainer>

11. <MainNavigator />

12. <StatusBar style="auto" />

13. </NavigationContainer>

14. );

15. }

16.

TimeUtils.js

1. import moment from "moment-timezone";

2. import \* as ct from "countries-and-timezones";

3.

4. /\*\*

5. \* Gets the current time for a given country's timezone.

6. \* @param {string} countryCode - The ISO country code.

7. \* @returns {string | null} The formatted time or null if the country is not found.

8. \*/

9. export function getTimeByCountry(countryCode) {

10. const country = ct.getCountry(countryCode);

11. if (!country || !country.timezones.length) return null;

12.

13. const timezone = country.timezones[0];

14. return moment.tz(timezone).format("HH:mm");

15. }

16.

17. /\*\*

18. \* Gets the current time for a given city's timezone.

19. \* @param {string} city - The city name.

20. \* @param {string} countryCode - The ISO country code.

21. \* @param {Date} [time] - The specific time to convert.

22. \* @returns {string | null} The formatted time or null if the city is not found.

23. \*/

24. export function getTimeByCity(city, countryCode, time) {

25. const country = ct.getCountry(countryCode);

26. if (!country || !country.timezones.length) return null;

27.

28. const timezone = country.timezones[0];

29. const momentTime = time ? moment(time) : moment();

30. return momentTime.tz(timezone).format("HH:mm");

31. }

32.

33. /\*\*

34. \* Gets the current time for all countries' timezones.

35. \* @returns {Record<string, string | null>} An object with country codes as keys and formatted times as values.

36. \*/

37. export function getAllCountriesTime() {

38. const countries = ct.getAllCountries();

39. const countryTimes = {};

40.

41. for (const countryCode in countries) {

42. countryTimes[countryCode] = getTimeByCountry(countryCode);

43. }

44.

45. return countryTimes;

46. }

47.

48. /\*\*

49. \* Gets the current time for all popular cities' timezones.

50. \* @param {Array<{ city: string, country: string }>} cities - The list of cities and their country codes.

51. \* @returns {Record<string, string | null>} An object with city names as keys and formatted times as values.

52. \*/

53. export function getAllCitiesTime(cities) {

54. const cityTimes = {};

55.

56. cities.forEach(({ city, country }) => {

57. cityTimes[city] = getTimeByCity(city, country);

58. });

59.

60. return cityTimes;

61. }

AddAlarm.js

1. import React, { useState } from "react";

2. import {

3. ScrollView,

4. TouchableOpacity,

5. Text,

6. View,

7. TextInput,

8. Switch,

9. } from "react-native";

10. import TimeSelector from "../components/TimeSelector";

11. import styles from "../styles/styles";

12.

13. export default function AddAlarm({ closeModal, saveAlarm }) {

14. const [time, setTime] = useState({ hour: 0, minute: 0 });

15. const [alarmName, setAlarmName] = useState("");

16. const [alarmSound, setAlarmSound] = useState(true);

17. const [vibration, setVibration] = useState(true);

18.

19. const handleSave = () => {

20. const formattedTime = `${time.hour

21. .toString()

22. .padStart(2, "0")}:${time.minute.toString().padStart(2, "0")}`;

23. saveAlarm(formattedTime, alarmName, alarmSound, vibration);

24. closeModal();

25. };

26.

27. const handleTimeChange = (hour, minute) => {

28. setTime({ hour, minute });

29. };

30.

31. return (

32. <ScrollView

33. style={[

34. styles.amcontainer,

35. {

36. backgroundColor: "#4A628A",

37. borderTopLeftRadius: 50,

38. borderTopRightRadius: 50,

39. paddingTop: 30,

40. },

41. ]}

42. >

43. <View

44. style={{

45. flexDirection: "row",

46. justifyContent: "space-between",

47. paddingHorizontal: 20,

48. }}

49. >

50. <TouchableOpacity

51. style={{ backgroundColor: "transparent" }}

52. onPress={closeModal}

53. >

54. <Text

55. style={{

56. fontSize: 36,

57. color: "#fff",

58. fontWeight: 500,

59. textAlign: "left",

60. }}

61. >

62. Cancel

63. </Text>

64. </TouchableOpacity>

65. <TouchableOpacity

66. style={{ backgroundColor: "transparent" }}

67. onPress={handleSave}

68. >

69. <Text

70. style={{

71. fontSize: 36,

72. color: "#fff",

73. fontWeight: 500,

74. textAlign: "right",

75. }}

76. >

77. Save

78. </Text>

79. </TouchableOpacity>

80. </View>

81. <View style={{ padding: 25 }}>

82. <TimeSelector onTimeChange={handleTimeChange} />

83. <View style={{ marginBottom: 50 }} />

84. <View style={styles.settingsContainer}>

85. <TextInput

86. style={styles.aminputTransparent}

87. placeholder="Alarm name"

88. placeholderTextColor="#B9B9B9"

89. value={alarmName}

90. onChangeText={setAlarmName}

91. />

92. <View style={styles.amseparator} />

93. <View style={styles.settingContainer}>

94. <Text style={[styles.settingText, { marginLeft: 10 }]}>

95. Alarm sound

96. </Text>

97. <Switch

98. value={alarmSound}

99. onValueChange={setAlarmSound}

100. thumbColor={alarmSound ? "#DCF5FC" : "#f4f3f4"}

101. trackColor={{ false: "#767577", true: "#0166EF" }}

102. />

103. </View>

104. <View style={styles.amseparator} />

105. <View style={styles.settingContainer}>

106. <Text style={[styles.settingText, { marginLeft: 10 }]}>

107. Vibration

108. </Text>

109. <Switch

110. value={vibration}

111. onValueChange={setVibration}

112. thumbColor={vibration ? "#DCF5FC" : "#f4f3f4"}

113. trackColor={{ false: "#767577", true: "#0166EF" }}

114. />

115. </View>

116. <View style={styles.amseparator} />

117. <View style={{ marginBottom: 15 }} />

118. </View>

119. </View>

120. </ScrollView>

121. );

122. }

AddClock.js

1. import React, { useState, useEffect } from "react";

2. import { ScrollView, TouchableOpacity, Text, View, TextInput } from "react-native";

3. import AsyncStorage from "@react-native-async-storage/async-storage";

4. import styles from "../styles/styles";

5. import { Ionicons } from "@expo/vector-icons";

6. import axios from 'axios';

7.

8. export default function AddClock({ navigation, closeModal, route }) {

9. const [addedCities, setAddedCities] = useState([]);

10. const [availableCities, setAvailableCities] = useState([]);

11. const [searchQuery, setSearchQuery] = useState('');

12. const [searchType, setSearchType] = useState('city'); // 'city' หรือ 'country'

13. const { fromScreen } = route.params;

14.

15. useEffect(() => {

16. const loadAddedCities = async () => {

17. const storedCities = await AsyncStorage.getItem("addedCities");

18. if (storedCities) {

19. setAddedCities(JSON.parse(storedCities));

20. }

21. };

22. loadAddedCities();

23. }, []);

24.

25. useEffect(() => {

26. const fetchCities = async () => {

27. try {

28. const response = await axios.get('http://10.0.2.2/chronobreak-backend/api.php');

29. const data = response.data;

30. setAvailableCities(data.filter(

31. (city) => !addedCities.some((addedCity) => addedCity.city === city.city)

32. ));

33. } catch (error) {

34. console.error('Error fetching cities:', error);

35. }

36. };

37. fetchCities();

38. }, [addedCities]);

39.

40. const addCity = async (city, country) => {

41. const newCities = [...addedCities, { city, country }];

42. await AsyncStorage.setItem("addedCities", JSON.stringify(newCities));

43. closeModal();

44. navigation.navigate(fromScreen, { city, country });

45. };

46.

47. // กรองรายการเมืองตามคำค้นหาและประเภทการค้นหา

48. const filteredCities = availableCities.filter(city => {

49. if (!searchQuery) return true;

50.

51. if (searchType === 'city') {

52. return city.city.toLowerCase().includes(searchQuery.toLowerCase());

53. } else { // searchType === 'country'

54. return city.name.toLowerCase().includes(searchQuery.toLowerCase());

55. }

56. });

57.

58. return (

59. <ScrollView

60. style={[

61. styles.amcontainer,

62. {

63. backgroundColor: "#4A628A",

64. borderTopLeftRadius: 50,

65. borderTopRightRadius: 50,

66. paddingTop: 30,

67. },

68. ]}

69. >

70. <View

71. style={{

72. flexDirection: "row",

73. justifyContent: "flex-end",

74. paddingHorizontal: 20,

75. }}

76. >

77. <TouchableOpacity

78. style={{ backgroundColor: "transparent" }}

79. onPress={closeModal}

80. >

81. <Text

82. style={{

83. fontSize: 36,

84. color: "#fff",

85. fontWeight: 500,

86. textAlign: "right",

87. }}

88. >

89. Cancel

90. </Text>

91. </TouchableOpacity>

92. </View>

93.

94. {/\* SearchBar และ Toggle Button \*/}

95. <View style={searchStyles.searchContainer}>

96. <View style={searchStyles.searchInputContainer}>

97. <Ionicons name="search" size={20} color="#777" style={searchStyles.searchIcon} />

98. <TextInput

99. style={searchStyles.searchInput}

100. placeholder={`Search by ${searchType === 'city' ? 'city' : 'country'}...`}

101. placeholderTextColor="#777"

102. value={searchQuery}

103. onChangeText={setSearchQuery}

104. />

105. {searchQuery ? (

106. <TouchableOpacity onPress={() => setSearchQuery('')}>

107. <Ionicons name="close-circle" size={20} color="#777" />

108. </TouchableOpacity>

109. ) : null}

110. </View>

111.

112. <View style={searchStyles.toggleContainer}>

113. <TouchableOpacity

114. style={[searchStyles.toggleButton, searchType === 'city' && searchStyles.activeToggle]}

115. onPress={() => setSearchType('city')}

116. >

117. <Text style={[searchStyles.toggleText, searchType === 'city' && searchStyles.activeToggleText]}>

118. City

119. </Text>

120. </TouchableOpacity>

121. <TouchableOpacity

122. style={[searchStyles.toggleButton, searchType === 'country' && searchStyles.activeToggle]}

123. onPress={() => setSearchType('country')}

124. >

125. <Text style={[searchStyles.toggleText, searchType === 'country' && searchStyles.activeToggleText]}>

126. Country

127. </Text>

128. </TouchableOpacity>

129. </View>

130. </View>

131.

132. <View style={{ marginBottom: 100 }}>

133. {filteredCities.map((city, index) => (

134. <TouchableOpacity

135. key={index}

136. style={styles.cityItem}

137. onPress={() => addCity(city.city, city.country)}

138. >

139. <Text

140. style={{

141. fontSize: 26,

142. fontWeight: "bold",

143. padding: 5,

144. color: "#fff",

145. }}

146. >

147. {city.city}, {city.name}

148. </Text>

149. </TouchableOpacity>

150. ))}

151. </View>

152. </ScrollView>

153. );

154. }

155.

156. // สไตล์สำหรับ SearchBar และ Toggle Button

157. const searchStyles = {

158. searchContainer: {

159. padding: 16,

160. marginBottom: 10,

161. },

162. searchInputContainer: {

163. backgroundColor: '#fff',

164. borderRadius: 12,

165. flexDirection: 'row',

166. alignItems: 'center',

167. paddingHorizontal: 10,

168. marginBottom: 10,

169. },

170. searchIcon: {

171. marginRight: 8,

172. },

173. searchInput: {

174. flex: 1,

175. padding: 12,

176. fontSize: 16,

177. color: '#333',

178. },

179. toggleContainer: {

180. flexDirection: 'row',

181. backgroundColor: '#2a3c5c',

182. borderRadius: 8,

183. marginVertical: 8,

184. padding: 4,

185. },

186. toggleButton: {

187. flex: 1,

188. paddingVertical: 8,

189. alignItems: 'center',

190. },

191. activeToggle: {

192. backgroundColor: '#fff',

193. borderRadius: 8,

194. },

195. toggleText: {

196. color: '#ddd',

197. fontWeight: 'bold',

198. fontSize: 16,

199. },

200. activeToggleText: {

201. color: '#2a3c5c',

202. }

203. };

204.

Alarm.js

1. import React, { useEffect, useState } from "react";

2. import { useNavigation, useIsFocused } from "@react-navigation/native";

3. import {

4. View,

5. Text,

6. ScrollView,

7. TouchableOpacity,

8. RefreshControl,

9. Modal,

10. Vibration,

11. } from "react-native";

12. import styles from "../styles/styles";

13. import { Ionicons } from "@expo/vector-icons";

14. import AsyncStorage from "@react-native-async-storage/async-storage";

15. import AddAlarm from "./AddAlarm";

16. import AlarmCard from "../components/AlarmCard";

17. import { Audio } from "expo-av";

18.

19. export default function Alarm() {

20. const [refreshing, setRefreshing] = useState(false);

21. const navigation = useNavigation();

22. const [alarms, setAlarms] = useState([]);

23. const isFocused = useIsFocused();

24. const [modalVisible, setModalVisible] = useState(false);

25. const [deletingAlarm, setDeletingAlarm] = useState(null);

26. const [showDeleteButtons, setShowDeleteButtons] = useState(false);

27. const [sound, setSound] = useState(null);

28.

29. useEffect(() => {

30. const refreshControl = async () => {

31. setRefreshing(true);

32. const storedAlarms = await AsyncStorage.getItem("alarms");

33. if (storedAlarms) {

34. setAlarms(JSON.parse(storedAlarms));

35. }

36. setRefreshing(false);

37. };

38. }, []);

39.

40. useEffect(() => {

41. const loadAlarms = async () => {

42. const storedAlarms = await AsyncStorage.getItem("alarms");

43. if (storedAlarms) {

44. setAlarms(JSON.parse(storedAlarms));

45. }

46. };

47. if (isFocused) {

48. loadAlarms();

49. }

50. }, [isFocused]);

51.

52. useEffect(() => {

53. let previousMinute = new Date().getMinutes();

54.

55. const checkAlarms = async () => {

56. const now = new Date();

57. const currentTime = `${now.getHours().toString().padStart(2, "0")}:${now

58. .getMinutes()

59. .toString()

60. .padStart(2, "0")}`;

61. const currentMinute = now.getMinutes();

62.

63. if (currentMinute !== previousMinute) {

64. const updatedAlarms = [...alarms];

65. for (const alarm of updatedAlarms) {

66. if (alarm.isEnabled && alarm.time === currentTime) {

67. alarm.isEnabled = false;

68. if (alarm.sound) {

69. await playSound();

70. }

71. if (alarm.vibration) {

72. Vibration.vibrate([1000, 1000, 1000, 1000]);

73. }

74. }

75. }

76. setAlarms(updatedAlarms);

77. await AsyncStorage.setItem("alarms", JSON.stringify(updatedAlarms));

78. previousMinute = currentMinute;

79. }

80. };

81.

82. const interval = setInterval(checkAlarms, 1000);

83. return () => clearInterval(interval);

84. }, [alarms]);

85.

86. const playSound = async () => {

87. const { sound } = await Audio.Sound.createAsync(

88. require("../assets/alarm-sound.mp3")

89. );

90. setSound(sound);

91. await sound.playAsync();

92. };

93.

94. const saveAlarm = async (time, name, sound, vibration) => {

95. const newAlarm = { time, name, sound, vibration, isEnabled: false };

96. const newAlarms = [...alarms, newAlarm];

97. setAlarms(newAlarms);

98. await AsyncStorage.setItem("alarms", JSON.stringify(newAlarms));

99. };

100.

101. const toggleDeleteButtons = () => {

102. setShowDeleteButtons(!showDeleteButtons);

103. setDeletingAlarm(null);

104. };

105.

106. const handleDeleteAlarm = async (time) => {

107. const newAlarms = alarms.filter((alarm) => alarm.time !== time);

108. await AsyncStorage.setItem("alarms", JSON.stringify(newAlarms));

109. setAlarms(newAlarms);

110. setDeletingAlarm(null);

111. setShowDeleteButtons(false);

112. };

113.

114. const toggleAlarm = async (time) => {

115. const newAlarms = alarms.map((alarm) =>

116. alarm.time === time ? { ...alarm, isEnabled: !alarm.isEnabled } : alarm

117. );

118. setAlarms(newAlarms);

119. await AsyncStorage.setItem("alarms", JSON.stringify(newAlarms));

120. };

121.

122. return (

123. <ScrollView

124. style={styles.container}

125. refreshControl={

126. <RefreshControl

127. refreshing={refreshing}

128. onRefresh={() => {

129. RefreshControl;

130. }}

131. />

132. }

133. >

134. <View style={styles.headerContainer}>

135. <Text style={styles.header}>Alarm</Text>

136. <View style={{ flexDirection: "row" }}>

137. <TouchableOpacity

138. style={styles.headerButton}

139. onPress={() => setModalVisible(true)}

140. >

141. <Ionicons

142. name="add-outline"

143. style={styles.headerButtonText}

144. ></Ionicons>

145. </TouchableOpacity>

146. <TouchableOpacity

147. style={styles.headerButton}

148. onPress={toggleDeleteButtons}

149. >

150. <Ionicons

151. name="trash-bin-outline"

152. style={styles.headerButtonText}

153. ></Ionicons>

154. </TouchableOpacity>

155. </View>

156. </View>

157. {alarms.map((alarm, index) => (

158. <AlarmCard

159. key={index}

160. time={alarm.time}

161. name={alarm.name}

162. sound={alarm.sound}

163. vibration={alarm.vibration}

164. deleting={deletingAlarm === alarm.time}

165. onConfirmDelete={() => handleDeleteAlarm(alarm.time)}

166. showDeleteButton={showDeleteButtons}

167. isEnabled={alarm.isEnabled}

168. toggleSwitch={() => toggleAlarm(alarm.time)}

169. />

170. ))}

171. <Modal

172. animationType="slide"

173. transparent={true}

174. visible={modalVisible}

175. onRequestClose={() => {

176. setModalVisible(!modalVisible);

177. }}

178. >

179. <AddAlarm

180. navigation={navigation}

181. closeModal={() => setModalVisible(false)}

182. saveAlarm={saveAlarm}

183. />

184. </Modal>

185. </ScrollView>

186. );

187. }

188.

Stopwatch.js

1. import React, { useState, useRef } from "react";

2. import { View, Text, TouchableOpacity, FlatList } from "react-native";

3.

4. import styles from "../styles/styles";

5.

6. export default function Stopwatch() {

7. const [time, setTime] = useState(0);

8. const [isRunning, setIsRunning] = useState(false);

9. const [laps, setLaps] = useState([]);

10. const timerRef = useRef(null);

11.

12. const startStop = () => {

13. if (isRunning) {

14. if (timerRef.current !== null) {

15. clearInterval(timerRef.current);

16. }

17. } else {

18. const startTime = Date.now() - time;

19. timerRef.current = setInterval(() => {

20. setTime(Date.now() - startTime);

21. }, 10);

22. }

23. setIsRunning(!isRunning);

24. };

25.

26. const reset = () => {

27. if (timerRef.current !== null) {

28. clearInterval(timerRef.current);

29. }

30. setTime(0);

31. setLaps([]);

32. setIsRunning(false);

33. };

34.

35. const recordLap = () => {

36. if (isRunning) {

37. setLaps([...laps, time]);

38. }

39. };

40.

41. const formatTime = (milliseconds) => {

42. const minutes = Math.floor(milliseconds / 60000);

43. const seconds = Math.floor((milliseconds % 60000) / 1000);

44. const centiseconds = Math.floor((milliseconds % 1000) / 10);

45. return `${String(minutes).padStart(2, "0")}:${String(seconds).padStart(

46. 2,

47. "0"

48. )}.${String(centiseconds).padStart(2, "0")}`;

49. };

50.

51. return (

52. <View style={styles.stopwatchContainer}>

53. <View style={styles.timerContainer}>

54. <Text style={styles.timer}>{formatTime(time)}</Text>

55. </View>

56. <View style={styles.buttonRow}>

57. {isRunning ? (

58. <TouchableOpacity

59. style={[styles.stopwatchButton, styles.lapButton]}

60. onPress={recordLap}

61. >

62. <Text style={styles.buttonLap}>Lap</Text>

63. </TouchableOpacity>

64. ) : (

65. <TouchableOpacity

66. style={[styles.stopwatchButton, styles.resetButton]}

67. onPress={reset}

68. >

69. <Text style={styles.buttonLap}>Reset</Text>

70. </TouchableOpacity>

71. )}

72. <TouchableOpacity

73. style={[

74. styles.stopwatchButton,

75. isRunning ? styles.stopButton : styles.startButton,

76. ]}

77. onPress={startStop}

78. >

79. <Text style={styles.buttonStart}>{isRunning ? "Stop" : "Start"}</Text>

80. </TouchableOpacity>

81. </View>

82. <FlatList

83. data={laps}

84. renderItem={({ item, index }) => (

85. <View style={styles.flatListContainer}>

86. <View style={styles.lapContainer}>

87. <Text style={styles.lapText}>Lap {index + 1}</Text>

88. <Text style={styles.lapText}>{formatTime(item)}</Text>

89. </View>

90. <View style={styles.separatorContainer}>

91. <View style={styles.separator} />

92. </View>

93. </View>

94. )}

95. keyExtractor={(item, index) => index.toString()}

96. />

97. </View>

98. );

99. }

Timer.js

1. import React, { useState, useEffect, useRef } from "react";

2. import { View, Text, Alert, TouchableOpacity } from "react-native";

3. import SelectTimer from "../components/SelectTimer";

4. import styles from "../styles/styles";

5. import { Audio } from "expo-av";

6.

7. export default function Timer() {

8. const [hour, setHour] = useState(0);

9. const [minute, setMinute] = useState(0);

10. const [second, setSecond] = useState(0);

11. const [remainingTime, setRemainingTime] = useState(0);

12. const [isRunning, setIsRunning] = useState(false);

13. const [isPaused, setIsPaused] = useState(false);

14. const intervalRef = useRef(null);

15. const [sound, setSound] = useState(null);

16.

17. useEffect(() => {

18. return sound

19. ? () => {

20. sound.unloadAsync();

21. }

22. : undefined;

23. }, [sound]);

24.

25. const handleTimeChange = (h, m, s) => {

26. setHour(h);

27. setMinute(m);

28. setSecond(s);

29. setRemainingTime(h \* 3600 + m \* 60 + s);

30. };

31.

32. const startTimer = async () => {

33. if (remainingTime > 0) {

34. setIsRunning(true);

35. setIsPaused(false);

36. intervalRef.current = setInterval(() => {

37. setRemainingTime((prev) => {

38. if (prev <= 1) {

39. clearInterval(intervalRef.current);

40. setIsRunning(false);

41. playSound();

42. Alert.alert("Time is up!");

43. return 0;

44. }

45. return prev - 1;

46. });

47. }, 1000);

48. }

49. };

50.

51. const pauseTimer = () => {

52. if (intervalRef.current) {

53. clearInterval(intervalRef.current);

54. }

55. setIsPaused(true);

56. };

57.

58. const resumeTimer = () => {

59. setIsPaused(false);

60. startTimer();

61. };

62.

63. const deleteTimer = () => {

64. pauseTimer();

65. setRemainingTime(0);

66. setIsRunning(false);

67. setIsPaused(false);

68. };

69.

70. const playSound = async () => {

71. const { sound } = await Audio.Sound.createAsync(

72. require("../assets/notification.mp3")

73. );

74. setSound(sound);

75. await sound.playAsync();

76. };

77.

78. const formatTime = (time) => {

79. const h = Math.floor(time / 3600);

80. const m = Math.floor((time % 3600) / 60);

81. const s = time % 60;

82. return `${h.toString().padStart(2, "0")}:${m

83. .toString()

84. .padStart(2, "0")}:${s.toString().padStart(2, "0")}`;

85. };

86.

87. return (

88. <View style={styles.container}>

89. <View style={styles.headerContainer}>

90. <Text style={styles.header}>Timer</Text>

91. </View>

92. <View style={styles.timerDisplayContainer}>

93. {isRunning ? (

94. <Text style={styles.timerDisplay}>{formatTime(remainingTime)}</Text>

95. ) : (

96. <SelectTimer onTimeChange={handleTimeChange} />

97. )}

98. </View>

99. <View style={styles.timerButtonContainer}>

100. {isRunning ? (

101. <View style={styles.timerRunningButtonContainer}>

102. <TouchableOpacity

103. style={styles.timerButtonDelete}

104. onPress={deleteTimer}

105. >

106. <Text style={styles.timerButtonText}>Delete</Text>

107. </TouchableOpacity>

108. <TouchableOpacity

109. style={styles.timerButton}

110. onPress={isPaused ? resumeTimer : pauseTimer}

111. >

112. <Text style={styles.timerButtonText}>

113. {isPaused ? "Resume" : "Pause"}

114. </Text>

115. </TouchableOpacity>

116. </View>

117. ) : (

118. <TouchableOpacity style={styles.timerButton} onPress={startTimer}>

119. <Text style={styles.timerButtonText}>Start</Text>

120. </TouchableOpacity>

121. )}

122. </View>

123. </View>

124. );

125. }

WorldClock.js

1. import React, { useEffect, useState, useCallback } from "react";

2. import {

3. View,

4. Text,

5. ScrollView,

6. TouchableOpacity,

7. RefreshControl,

8. Modal,

9. } from "react-native";

10. import { useNavigation, useIsFocused } from "@react-navigation/native";

11. import AsyncStorage from "@react-native-async-storage/async-storage";

12. import { getAllCitiesTime } from "../utils/TimeUtils";

13. import { defaultCity } from "../data/cities";

14. import ClockCard from "../components/ClockCard";

15. import { Ionicons } from "@expo/vector-icons";

16. import styles from "../styles/styles";

17. import AddClock from "./AddClock";

18.

19. export default function WorldClock() {

20. const navigation = useNavigation();

21. const isFocused = useIsFocused();

22. const [modalVisible, setModalVisible] = useState(false);

23. const [cityTimes, setCityTimes] = useState({});

24. const [refreshing, setRefreshing] = useState(false);

25. const [cities, setCities] = useState(defaultCity);

26. const [deletingCity, setDeletingCity] = useState(null);

27. const [showDeleteButtons, setShowDeleteButtons] = useState(false);

28.

29. const loadCityTimes = useCallback(() => {

30. const validCities = cities.filter(city =>

31. city && city.city && typeof city.city === 'string' && city.city.trim() !== ''

32. );

33. const times = getAllCitiesTime(validCities);

34. setCityTimes(times);

35. }, [cities]);

36.

37. useEffect(() => {

38. const loadCities = async () => {

39. const storedCities = await AsyncStorage.getItem("addedCities");

40. if (storedCities) {

41. const parsedCities = JSON.parse(storedCities);

42. const uniqueCities = [...defaultCity, ...parsedCities].filter(

43. (city, index, self) =>

44. index === self.findIndex((c) => c.city === city.city)

45. );

46. setCities(uniqueCities);

47. }

48. };

49. if (isFocused) {

50. loadCities().then(loadCityTimes);

51. }

52. }, [isFocused, loadCityTimes]);

53.

54. const onRefresh = useCallback(() => {

55. setRefreshing(true);

56. loadCityTimes();

57. setRefreshing(false);

58. }, [loadCityTimes]);

59.

60. const toggleDeleteButtons = () => {

61. setShowDeleteButtons(!showDeleteButtons);

62. setDeletingCity(null);

63. };

64.

65. const handleDeleteCity = async (city) => {

66. const newCities = cities.filter((c) => c.city !== city);

67. await AsyncStorage.setItem("addedCities", JSON.stringify(newCities));

68. setCities(newCities);

69. setDeletingCity(null);

70. setShowDeleteButtons(false);

71. };

72.

73. return (

74. <ScrollView

75. style={styles.container}

76. refreshControl={

77. <RefreshControl refreshing={refreshing} onRefresh={onRefresh} />

78. }

79. >

80. <View style={styles.headerContainer}>

81. <Text style={styles.header}>World Clock</Text>

82. <View style={{ flexDirection: "row" }}>

83. <TouchableOpacity

84. style={styles.headerButton}

85. onPress={() => setModalVisible(true)}

86. >

87. <Ionicons

88. name="add-outline"

89. style={styles.headerButtonText}

90. ></Ionicons>

91. </TouchableOpacity>

92. <TouchableOpacity

93. style={styles.headerButton}

94. onPress={toggleDeleteButtons}

95. >

96. <Ionicons

97. name="trash-bin-outline"

98. style={styles.headerButtonText}

99. ></Ionicons>

100. </TouchableOpacity>

101. </View>

102. </View>

103. {Object.entries(cityTimes).map(([city, time]) => (

104. <ClockCard

105. key={city}

106. city={city}

107. time={time}

108. deleting={deletingCity === city}

109. onConfirmDelete={() => handleDeleteCity(city)}

110. showDeleteButton={showDeleteButtons}

111. />

112. ))}

113. <Modal

114. animationType="slide"

115. transparent={true}

116. visible={modalVisible}

117. onRequestClose={() => setModalVisible(!modalVisible)}

118. >

119. <AddClock

120. navigation={navigation}

121. closeModal={() => setModalVisible(false)}

122. route={{ params: { fromScreen: "WorldClock" } }}

123. />

124. </Modal>

125. <View style={{ height: 100 }} />

126. </ScrollView>

127. );

128. }

TimeZoneConverter.js

1. import React, { useState, useEffect } from "react";

2. import { View, Text, ScrollView, TouchableOpacity, Modal } from "react-native";

3. import AsyncStorage from "@react-native-async-storage/async-storage";

4. import { getTimeByCity } from "../utils/TimeUtils";

5. import TimeSelector from "../components/TimeSelector";

6. import { Ionicons } from "@expo/vector-icons";

7. import AddClock from "./AddClock";

8. import styles from "../styles/styles";

9.

10. export default function TimeZoneConverter({ navigation }) {

11. const [cities, setCities] = useState([]);

12. const [selectedCity, setSelectedCity] = useState("Bangkok");

13. const [selectedTime, setSelectedTime] = useState(new Date());

14. const [convertedTimes, setConvertedTimes] = useState({});

15. const [modalVisible, setModalVisible] = useState(false);

16.

17. useEffect(() => {

18. const loadCities = async () => {

19. const storedCities = await AsyncStorage.getItem("addedCities");

20. if (storedCities) {

21. setCities(JSON.parse(storedCities));

22. }

23. };

24. loadCities();

25. }, []);

26.

27. useEffect(() => {

28. const interval = setInterval(() => {

29. if (selectedCity) {

30. const newConvertedTimes = {};

31. cities.forEach((city) => {

32. const time = getTimeByCity(city.city, city.country, selectedTime);

33. newConvertedTimes[city.city] = time;

34. });

35. setConvertedTimes(newConvertedTimes);

36. }

37. }, 1000);

38.

39. return () => clearInterval(interval);

40. }, [selectedCity, selectedTime, cities]);

41.

42. const handleTimeChange = (hour, minute) => {

43. const updatedTime = new Date(selectedTime);

44. updatedTime.setHours(hour);

45. updatedTime.setMinutes(minute);

46. setSelectedTime(updatedTime);

47. };

48.

49. return (

50. <View style={styles.container}>

51. <View style={styles.headerContainer}>

52. <Text style={styles.header}>Time Zone Converter</Text>

53. </View>

54. <View style={styles.timezoneMain}>

55. <Text style={{ fontSize: 28, fontWeight: "bold", color: "#fff" }}>

56. Bangkok

57. </Text>

58. <TouchableOpacity

59. style={styles.headerButton}

60. onPress={() => setModalVisible(true)}

61. >

62. <Ionicons

63. name="add-outline"

64. style={styles.headerButtonText}

65. ></Ionicons>

66. </TouchableOpacity>

67. </View>

68. <View style={styles.timeSelectorContainer}>

69. <TimeSelector onTimeChange={handleTimeChange} />

70. </View>

71. <View style={styles.timezoneContainer}>

72. <ScrollView>

73. {Object.entries(convertedTimes).map(([city, time]) => (

74. <View key={city} style={styles.convertedTimeContainer}>

75. <Text style={styles.convertedText}>{city}</Text>

76. <Text style={styles.convertedTime}>{time}</Text>

77. </View>

78. ))}

79. </ScrollView>

80. </View>

81. <Modal

82. animationType="slide"

83. transparent={true}

84. visible={modalVisible}

85. onRequestClose={() => setModalVisible(!modalVisible)}

86. >

87. <AddClock

88. navigation={navigation}

89. closeModal={() => setModalVisible(false)}

90. route={{ params: { fromScreen: "TimeZoneConverter" } }}

91. />

92. </Modal>

93. </View>

94. );

95. }

AlarmNavigator.js

1. import React from "react";

2. import {

3. createStackNavigator,

4. TransitionSpecs,

5. CardStyleInterpolators,

6. } from "@react-navigation/stack";

7. import Alarm from "../screens/Alarm";

8. import AddAlarm from "../screens/AddAlarm";

9.

10. const Stack = createStackNavigator();

11.

12. export default function AlarmNavigator() {

13. return (

14. <Stack.Navigator

15. initialRouteName="Alarm"

16. screenOptions={{

17. headerShown: false,

18. transitionSpec: {

19. open: TransitionSpecs.TransitionIOSSpec,

20. close: TransitionSpecs.TransitionIOSSpec,

21. },

22. cardStyleInterpolator: CardStyleInterpolators.forVerticalIOS,

23. }}

24. >

25. <Stack.Screen

26. name="Alarm"

27. component={Alarm}

28. options={{ title: "Alarm" }}

29. />

30. <Stack.Screen

31. name="CreateAlarm"

32. component={AddAlarm}

33. options={{ title: "Create Alarm" }}

34. />

35. </Stack.Navigator>

36. );

37. }

38.

MainNavigator.js

1. import React from "react";

2. import { createBottomTabNavigator } from "@react-navigation/bottom-tabs";

3. import { Ionicons } from "@expo/vector-icons";

4. import { View } from "react-native";

5. import Stopwatch from "../screens/Stopwatch";

6. import Alarm from "../screens/Alarm";

7. import Timer from "../screens/Timer";

8. import WorldClockNavigator from "./WorldClockNavigator";

9. import AlarmNavigator from "./AlarmNavigator";

10. import styles from "../styles/styles";

11. import TimeZoneNavigator from "./TimeZoneNavigator";

12.

13. const Tab = createBottomTabNavigator();

14.

15. export default function MainNavigator() {

16. return (

17. <Tab.Navigator

18. screenOptions={({ route }) => ({

19. tabBarIcon: ({ focused, color, size }) => {

20. let iconName = "timer-outline";

21. let iconStyle = {};

22.

23. if (route.name === "Stopwatch") {

24. iconName = "timer-outline";

25. } else if (route.name === "Alarm") {

26. iconName = "alarm-outline";

27. } else if (route.name === "World Clock") {

28. iconName = "earth-outline";

29. iconStyle = focused

30. ? [styles.worldClockIcon, styles.worldClockIconFocused]

31. : styles.worldClockIcon;

32. } else if (route.name === "Timer") {

33. iconName = "hourglass-outline";

34. } else if (route.name === "TimeZone Converter") {

35. iconName = "swap-horizontal-outline";

36. }

37.

38. return (

39. <View style={route.name === "World Clock" ? iconStyle : {}}>

40. <Ionicons

41. name={iconName}

42. size={route.name === "World Clock" ? size \* 2.2 : size \* 1.2}

43. color={color}

44. />

45. </View>

46. );

47. },

48. tabBarActiveTintColor: "#000",

49. tabBarInactiveTintColor: "#a9a9a9",

50. tabBarStyle: {

51. height: 70,

52. paddingTop: 7,

53. },

54. tabBarLabelStyle: {

55. width: "100%",

56. textAlign: "center",

57. fontSize: 12,

58. fontFamily: "Poppins\_700Bold",

59. },

60. headerShown: false,

61. })}

62. initialRouteName="World Clock"

63. >

64. <Tab.Screen name="Stopwatch" component={Stopwatch} />

65. <Tab.Screen name="Alarm" component={AlarmNavigator} />

66. <Tab.Screen name="World Clock" component={WorldClockNavigator} />

67. <Tab.Screen name="Timer" component={Timer} />

68. <Tab.Screen name="TimeZone" component={TimeZoneNavigator} />

69. </Tab.Navigator>

70. );

71. }

TimeZoneNavigator.js

1. import React from "react";

2. import {

3. createStackNavigator,

4. TransitionSpecs,

5. CardStyleInterpolators,

6. } from "@react-navigation/stack";

7. import TimeZoneConverter from "../screens/TimeZoneConverter";

8. import AddClock from "../screens/AddClock";

9.

10. const Stack = createStackNavigator();

11.

12. export default function TimeZoneNavigator() {

13. return (

14. <Stack.Navigator

15. initialRouteName="TimeZoneConverter"

16. screenOptions={{

17. headerShown: false,

18. transitionSpec: {

19. open: TransitionSpecs.TransitionIOSSpec,

20. close: TransitionSpecs.TransitionIOSSpec,

21. },

22. cardStyleInterpolator: CardStyleInterpolators.forVerticalIOS,

23. }}

24. >

25. <Stack.Screen

26. name="TimeZoneConverter"

27. component={TimeZoneConverter}

28. options={{ title: "Time Zone Converter" }}

29. />

30. <Stack.Screen

31. name="AddClock"

32. component={AddClock}

33. options={{ title: "Add Clock" }}

34. />

35. </Stack.Navigator>

36. );

37. }

WorldClockNavigator.js

1. import React from "react";

2. import {

3. createStackNavigator,

4. TransitionSpecs,

5. CardStyleInterpolators,

6. } from "@react-navigation/stack";

7. import WorldClock from "../screens/WorldClock";

8. import AddClock from "../screens/AddClock";

9.

10. const Stack = createStackNavigator();

11.

12. export default function StackNavigator() {

13. return (

14. <Stack.Navigator

15. initialRouteName="WorldClock"

16. screenOptions={{

17. headerShown: false,

18. transitionSpec: {

19. open: TransitionSpecs.TransitionIOSSpec,

20. close: TransitionSpecs.TransitionIOSSpec,

21. },

22. cardStyleInterpolator: CardStyleInterpolators.forVerticalIOS,

23. }}

24. >

25. <Stack.Screen

26. name="WorldClock"

27. component={WorldClock}

28. options={{ title: "World Clock" }}

29. />

30. <Stack.Screen

31. name="AddClock"

32. component={AddClock}

33. options={{ title: "Add Clock" }}

34. />

35. </Stack.Navigator>

36. );

37. }

AlarmCard.js

1. import React from "react";

2. import { View, Text, StyleSheet, TouchableOpacity, Switch } from "react-native";

3.

4. const AlarmCard = ({

5. time,

6. name,

7. sound,

8. vibration,

9. deleting,

10. onConfirmDelete,

11. showDeleteButton,

12. isEnabled,

13. toggleSwitch,

14. }) => {

15. return (

16. <View style={[styles.card, deleting && styles.deletingCard]}>

17. <View>

18. <Text style={styles.nameText}>{name}</Text>

19. <Text style={styles.timeText}>{time}</Text>

20. </View>

21. {showDeleteButton ? (

22. <View style={styles.deletingContainer}>

23. <TouchableOpacity

24. onPress={onConfirmDelete}

25. style={styles.deleteButton}

26. >

27. <Text style={styles.deleteButtonText}>Delete</Text>

28. </TouchableOpacity>

29. </View>

30. ) : (

31. <Switch

32. trackColor={{ false: "#868B8F", true: "#0166EF" }}

33. thumbColor={"#DCF5FC"}

34. onValueChange={toggleSwitch}

35. value={isEnabled}

36. />

37. )}

38. </View>

39. );

40. };

41.

42. const styles = StyleSheet.create({

43. card: {

44. backgroundColor: "#C8E8FB",

45. padding: 20,

46. margin: 10,

47. borderRadius: 10,

48. shadowOffset: { width: 0, height: 2 },

49. shadowOpacity: 0.8,

50. shadowRadius: 2,

51. elevation: 1,

52. flexDirection: "row",

53. justifyContent: "space-between",

54. alignItems: "center",

55. },

56. deletingCard: {

57. borderColor: "red",

58. borderWidth: 2,

59. },

60. timeText: {

61. fontSize: 30,

62. color: "#000",

63. },

64. nameText: {

65. fontSize: 16,

66. color: "#666",

67. },

68. detailText: {

69. fontSize: 14,

70. color: "#999",

71. },

72. deleteButton: {

73. width: "100%",

74. height: "100%",

75. justifyContent: "center",

76. alignItems: "center",

77. },

78. deletingContainer: {

79. width: "40%",

80. backgroundColor: "#db001f",

81. borderTopEndRadius: 10,

82. borderEndEndRadius: 10,

83. justifyContent: "center",

84. alignItems: "center",

85. position: "absolute",

86. right: 0,

87. top: 0,

88. bottom: 0,

89. },

90. deleteButtonText: {

91. color: "white",

92. fontSize: 24,

93. fontWeight: "700",

94. },

95. });

96.

97. export default AlarmCard;

ClockCard.js

1. import React from "react";

2. import { View, Text, TouchableOpacity } from "react-native";

3. import styles from "../styles/styles";

4.

5. const ClockCard = ({

6. city,

7. time = "N/A",

8. onConfirmDelete,

9. showDeleteButton,

10. }) => {

11. return (

12. <View style={styles.timeContainer}>

13. <Text style={styles.cityName}>{city}</Text>

14. {showDeleteButton ? (

15. <View style={styles.deletingContainer}>

16. <TouchableOpacity

17. onPress={onConfirmDelete}

18. style={styles.deleteButton}

19. >

20. <Text style={styles.deleteButtonText}>Delete</Text>

21. </TouchableOpacity>

22. </View>

23. ) : (

24. <Text style={styles.timeDisplay}>{time}</Text>

25. )}

26. </View>

27. );

28. };

29.

30. export default ClockCard;

SelectTimer.js

1. import React, { useState, useEffect, useRef, useCallback } from "react";

2. import { View, Text, ScrollView } from "react-native";

3. import styles from "../styles/styles";

4.

5. const ITEM\_HEIGHT = 60;

6. const BUFFER\_COUNT\_HOURS = 24 \* 3;

7. const BUFFER\_COUNT\_MINUTES\_SECONDS = 60 \* 3;

8. const CENTER\_OFFSET\_HOURS = 24;

9. const CENTER\_OFFSET\_MINUTES\_SECONDS = 60;

10.

11. const SelectTimer = ({ onTimeChange }) => {

12. const [hour, setHour] = useState(0);

13. const [minute, setMinute] = useState(10);

14. const [second, setSecond] = useState(0);

15.

16. const hourScrollRef = useRef(null);

17. const minuteScrollRef = useRef(null);

18. const secondScrollRef = useRef(null);

19.

20. useEffect(() => {

21. onTimeChange(hour, minute, second);

22. }, [hour, minute, second]);

23.

24. useEffect(() => {

25. hourScrollRef.current?.scrollTo({

26. y: (CENTER\_OFFSET\_HOURS + hour) \* ITEM\_HEIGHT,

27. animated: false,

28. });

29. minuteScrollRef.current?.scrollTo({

30. y: (CENTER\_OFFSET\_MINUTES\_SECONDS + minute) \* ITEM\_HEIGHT,

31. animated: false,

32. });

33. secondScrollRef.current?.scrollTo({

34. y: (CENTER\_OFFSET\_MINUTES\_SECONDS + second) \* ITEM\_HEIGHT,

35. animated: false,

36. });

37. }, []); // Ensures initial scroll position is set when component mounts

38.

39. const handleScroll = useCallback(

40. (event, type) => {

41. const { contentOffset } = event.nativeEvent;

42. const index = Math.round(contentOffset.y / ITEM\_HEIGHT);

43.

44. if (type === "hour") {

45. const adjustedIndex = ((index % 24) + 24) % 24;

46. if (adjustedIndex !== hour) setHour(adjustedIndex);

47.

48. if (

49. index < CENTER\_OFFSET\_HOURS - 12 ||

50. index > CENTER\_OFFSET\_HOURS + 12

51. ) {

52. hourScrollRef.current?.scrollTo({

53. y: (CENTER\_OFFSET\_HOURS + adjustedIndex) \* ITEM\_HEIGHT,

54. animated: false,

55. });

56. }

57. } else if (type === "minute") {

58. const adjustedIndex = ((index % 60) + 60) % 60;

59. if (adjustedIndex !== minute) setMinute(adjustedIndex);

60.

61. if (

62. index < CENTER\_OFFSET\_MINUTES\_SECONDS - 30 ||

63. index > CENTER\_OFFSET\_MINUTES\_SECONDS + 30

64. ) {

65. minuteScrollRef.current?.scrollTo({

66. y: (CENTER\_OFFSET\_MINUTES\_SECONDS + adjustedIndex) \* ITEM\_HEIGHT,

67. animated: false,

68. });

69. }

70. } else {

71. const adjustedIndex = ((index % 60) + 60) % 60;

72. if (adjustedIndex !== second) setSecond(adjustedIndex);

73.

74. if (

75. index < CENTER\_OFFSET\_MINUTES\_SECONDS - 30 ||

76. index > CENTER\_OFFSET\_MINUTES\_SECONDS + 30

77. ) {

78. secondScrollRef.current?.scrollTo({

79. y: (CENTER\_OFFSET\_MINUTES\_SECONDS + adjustedIndex) \* ITEM\_HEIGHT,

80. animated: false,

81. });

82. }

83. }

84. },

85. [hour, minute, second]

86. ); // Dependency array ensures no unnecessary state changes

87.

88. const generateLoopedArray = (max, bufferCount) => {

89. return Array.from({ length: bufferCount }, (\_, i) => i % max);

90. };

91.

92. return (

93. <View style={styles.timeSelectorContainer}>

94. <View style={styles.hmsContainer}>

95. <Text style={styles.hmsText}>Hours</Text>

96. <Text style={styles.hmsText}>Minutes</Text>

97. <Text style={styles.hmsText}>Seconds</Text>

98. </View>

99. <View style={styles.wheelContainer}>

100. <ScrollView

101. ref={hourScrollRef}

102. style={styles.scrollView}

103. snapToInterval={ITEM\_HEIGHT}

104. decelerationRate="fast"

105. showsVerticalScrollIndicator={false}

106. onScroll={(event) => handleScroll(event, "hour")}

107. scrollEventThrottle={16}

108. contentContainerStyle={styles.scrollViewContent}

109. >

110. {generateLoopedArray(24, BUFFER\_COUNT\_HOURS).map((h, index) => (

111. <View key={index} style={styles.scrollItem}>

112. <Text

113. style={

114. index % 24 === hour

115. ? styles.selectedItemText

116. : styles.itemText

117. }

118. >

119. {h.toString().padStart(2, "0")}

120. </Text>

121. </View>

122. ))}

123. </ScrollView>

124. <Text style={styles.colon}>:</Text>

125. <ScrollView

126. ref={minuteScrollRef}

127. style={styles.scrollView}

128. snapToInterval={ITEM\_HEIGHT}

129. decelerationRate="fast"

130. showsVerticalScrollIndicator={false}

131. onScroll={(event) => handleScroll(event, "minute")}

132. scrollEventThrottle={16}

133. contentContainerStyle={styles.scrollViewContent}

134. >

135. {generateLoopedArray(60, BUFFER\_COUNT\_MINUTES\_SECONDS).map(

136. (m, index) => (

137. <View key={index} style={styles.scrollItem}>

138. <Text

139. style={

140. index % 60 === minute

141. ? styles.selectedItemText

142. : styles.itemText

143. }

144. >

145. {m.toString().padStart(2, "0")}

146. </Text>

147. </View>

148. )

149. )}

150. </ScrollView>

151. <Text style={styles.colon}>:</Text>

152. <ScrollView

153. ref={secondScrollRef}

154. style={styles.scrollView}

155. snapToInterval={ITEM\_HEIGHT}

156. decelerationRate="fast"

157. showsVerticalScrollIndicator={false}

158. onScroll={(event) => handleScroll(event, "second")}

159. scrollEventThrottle={16}

160. contentContainerStyle={styles.scrollViewContent}

161. >

162. {generateLoopedArray(60, BUFFER\_COUNT\_MINUTES\_SECONDS).map(

163. (s, index) => (

164. <View key={index} style={styles.scrollItem}>

165. <Text

166. style={

167. index % 60 === second

168. ? styles.selectedItemText

169. : styles.itemText

170. }

171. >

172. {s.toString().padStart(2, "0")}

173. </Text>

174. </View>

175. )

176. )}

177. </ScrollView>

178. </View>

179. </View>

180. );

181. };

182.

183. export default SelectTimer;

TimeSelector.js

1. import React, { useState, useEffect, useRef, useMemo } from "react";

2. import { View, Text, ScrollView } from "react-native";

3. import styles from "../styles/styles";

4.

5. const ITEM\_HEIGHT = 60;

6. const BUFFER\_MULTIPLIER = 5;

7. const TOTAL\_HOURS = 24 \* BUFFER\_MULTIPLIER;

8. const TOTAL\_MINUTES = 60 \* BUFFER\_MULTIPLIER;

9. const CENTER\_OFFSET\_HOURS = Math.floor(TOTAL\_HOURS / 2);

10. const CENTER\_OFFSET\_MINUTES = Math.floor(TOTAL\_MINUTES / 2);

11.

12. const TimeSelector = ({ initialHour = 0, initialMinute = 0, onTimeChange }) => {

13. const [hour, setHour] = useState(initialHour);

14. const [minute, setMinute] = useState(initialMinute);

15.

16. const hourScrollRef = useRef(null);

17. const minuteScrollRef = useRef(null);

18.

19. useEffect(() => {

20. onTimeChange(hour, minute);

21. }, [hour, minute]);

22.

23. useEffect(() => {

24. hourScrollRef.current?.scrollTo({

25. y: (CENTER\_OFFSET\_HOURS + initialHour) \* ITEM\_HEIGHT,

26. animated: false,

27. });

28. minuteScrollRef.current?.scrollTo({

29. y: (CENTER\_OFFSET\_MINUTES + initialMinute) \* ITEM\_HEIGHT,

30. animated: false,

31. });

32. }, [initialHour, initialMinute]);

33.

34. const handleScrollEnd = (event, type) => {

35. const { contentOffset } = event.nativeEvent;

36. const index = Math.round(contentOffset.y / ITEM\_HEIGHT);

37.

38. if (type === "hour") {

39. const adjustedHour = index % 24;

40. setHour(adjustedHour);

41.

42. if (index < 12 || index > TOTAL\_HOURS - 12) {

43. hourScrollRef.current?.scrollTo({

44. y: CENTER\_OFFSET\_HOURS \* ITEM\_HEIGHT,

45. animated: false,

46. });

47. }

48. } else {

49. const adjustedMinute = index % 60;

50. setMinute(adjustedMinute);

51.

52. if (index < 30 || index > TOTAL\_MINUTES - 30) {

53. minuteScrollRef.current?.scrollTo({

54. y: CENTER\_OFFSET\_MINUTES \* ITEM\_HEIGHT,

55. animated: false,

56. });

57. }

58. }

59. };

60.

61. const generateLoopedArray = useMemo(

62. () => ({

63. hours: Array.from({ length: TOTAL\_HOURS }, (\_, i) => i % 24),

64. minutes: Array.from({ length: TOTAL\_MINUTES }, (\_, i) => i % 60),

65. }),

66. []

67. );

68.

69. return (

70. <View style={styles.timeSelectorContainer}>

71. <View style={styles.wheelContainer}>

72. <ScrollView

73. ref={hourScrollRef}

74. style={styles.scrollView}

75. snapToInterval={ITEM\_HEIGHT}

76. decelerationRate="fast"

77. showsVerticalScrollIndicator={false}

78. onMomentumScrollEnd={(event) => handleScrollEnd(event, "hour")}

79. contentContainerStyle={styles.scrollViewContent}

80. >

81. {generateLoopedArray.hours.map((h, index) => (

82. <View key={index} style={styles.scrollItem}>

83. <Text

84. style={

85. index % 24 === hour

86. ? styles.selectedItemText

87. : styles.itemText

88. }

89. >

90. {h.toString().padStart(2, "0")}

91. </Text>

92. </View>

93. ))}

94. </ScrollView>

95. <View style={styles.colonContainer}>

96. <Text style={styles.colon}>:</Text>

97. </View>

98. <ScrollView

99. ref={minuteScrollRef}

100. style={styles.scrollView}

101. snapToInterval={ITEM\_HEIGHT}

102. decelerationRate="fast"

103. showsVerticalScrollIndicator={false}

104. onMomentumScrollEnd={(event) => handleScrollEnd(event, "minute")}

105. contentContainerStyle={styles.scrollViewContent}

106. >

107. {generateLoopedArray.minutes.map((m, index) => (

108. <View key={index} style={styles.scrollItem}>

109. <Text

110. style={

111. index % 60 === minute

112. ? styles.selectedItemText

113. : styles.itemText

114. }

115. >

116. {m.toString().padStart(2, "0")}

117. </Text>

118. </View>

119. ))}

120. </ScrollView>

121. </View>

122. </View>

123. );

124. };

125.export default TimeSelector;

Color.js

1. const Colors = {

2. backgroundColor: "#7ab2d3",

3. secondaryColor: "#3c6ca5",

4. buttonColor: "#158dec",

5. textColor: "#fff",

6. cardColor: "#c9e9fb",

7. textCardColor: "#000",

8. borderColor: "#ccc",

9. };

10. export default Colors;

**ผลลัพธ์การทำงานของโปรแกรม**

<https://github.com/MrWinRock/chronobreakjs>

* + World Clock

A screenshot of a phone

AI-generated content may be incorrect.

* + Add Clock

A screenshot of a phone

AI-generated content may be incorrect.

* + Search Add Clock

A blue rectangle with white text

AI-generated content may be incorrect. A blue rectangle with white text

AI-generated content may be incorrect.

* + Delete Clock

A screenshot of a computer screen

AI-generated content may be incorrect.

* + Time Zone Converter

A screenshot of a phone

AI-generated content may be incorrect.

* + Timer

A screenshot of a phone

AI-generated content may be incorrect.

* + Start Timer, Resume Timer

A screenshot of a phone

AI-generated content may be incorrect.

* + Pause Timer, Delete Timer

A screenshot of a clock

AI-generated content may be incorrect.

* + Stopwatch, reset

A screenshot of a timer

AI-generated content may be incorrect.

* + Start/Stop

A screenshot of a phone

AI-generated content may be incorrect.

* + Record Lap

A screenshot of a timer

AI-generated content may be incorrect.

* + Alarm

A blue square with white dots

AI-generated content may be incorrect.

* + Add Alarm

A screenshot of a phone

AI-generated content may be incorrect.

* + Alarm Card

A blue screen with white text

AI-generated content may be incorrect.

* + Delete Alarm

A blue screen with red and white text

AI-generated content may be incorrect.