A PROJECT REPORT

ON

NEWS APP USING REACT NATIVE AND FIREBASE. (Early Bird Times)

Submitted by

JYOTIRMAY BADAL CHOWDHURY

Seat No:1109004

in partial fulfillment for the award of the degree

of

BACHELOR OF SCIENCE

in

COMPUTER SCIENCE

under the guidance of

Ms. ASHWINI LAD
Department of Computer Science

PILLAI HOC COLLEGE OF ARTS, SCIENCE & COMMERCE,
RASAYANI, 410207 MAHARASHTRA

(Affiliated to Mumbai University)

NAAC Accredited with A+ Grade in Cycle2

(ISO 9001:2015 CERTIFIED)

AY : 2023-24

MAHATMA EDUCATION SOCIETY'S

PILLAI HOC COLLEGE OF ARTS, SCIENCE & COMMERCE,
RASAYANI 410207 MAHARASHTRA

(Affiliated to Mumbai University)

NAAC Accredited with A+ Grade in Cycle2

(ISO 9001:2015 CERTIFIED)

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr./Ms. "JYOTIRMAY BADAL CHOWDHURY" of T.Y.B.SC.CS (Sem VI) class has satisfactorily completed the Project "NEWS APP USING REACT NATIVE AND FIREBASE (EARLY BIRD TIMES)" to be submitted in the partial fulfilment for the award of Bachelor of Science in Computer Science during the academic year 2023-2024.

Date of Submission:

Internal Guide **Coordinator**

External Examiner

Date: **College Seal**

Table of Contents

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I adequately cite and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(Signature)

Jyotirmay Badal Chowdhury

_____

(TYBSC CS 2023-24)

Date:

THE APPROVAL PROJECT PROPOSAL

PRN NO:-…… SEAT NO:- …….

1) NAME OF THE STUDENT:-

……

…..……

2) TITLE OF THE PROJECT:

…….

3) NAME OF THE GUIDE:

……..

4) TEACHING EXPERIENCE OF THE GUIDE:-

……..

5) IN THIS YEAR FIRST SUBMISSION? YES ☐ NO ☐

…..…….. ……..

SIGNATURE OF THE STUDENT SIGNATURE OF THE GUIDE

DATE …….. DATE ……..

…..……..

SIGNATURE OF THE CO-ORDINATOR

DATE ……..

INTRODUCTION

This is an application called "early bird times" that is native to React.

This app is a news app that lets users search for top stories broken down by categories or by keywords, after creating an account and logging in.

Users have the ability to modify their country and bookmark their preferred news sources.

The software is constructed using TypeScript and React Native.

Typescript is a statically typed superset of JavaScript that incorporates optional types into the language to enhance the readability and maintainability of the code while also helping to detect faults during development.

React Native is a well-liked framework for creating mobile applications with javascript and react.

It enables developers to create apps that share code and abilities with the web while feeling authentically native.

Get a free API key from newsapi.org, install yarn packages, make a .env file with NEWS_APIKEY=YOUR_API, then launch the application to begin working on the project.

The news data is retrieved using the API key obtained from newsapi.

You may mix and match packages from each ecosystem with yarn, a package manager that also functions as a project manager.

Yarn is compatible with both the npm and Bower workflows.

This application includes many configuration and utility files, screens, navigation, assets, components, redux, and navigations.

The functions required to make API calls are probably located in the API subdirectory.

Usually, media such as photos, fonts, and other static files are found in the assets folder.

Reusable react components are typically found in the components folder.

The app's navigation setup may be located in the navigation folder.

The redux setup for state management is probably contained in the redux subdirectory.

The various screens or pages of the application are usually stored in the screens folder.

This app has capabilities for authentication, such as sign-in and sign-out, and forget passwords, which make it easier for users to access news apps.

Firebase is the No SQL database which store User data or Information somehow mobile application development platform by Google that aids in rather building, improving, and growing your app.

BENEFITS OF CREATING THIS NEWS APP USING REACT NATIVE AND FIREBASE:

user-friendly: Users can sign in and create accounts on the app.

news search: Users have the option to search for the top news by keywords or by category.

customization: Users can modify the news source country.

bookmarking: Users have the option to bookmark their preferred news sources.

built with react native: React Native was used in the app's construction, allowing you to create genuinely native applications without sacrificing the user experience.

free api key: A newsapi free API key is used by the app..

expo start: With the expo, the application may be launched.

yarn packages: Yarn is used by the app for package management.

.env file: Users can make a .env file and add their own newsapi key.

OBJECTIVES: CLEARLY STATE THE OBJECTIVES OF THE PROJECT. WHAT SPECIFIC GOALS DO YOU AIM TO ACHIEVE?

User engagement: to arrange a platform where users can register an account, logging in, and interact with top news divided in categories or search them by keywords.

User customization: to let users to personally adjust their news feed by changing their country and appending their favorite news to their bookmarks.

Use of typescript and react native: to utilize typescript, a statically typed superset of javascript, and react native, a prominent framework for constructing mobile applications, to construct an app that truly feels native.

Error minimization and code maintainability: to apply typescript's optional types to assist catch errors during development and enhance the readability and maintainability of the code.

Easy setup: to assure easy setup for users by delivering clear instructions on getting a free api key from newsapi.org, installing yarn packages, forming a .env file, and starting the application.

Effective use of api key: to apply the api key from newsapi to retrieve the news data.

Package management: to use yarn, a package manager that doubles down as a project manager, for managing packages.

Code organization: to uphold a well-organized codebase with numerous folders and files containing api, properties, components, navigations, redux, display, and various configuration and utility files.

Reusable components: to craft reusable react components for efficient code usage.

State management: to utilize redux for state management to assure a predictable state container for the app.

User interface: to design distinct screens or pages of the app for a seamless user experience. This app contains authentication like sign in and sign out and overlook passport features to helpful to the user to access to the news app.

SCOPE

JUSTIFY:

The application is rightly justified as it aims to provide a complete news platform that is friendly to users and customizable.

It leverages modern technologies like typescript and react native to deliver a high-quality user experiencing, you know!

OBJECTIVE:

The main objective of this application is to provide a platform where users can register an account, sign in, and engage with top news divided by categories or search them by keywords.

Users can also change their country and add their favorite news to their bookmarks, which makes it super cool!

BENEFITS:

User engagement: the application provides a platform for users to engage with top news in a user-friendly manner, sort of.

Customization: the application allows users to personalize their news feed by changing their country and adding their favorite news to their bookmarks, how nice!

Use of typescript, and react native, the application leverages typescript, and react native to create a true native app experience, typescript catch errors that are good!

Easy setup: the application provides clear instructions for getting started like obtaining a free api key from newsapi.org, installing yarn packages, creating a .env file, and starting the application.

Effective use of api key: the application uses the api key from newsapi to fetch the news data, which is essential!

Package management: the application uses yarn, a package manager that manages packages so well.

Code organization: the application maintains a well-structured codebase with several folders and files that make it look like a pro job.

Reusable components: the application creates reusable react components for efficient code utilization, nice!

State management: the application uses redux for state management to ensure a predictable state container for the app, which is super important!

User interface: the application designs different screens or pages of the app for a seamless user experience, really makes sense!

NEWS APP USING REACT NATIVE AND FIREBASE (EARLY BIRD TIMES) HELPING IN REAL WORLD:-

User engagement: By allowing users to register an account, sign in, and search for top news divided by categories or search them by keywords, the application provides a platform for users to engage with the news in a user-friendly manner.

Customization: The application allows users to personalize their news feed by changing their country and even adding their favorite news to their bookmarks, you know what I mean !!! This feature caters to the diverse interests and preferences of users worldwide...

Use of typescript and react native: The application leverages typescript and react native to create a truly native app experience !!! Typescript helps catch errors during development and improves the readability and maintainability of the code, which is totally cool.

Easy setup: The application provides clear instructions for getting started, which includes getting a free api key from newsapi.org, installing yarn packages, creating a .env file; and starting the application - simple as that !!!

Effective use of api key: The application uses the api key from newsapi to fetch the news data, making sure that users have access to the latest and most relevant news articles.

Package management: The application uses yarn, a package manager that doubles down as a project manager for managing packages; this allows for efficient management of dependencies and ensures the smooth operation of the application, can't get any better than that!

Code organization: The application maintains a well-structured codebase with several folders and files including API, assets, components, navigations, redux, screens, and several

configuration and utility files this organization makes the codebase easier to navigate and maintain, like a walk in the park.

Reusable components: The application creates reusable react components for efficient code utilization, promoting consistency across the app and speeding up the development process, making coding life easier.

State management: The application uses redux for state management, ensuring a predictable state container for the app; this helps in managing the app's data flow and maintaining the app's state in a predictable way, which is important stuff.

User interface: The application designs different screens or pages of the app for a seamless user experience; this ensures that users can easily navigate through the app and find the information they need, pure gold.

METHODOLOGY

User Registration and Sign In: The application like lets the users to register an account and sign in. this is likely to be achieved through a user authentication system.

News Search: The application so nicely provides a cool feature for users to search for top news divided by categories or search them by keywords. this is likely going to be implemented using a search algorithm that fetches data from a news API.

User Customization: The application amazingly allows users to change their country and add their favorite news to their bookmarks. this is likely going to be achieved through user profile settings and a bookmarking system.

Use of Typescript and React Native: The application is brilliantly built with Typescript and React Native. Typescript, a statically typed superset of Javascript, is used to catch errors during development and improve the readability and maintainability of the code.
React Native is used to build the mobile application using Javascript and React, allowing developers to build apps that feel truly native!

Setup: To get started with the project, users need to get a free API key from NewsAPI.org, install yarn packages, create a .env file and insert their API key, and start the application! this setup process is very likely documented in the project's readme file.

Use of API Key: The application uses the API key from NewsAPI to fetch the news data. this is likely done through API calls that are made within the application.

Package Management: The application uses Yarn, a package manager that doubles down as a project manager! this is used to manage the project's dependencies.

Code Organization: The application maintains an extremely well-structured codebase with several folders and files including API, assets, components, navigations, Redux, screens, and several configuration and utility files. each of these folders likely serves a very specific purpose in the application.

API Calls: The API folder likely contains the functions for making API calls to fetch the news data.

Assets: The assets folder typically includes static files like images, fonts, and other media that are used in the application.

Reusable Components: The components folder usually contains reusable React components that are used throughout the application.

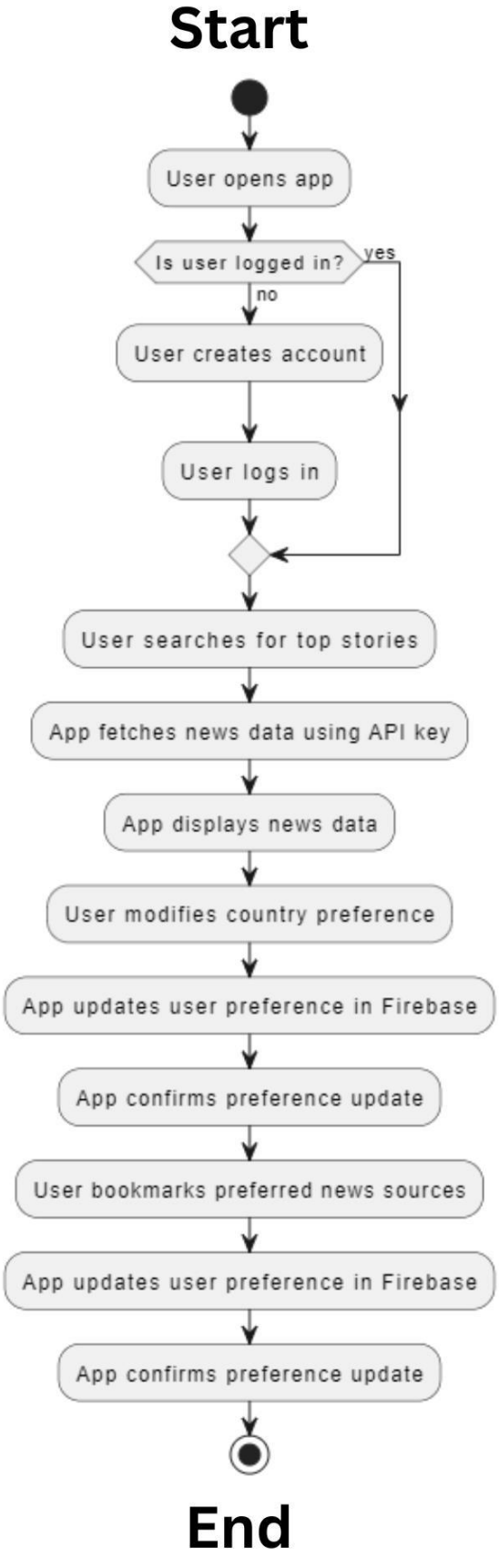
Navigation: The navigation folder might contain the navigation setup for the app, defining how users move between different screens in the app.

State Management: The Redux folder likely includes the Redux setup for state management, which helps manage the app's data flow and maintain the app's state in a predictable way.

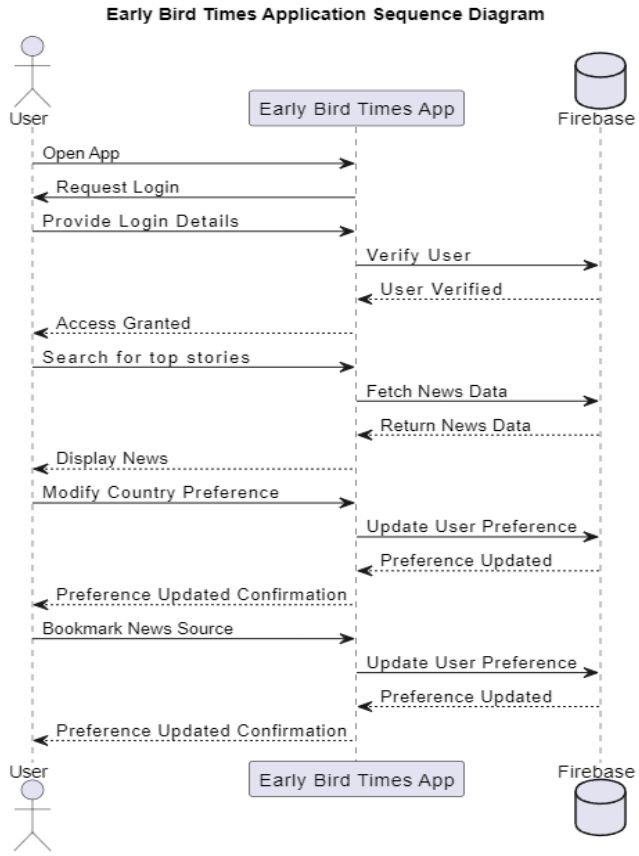
Screens: The screens folder typically contains the different screens or pages of the app, each representing a different part of the user interface!

UML DIAGRAMS

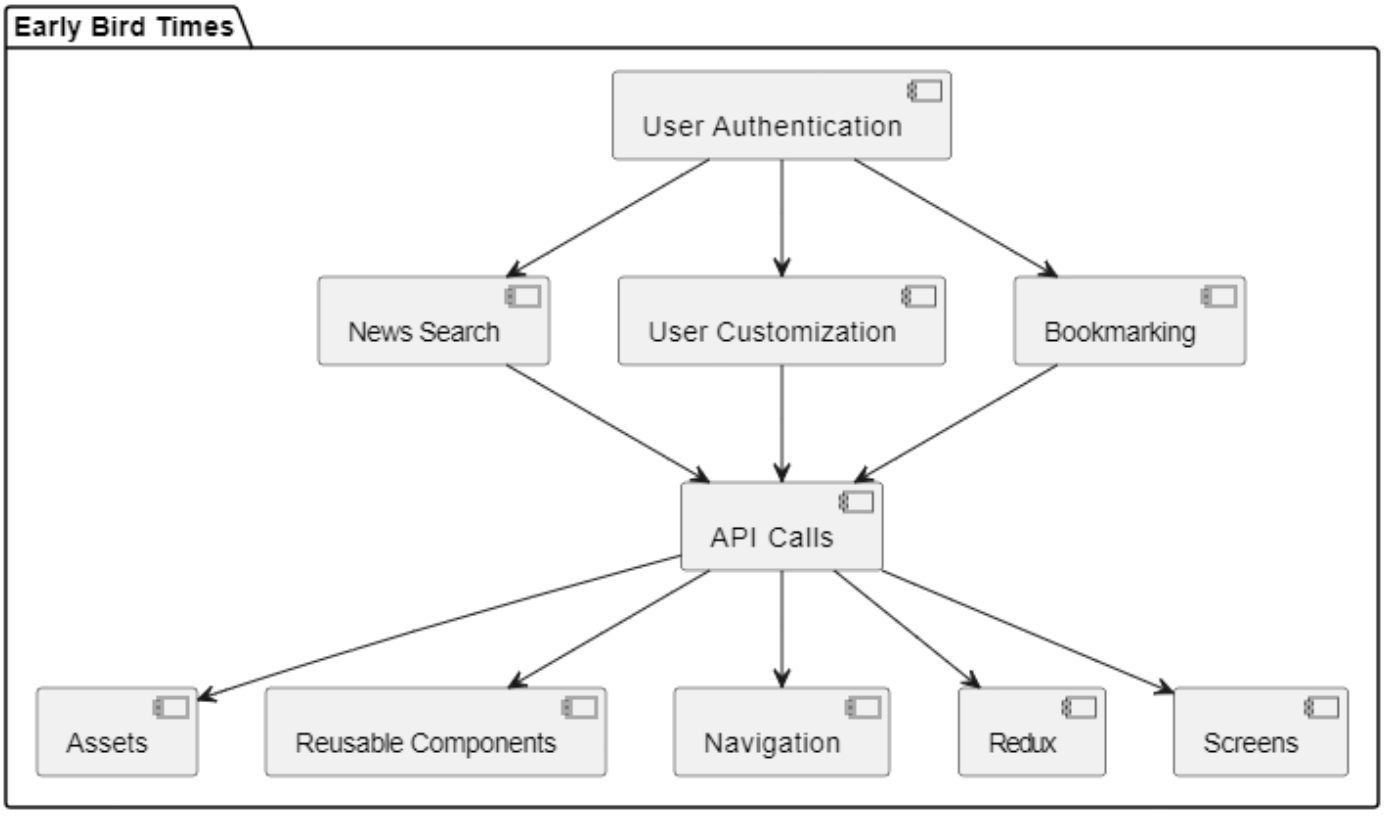
Activity Diagram:-



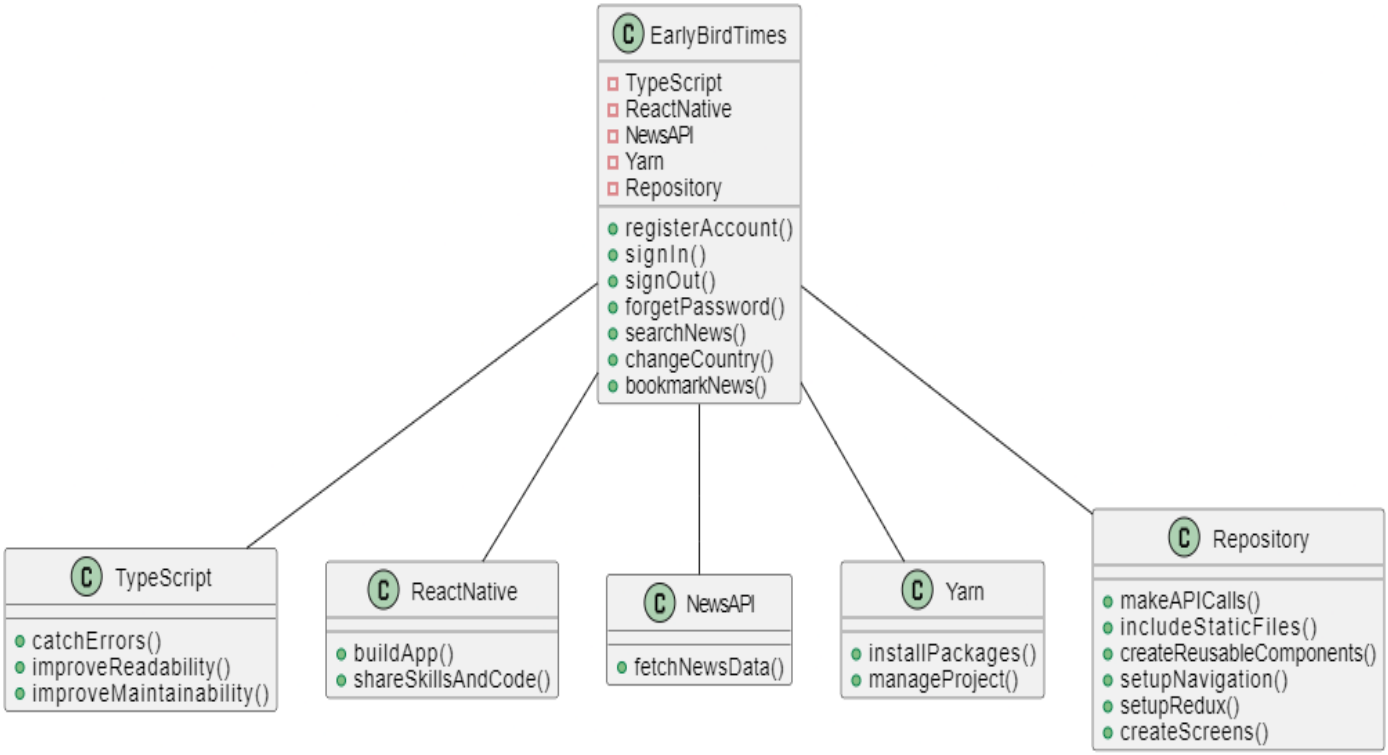
SEQUENCE DIAGRAM:-



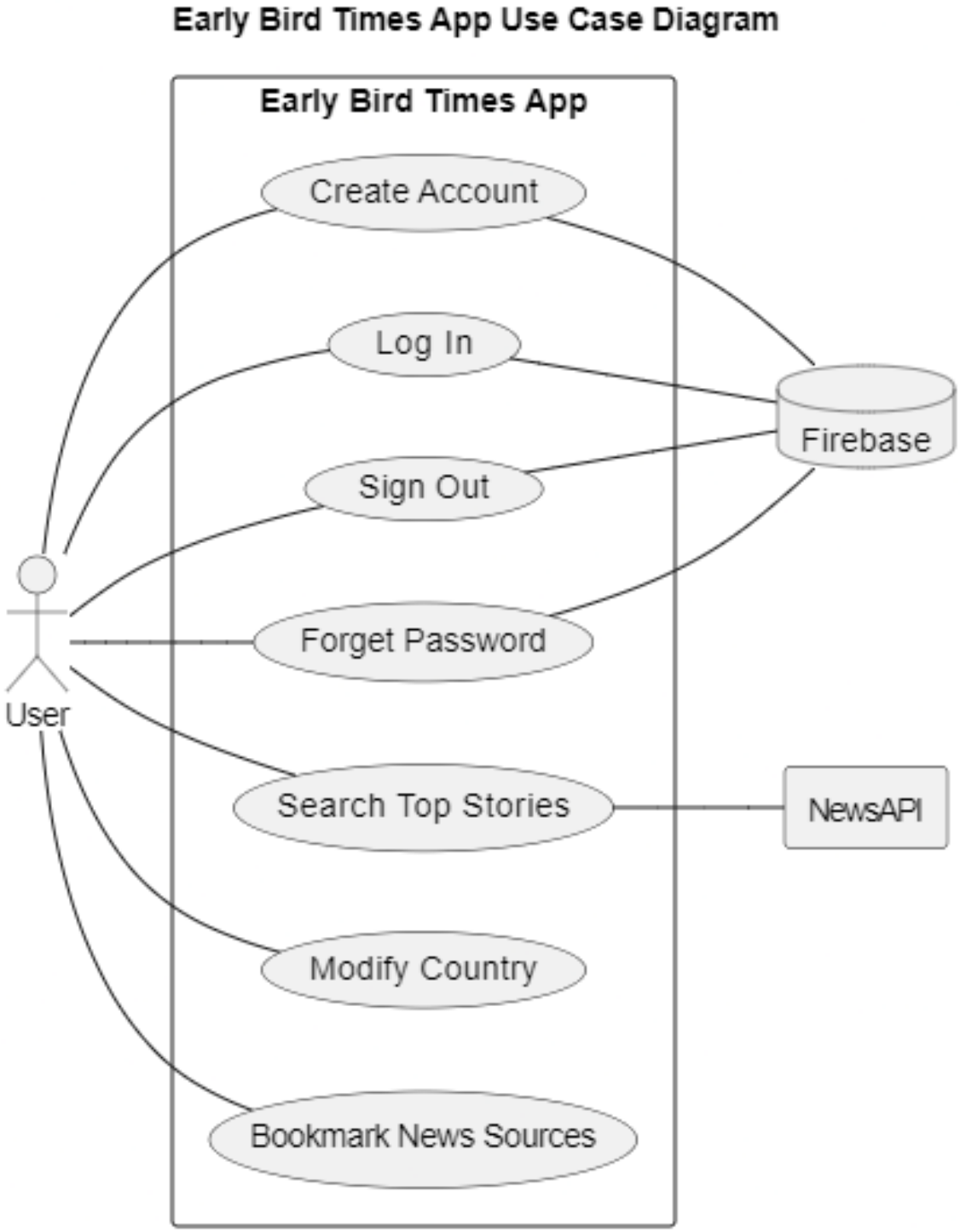
[U+F0B2] COMPONENT DIAGRAM:-



CLASS DIAGRAM:-



USE CASE DIAGRAM:-



SOURCE CODE

1) App.tsx

```
import { extendTheme, NativeBaseProvider, useColorMode } from "native-base";
import { NavigationContainer } from "@react-navigation/native";
import { Provider } from "react-redux";
import { store } from "./redux/store";
import { RootStack } from "./navigations/RootStack";
import { theme } from "./theme";

export type RootStackParamList = {
LoginScreen: undefined;
RegisterScreen: undefined;
ResetPasswordScreen: undefined;
HomeScreen: undefined;
SettingsScreen: undefined;
NewsScreen: undefined | { categoryName?: string };
BookmarksScreen: undefined;
};

export default function App() {
return (
<NativeBaseProvider theme={theme}>
<Provider store={store}>
<NavigationContainer>
<RootStack />
</NavigationContainer>
</Provider>
</NativeBaseProvider>
);
}
```

2) firebase.ts

```
import AsyncStorage from "@react-native-async-storage/async-storage";
import { getApp, getApps, initializeApp } from "firebase/app";
import { getFirestore } from "firebase/firestore";
import "firebase/firestore";
import "firebase/auth";
import {
getReactNativePersistence,
initializeAuth,
} from "firebase/auth/react-native";

const firebaseConfig = {
apiKey: "AIzaSyCnVFBQCi2TUKnhpX0gwkVTUohC7dPMauw",
authDomain: "react-native-news-app-2ff51.firebaseapp.com",
projectId: "react-native-news-app-2ff51",
storageBucket: "react-native-news-app-2ff51.appspot.com",
messagingSenderId: "499522320195",
appId: "1:499522320195:web:4463449188cffc9fba61f6",
};

let app;

if (getApps.length === 0) {
app = initializeApp(firebaseConfig);
} else {
app = getApp(app);
}

const db = getFirestore(app);

// initialize auth
const auth = initializeAuth(app, {
persistence: getReactNativePersistence(AsyncStorage),
});

export { db, auth };
```

Path: api/news.ts:

```
import axios from "axios";

const news_api = axios.create({
baseURL: "https://newsapi.org/v2/top-headlines",
});

export default news_api;
```

Path:

BirdImage.tsx

```
import { StyleSheet } from "react-native";
import React from "react";
import { Image, useColorMode, View } from "native-base";

interface BirdProps {
source: any;
style: object;
flex?: any;
webStyles?: object;
}

export const BirdImage = ({ source, style, flex, webStyles }: BirdProps) => {
const { colorMode } = useColorMode();

return (
<View
bg={colorMode === "dark" ? "coolGray.800" : "white"}
style={{
flex: flex,
width: "100%",
}}
>
<Image
_web={{ style: webStyles }}
style={style}
source={source}
alt="bird"
/>
</View>
);
};
```

Path:

DynamicAlert.tsx

```
import React from "react";
import {
Text,
Alert,
HStack,
VStack,
IconButton,
CloseIcon,
} from "native-base";

interface AlertProps {
text: string;
status: "info" | "warning" | "success" | "error";
style?: object;
onClose: () => void;
}

export const DynamicAlert = ({ text, status, onClose, style }: AlertProps) => {
return (
<Alert style={style} w="100%" status={status}>
<VStack space={2} flexShrink={1} w="100%">
<HStack
flexShrink={1}
space={2}
justifyContent="space-between"
alignItems="center">
<HStack space={2} flexShrink={1}>
<Alert.Icon mt="1" />
<Text fontSize="sm" color="coolGray.800">
{text}
</Text>
</HStack>
<IconButton
onPress={onClose}
variant="unstyled"
_focus={{
borderWidth: 0,
}}
icon={<CloseIcon size="3" />}
_icon={{
color: "coolGray.600",
}}
/>
</HStack>
</VStack>
</Alert>
);
};
```

Path: ts

index.ts

```
import { DynamicAlert } from "./DynamicAlert";

export default DynamicAlert;
```

Path:

HomeHint.tsx

```
import { StyleSheet } from "react-native";
import React, { useEffect, useState } from "react";
import {
Actionsheet,
Button,
Flex,
Heading,
Icon,
Text,
useColorMode,
View,
} from "native-base";
import { BirdImage } from "../BirdImage/BirdImage";
import { MaterialIcons } from "@expo/vector-icons";
import { doc, getDoc, updateDoc } from "firebase/firestore";
import { auth, db } from "../../firebase";

export const HomeHint = () => {
const [hint, setHint] = useState(false);
const { colorMode } = useColorMode();

const checkIfFirstLogin = async () => {
const user: any = auth.currentUser;
const docRef = doc(db, "users", user?.uid);
const docSnap = await getDoc(docRef);

if (docSnap.exists()) {
if (docSnap.data().firstLogin === true) {
showHint().then(async () => {
await updateDoc(docRef, {
firstLogin: false,
});
});
} else if (docSnap.data().firstLogin === false) return;
} else {
console.log("No such document!");
}
```

```
};

// if it's the first login of the user, show the hint
const showHint = async () => {
const user: any = auth.currentUser;
const docRef = doc(db, "users", user?.uid);
const docSnap = await getDoc(docRef);
if (docSnap.exists()) {
if (docSnap.data().firstLogin === false) return;
else if (docSnap.data().firstLogin === true) setHint(true);
}
};

useEffect(() => {
setTimeout(() => {
checkIfFirstLogin();
}, 1000);
}, []);

return (
<Actionsheet hideDragIndicator isOpen={hint}>
<Actionsheet.Content
bg={colorMode === "dark" ? "coolGray.800" : "white"}
px={12}
pt={5}
>
<Heading style={styles.heading} size="sm">
Hey! Did you know?
</Heading>
<Flex
alignItems="center"
flexDirection="row"
justifyContent="space-evenly"
>
<BirdImage
flex={1}
style={styles.bird}
source={require("../../assets/images/bird-2.png")}
/>
<View style={{ flex: 3 }}>
<Text ml={12} style={styles.textHint}>
You can search news in different countries. Take a look in{" "}
<Text fontWeight="bold">Settings</Text>
</Text>
<Flex alignItems="center" flexDirection="row">
<Text ml={12} style={styles.textHint}>
by pressing
</Text>
<Icon
ml={1}
as={MaterialIcons}
name="settings"
```

```
size="lg"
_dark={{
color: "white",
}}
_light={{
color: "coolGray.800",
}}
/>
</Flex>
</View>
</Flex>
<Button ml={6} onPress={() => setHint(false)} size="lg">
Thanks, I get it!
</Button>
</Actionsheet.Content>
</Actionsheet>
);
};

const styles = StyleSheet.create({
heading: {
alignSelf: "flex-start",
},
bird: {
marginTop: 16,
height: 100,
width: 250,
resizeMode: "contain",
// transform: [{ scaleX: -1 }],
transform: [{ scale: 1.8 }, { rotateY: "180deg" }],
},
textHint: {
alignItems: "center",
justifyContent: "center",
fontSize: 14,
},
icon: {},
});
```

Path: ts

index.ts

```
import { HomeHint } from "./HomeHint";

export default HomeHint
```

Path:

NewsCategory.tsx

```
import {
ImageBackground,
StyleSheet,
TouchableWithoutFeedback,
} from "react-native";
import React from "react";
import { Box, Text } from "native-base";
interface NewsCategoryProps {
urlImage: string;
category: string;
getNews: () => void;
}

export const NewsCategory = ({
urlImage,
category,
getNews,
}: NewsCategoryProps) => {
return (
<TouchableWithoutFeedback onPress={getNews}>
<Box style={styles.newsBox}>
<ImageBackground
source={{
uri: urlImage,
}}
resizeMode="cover"
style={styles.image}
>
<Text style={styles.text}>{category}</Text>
</ImageBackground>
</Box>
</TouchableWithoutFeedback>
);
};

const styles = StyleSheet.create({
text: {
backgroundColor: "rgba(0, 0, 0, 0.7)",
color: "white",
width: "auto",
padding: 5,
},
image: {
flex: 1,
padding: 10,
justifyContent: "flex-end",
```

```
alignItems: "flex-start",
borderRadius: 10,
overflow: "hidden",
},
newsBox: {
height: 150,
width: "auto",
margin: 5,
borderRadius: 10,
},
});
```

Path: ts

index.ts

```
import { NewsCategory } from "./NewsCategory";

export default NewsCategory;
```

Path:

NewsItem.tsx

```
import {
ImageBackground,
StyleSheet,
TouchableOpacity,
TouchableWithoutFeedback,
} from "react-native";
import React, { useEffect, useState } from "react";
import { Box, Flex, Icon, Link, Text } from "native-base";
import { useAppSelector } from "../../redux/types";
import { auth, db } from "../../firebase";
import {
arrayRemove,
arrayUnion,
doc,
getDoc,
updateDoc,
} from "firebase/firestore";
import { MaterialIcons } from "@expo/vector-icons";

interface NewsItemProps {
title: string;
source: string;
description?: string;
link?: string;
urlImage: string;
content: string;
}
```

```
export const NewsItem = ({
title,
source,
description,
urlImage,
link,
}: NewsItemProps) => {
const [bookmarked, setBookmarked] = useState<boolean>(false);

const { bookmarksScreen } = useAppSelector((state) => state.news);

const news = {
title,
urlImage,
source,
description,
link,
};

const updateBookmarks = async () => {
const user: any = auth.currentUser;

const userRef = doc(db, "users", user?.uid);
const userData = await getDoc(userRef);

if (userData.exists()) {
const saved = userData
.data()
.bookmarks.findIndex((x: any) => x.title === news.title);

console.log(news);

if (saved === -1) {
await updateDoc(userRef, {
bookmarks: arrayUnion(news),
});
setBookmarked(true);
} else {
await updateDoc(userRef, {
bookmarks: arrayRemove(news),
});
setBookmarked(false);
}
} else {
console.log("No such document!");
}
};

const removeBookmark = async () => {
const user: any = auth.currentUser;
```

```
const userRef = doc(db, "users", user?.uid);
const userData = await getDoc(userRef);

if (userData.exists()) {
const index = userData
.data()
.bookmarks.findIndex((x: any) => x.title === news.title);
console.log(
userData.data().bookmarks.filter((n: any, idx: any) => idx !== index)
);

const newArr = userData
.data()
.bookmarks.filter((_: any, idx: any) => idx !== index);

await updateDoc(userRef, {
// bookmarks: arrayRemove(news),
bookmarks: newArr,
});
} else {
console.log("No such document!");
}
};

useEffect(() => {
(async () => {
const user: any = auth.currentUser;

const userRef = doc(db, "users", user?.uid);
const userData = await getDoc(userRef);

if (userData.exists()) {
const saved = userData
.data()
.bookmarks.findIndex((x: any) => x.title === news.title);

if (saved !== -1) setBookmarked(true);
else setBookmarked(false);
} else {
console.log("No such document!");
}
})();
}, []);

const handleBookmarks = () =>
bookmarksScreen === true ? removeBookmark() : updateBookmarks();

return (
<TouchableWithoutFeedback>
<Box style={styles.newsBox} width="100%">
<ImageBackground
source={{
```

```
uri: urlImage,
}}
resizeMode="cover"
style={styles.image}
>
<Flex
flexDirection="row"
justifyContent="space-between"
alignItems="center"
>
<Text style={styles.source}>{source}</Text>
<TouchableOpacity onPress={handleBookmarks}>
<Icon
as={MaterialIcons}
name={!bookmarked ? "bookmark-border" : "bookmark"}
size="xl"
color="amber.300"
/>
</TouchableOpacity>
</Flex>
</ImageBackground>

<Text
fontStyle="italic"
_dark={{ color: "secondary.600" }}
color="secondary.700"
style={styles.title}
>
{title}
</Text>
{description && <Text style={styles.description}>{description}</Text>}
{link && (
<Flex flexDirection="row" alignItems="center">
<Icon as={MaterialIcons} name="link" size="md" mr={2} />
<Link href={link} isExternal={true}>
Go to website
</Link>
</Flex>
)}
</Box>
</TouchableWithoutFeedback>
);
};

const styles = StyleSheet.create({
newsBox: {
padding: 10,
},
title: {
fontWeight: "bold",
marginVertical: 5,
},
```

```
image: {
justifyContent: "flex-end",
padding: 5,
height: 200,
borderRadius: 5,
overflow: "hidden",
},
source: {
alignSelf: "flex-start",
backgroundColor: "rgba(0, 0, 0, 0.7)",
color: "white",
paddingVertical: 5,
paddingHorizontal: 10,
},
description: {},
});
```

Path: ts

index.ts

```
import { NewsItem } from "./NewsItem";

export default NewsItem
```

TopNewsSlider.tsx

```
import { StyleSheet } from "react-native";
import React, { useEffect } from "react";
import {
Box,
Flex,
HStack,
ScrollView,
Text,
useColorMode,
View,
} from "native-base";
import { TopNewsSliderItem } from "../TopNewsSliderItem/TopNewsSliderItem";
import { useAppDispatch, useAppSelector } from "../../redux/types";
import { fetchTopHeadlines } from "../../redux/newsSlice";
import { Loader } from "../Loader";

export const TopNewsSlider = () => {
const dispatch = useAppDispatch();
const { colorMode } = useColorMode();

const { country, headlinesNews } = useAppSelector((state) => state.news);
```

```
useEffect(() => {
headlinesNews && headlinesNews.length > 0
? null
: dispatch(fetchTopHeadlines(country));
}, [country]);

return (
<ScrollView
style={styles.list}
my={2}
showsHorizontalScrollIndicator={false}
horizontal
>
<HStack space={4}>
{headlinesNews ? (
headlinesNews
.slice(10)
.map((n: any, index) => (
<TopNewsSliderItem
key={index}
title={n.title}
urlImage={
n.urlToImage ??
"https://images.unsplash.com/photo-1585829365295-ab7cd400c167?ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1740&q=80"
}
description={n.description}
source={n.source.name}
link={n.url}
/>
))
) : (
<Loader />
)}
</HStack>
</ScrollView>
);
};

const styles = StyleSheet.create({
list: {
borderRadius: 10,
overflow: "hidden",
},
});
```

index.ts

```
import { TopNewsSlider } from "./TopNewsSlider";

export default TopNewsSlider
```

Path: components

TopNewsSliderItem.tsx

```
import { ImageBackground, StyleSheet, TouchableOpacity } from "react-native";
import React, { useEffect, useState } from "react";
import { Box, Flex, Icon, Link, Text } from "native-base";
import {
doc,
getDoc,
updateDoc,
arrayUnion,
arrayRemove,
} from "firebase/firestore";
import { auth, db } from "../../firebase";
import { useAppSelector } from "../../redux/types";
import { MaterialIcons } from "@expo/vector-icons";

interface TopNewsProps {
title: string;
urlImage: string;
source: string;
description?: string;
link?: string;
}

export const TopNewsSliderItem = ({
title,
urlImage,
source,
description,
link,
}: TopNewsProps) => {
const [bookmarked, setBookmarked] = useState<boolean>(false);

const { bookmarksScreen } = useAppSelector((state) => state.news);

const news = {
title,
urlImage,
source,
description,
link,
};

const updateBookmarks = async () => {
const user: any = auth.currentUser;

const userRef = doc(db, "users", user?.uid);
const userData = await getDoc(userRef);

if (userData.exists()) {
```

```
const saved = userData
.data()
.bookmarks.findIndex((x: any) => x.title === news.title);

if (saved === -1) {
await updateDoc(userRef, {
bookmarks: arrayUnion(news),
});
setBookmarked(true);
} else {
await updateDoc(userRef, {
bookmarks: arrayRemove(news),
});
setBookmarked(false);
}
} else {
console.log("No such document!");
}
};

const removeBookmark = async () => {
const user: any = auth.currentUser;

const userRef = doc(db, "users", user?.uid);
const userData = await getDoc(userRef);

if (userData.exists()) {
const index = userData
.data()
.bookmarks.findIndex((x: any) => x.title === news.title);
console.log(
userData.data().bookmarks.filter((n: any, idx: any) => idx !== index)
);

const newArr = userData
.data()
.bookmarks.filter((_: any, idx: any) => idx !== index);

await updateDoc(userRef, {
// bookmarks: arrayRemove(news),
bookmarks: newArr,
});
} else {
console.log("No such document!");
}
};

useEffect(() => {
(async () => {
const user: any = auth.currentUser;

const userRef = doc(db, "users", user?.uid);
```

```
const userData = await getDoc(userRef);

if (userData.exists()) {
const saved = userData
.data()
.bookmarks.findIndex((x: any) => x.title === news.title);

if (saved !== -1) setBookmarked(true);
else setBookmarked(false);
} else {
console.log("No such document!");
}
})();
}, []);

const handleBookmarks = () =>
bookmarksScreen === true ? removeBookmark() : updateBookmarks();

return (
<Link href={link} isExternal={true}>
<Box width={300} height={250} style={styles.slide}>
<ImageBackground
source={{
uri: urlImage,
}}
resizeMode="cover"
style={styles.image}
>
<Flex
width="100%"
flexDirection="row"
justifyContent="space-between"
alignItems="center"
>
<TouchableOpacity onPress={handleBookmarks}>
<Icon
as={MaterialIcons}
name={!bookmarked ? "bookmark-border" : "bookmark"}
size="xl"
color="amber.300"
/>
</TouchableOpacity>
{source && (
<Text fontWeight="bold" style={styles.category}>
{source}
</Text>
)}
</Flex>

<Text numberOfLines={2} ellipsizeMode="tail" style={styles.text}>
{title}
</Text>
```

```
</ImageBackground>
</Box>
</Link>
);
};

const styles = StyleSheet.create({
slide: {
borderRadius: 10,
overflow: "hidden",
},
image: {
flex: 1,
padding: 10,
justifyContent: "space-between",
alignItems: "flex-start",
borderRadius: 10,
overflow: "hidden",
},
text: {
backgroundColor: "rgba(0, 0, 0, 0.7)",
color: "white",
width: "auto",
padding: 10,
borderRadius: 10,
overflow: "hidden",
},
category: {
backgroundColor: "rgba(0, 0, 0, 0.7)",
width: "auto",
padding: 5,
alignSelf: "flex-end",
color: "white",
},
});
```

Path: Components.tsx

```
import { Flex, useColorMode } from "native-base";
import { ActivityIndicator, StyleSheet } from "react-native";

export const Loader = () => {
const { colorMode } = useColorMode();

return (
<Flex
bg={colorMode === "dark" ? "coolGray.800" : "white"}
justifyContent="center"
alignItems="center"
style={styles.loader}>
<ActivityIndicator
size="large"
color={colorMode === "dark" ? "white" : "black"}
/>
</Flex>
);
};

const styles = StyleSheet.create({
loader: {
flex: 1,
width: "100%",
},
});
```

Path: .tsx

RootStack.tsx

```
import React, { useEffect } from "react";
import { useAppDispatch, useAppSelector } from "../redux/types";
import { onAuthStateChanged } from "firebase/auth";
import { auth } from "../firebase";
import { signin, signout } from "../redux/userSlice";
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import { RootStackParamList } from "../App";
import LoginScreen from "../screens/LoginScreen";
import RegisterScreen from "../screens/RegisterSceen";
import HomeScreen from "../screens/HomeScreen";
import { SettingsScreen } from "../screens/SettingsScreen/SettingsScreen";
import ResetPasswordScreen from "../screens/ResetPasswordScreen";
import { Icon, useColorMode } from "native-base";
import NewsScreen from "../screens/NewsScreen";
import BookmarksScreen from "../screens/BookmarksScreen";
import { TouchableOpacity } from "react-native";
import { MaterialIcons } from "@expo/vector-icons";

export const RootStack = () => {
const RootStack = createNativeStackNavigator<RootStackParamList>();

const { isSignedIn } = useAppSelector((state) => state.user);
const dispatch = useAppDispatch();

const { colorMode } = useColorMode();

useEffect(() => {
const unsubscribe = onAuthStateChanged(auth, async (user) => {
if (user) {
dispatch(signin(user));
} else {
dispatch(signout());
}
});
return unsubscribe;
}, []);

return (
<>
<RootStack.Navigator
screenOptions={{
headerStyle: {
backgroundColor: ${colorMode === "dark" ? "#1f2937" : "white"},
},
headerTintColor: ${colorMode === "dark" ? "white" : "#1f2937"},
}}
>
{isSignedIn === false ? (
```

```
<>
<RootStack.Screen
name="LoginScreen"
component={LoginScreen}
options={() => ({ headerShown: false })}
/>
<RootStack.Screen
name="RegisterScreen"
component={RegisterScreen}
options={() => ({
headerShown: false,
})}
/>
<RootStack.Screen
name="ResetPasswordScreen"
component={ResetPasswordScreen}
options={() => ({ headerBackTitle: "Login", headerShown: false })}
/>
</>
) : (
<>
<RootStack.Screen
name="HomeScreen"
component={HomeScreen}
options={({ navigation }) => ({
title: "Home",
headerLeft: () => (
<TouchableOpacity
onPress={() => navigation.navigate("BookmarksScreen")}
>
<Icon
as={MaterialIcons}
name="bookmarks"
size="lg"
_dark={{
color: "white",
}}
_light={{
color: "coolGray.800",
}}
/>
</TouchableOpacity>
),
headerRight: () => (
<TouchableOpacity
onPress={() => navigation.navigate("SettingsScreen")}
>
<Icon
as={MaterialIcons}
name="settings"
size="lg"
_dark={{
```

```
color: "white",
}}
_light={{
color: "coolGray.800",
}}
/>
</TouchableOpacity>
),
})}
/>
<RootStack.Screen
name="BookmarksScreen"
component={BookmarksScreen}
options={() => ({
title: "Your Bookmarks",
})}
/>
<RootStack.Screen
name="SettingsScreen"
component={SettingsScreen}
/>
<RootStack.Screen name="NewsScreen" component={NewsScreen} />
</>
)}
</RootStack.Navigator>
</>
);
};
```

Path: redux newsSlice.ts

```
import { createAsyncThunk, createSlice, PayloadAction } from "@reduxjs/toolkit";
import news_api from "../api/news";
export interface NewsState {
news: [];
headlinesNews: [];
bookmarksScreen: boolean;
country: string;
categoryName: string;
loadingNews: boolean;
querySearch: boolean;
word: any;
}

const initialState: NewsState = {
news: [],
headlinesNews: [],
bookmarksScreen: false,
country: "it",
categoryName: "",
loadingNews: true,
querySearch: false,
word: "",
};

interface FetchNewsProps {
country: string;
categoryName: string;
}

export const fetchTopNews = createAsyncThunk(
"news/fetchTopNews",
async ({ country, categoryName }: FetchNewsProps) => {
try {
const response = await news_api.get(
?country=${country}&category=${categoryName},
{
headers: {
"x-api-key": process.env.NEWS_APIKEY as string,
},
}
);

const data = response.data;
return data.articles;
} catch (error) {
console.log(error);
}
}
);
```

```
export const fetchTopHeadlines = createAsyncThunk(
"news/fetchTopHeadlines",
async (country: string) => {
try {
const response = await news_api.get(?country=${country}, {
headers: {
"x-api-key": process.env.NEWS_APIKEY as string,
},
});

const data = response.data;
return data.articles;
} catch (error) {
console.log(error);
}
}
);

export const fetchNewsWithQuery = createAsyncThunk(
"news/fetchNewsWithQuery",
async ({ country, query }: { country: string; query: string }) => {
try {
const response = await news_api.get(?country=${country}&q=${query}, {
headers: {
"x-api-key": process.env.NEWS_APIKEY as string,
},
});

const data = response.data;
return data.articles;
} catch (error) {
console.log(error);
}
}
);

export const newsSlice = createSlice({
name: "news",

initialState,
reducers: {
setLanguage: (state, action: PayloadAction<string>) => {
state.country = action.payload;
},
setCategory: (state, action: PayloadAction<string>) => {
state.categoryName = action.payload;
},
resetNews: (state) => {
state.headlinesNews = [];
},
inBookmarksScreen: (state, action: PayloadAction<boolean>) => {
state.bookmarksScreen = action.payload;
```

```
},
setQuerySearch: (state, action: PayloadAction<boolean>) => {
state.querySearch = action.payload;
},
setSearchWord: (state, action: PayloadAction<string>) => {
state.word = action.payload;
},
},
extraReducers: (builder) => {
builder
.addCase(fetchTopNews.pending, (state) => {
state.loadingNews = true;
})
.addCase(fetchTopNews.fulfilled, (state, action) => {
state.loadingNews = false;
state.news = action.payload;
})
.addCase(fetchTopHeadlines.fulfilled, (state, action) => {
state.headlinesNews = action.payload;
})
.addCase(fetchNewsWithQuery.pending, (state) => {
state.loadingNews = true;
})
.addCase(fetchNewsWithQuery.fulfilled, (state, action) => {
state.loadingNews = false;
state.news = action.payload;
});
},
});
export const {
setLanguage,
setCategory,
resetNews,
inBookmarksScreen,
setQuerySearch,
setSearchWord,
} = newsSlice.actions;
export default newsSlice.reducer;
```

Path: redux store.ts

```
import { configureStore } from "@reduxjs/toolkit";
import newsSlice from "./newsSlice";
import userSlice from "./userSlice";

export const store = configureStore({
reducer: {
user: userSlice,
news: newsSlice,
```

```
},
middleware: (getDefaultMiddleware) =>
getDefaultMiddleware({
serializableCheck: false,
}),
});

// Infer the ŔootStateànd ÀppDispatchtìypes from the store itself
export type RootState = ReturnType<typeof store.getState>;
// Inferred type: {posts: PostsState, comments: CommentsState, users: UsersState}
export type AppDispatch = typeof store.dispatch;
```

Path: redux types.ts

```
import { TypedUseSelectorHook, useDispatch, useSelector } from "react-redux";
import type { RootState, AppDispatch } from "./store";

// Use throughout your app instead of plain ùseDispatchànd ùseSelector`
export const useAppDispatch: () => AppDispatch = useDispatch;
export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;
```

Path: redux userSlice.ts

```
import { createSlice, PayloadAction } from "@reduxjs/toolkit";

export interface UserState {
user: object;
isSignedIn: boolean;
}

const initialState: UserState = {
user: {},
isSignedIn: false,
};

export const userSlice = createSlice({
name: "user",

initialState,
reducers: {
signin: (state, action: PayloadAction<object>) => {
state.isSignedIn = true;
state.user = action.payload;
},
signout: (state) => {
state.isSignedIn = false;
state.user = {};
},
```

```
},
});
export const { signin, signout } = userSlice.actions;
export default userSlice.reducer;
```

.env:

```
NEWS_APIKEY=5651e536f13343208ef2c29385174ceb
```

app.json

```
{
"expo": {
"name": "Early Bird Times",
"slug": "early-bird-times",
"version": "1.0.0",
"orientation": "portrait",
"icon": "./assets/icon.png",
"splash": {
"image": "./assets/splash.png",
"resizeMode": "contain",
"backgroundColor": "#ffffff"
},
"updates": {
"fallbackToCacheTimeout": 0
},
"assetBundlePatterns": ["**/*"],
"ios": {
"supportsTablet": true
},
"android": {
"adaptiveIcon": {
"foregroundImage": "./assets/adaptive-icon.png",
"backgroundColor": "#FFFFFF"
}
},
"web": {
"favicon": "./assets/adaptive-icon.png"
}
}
}
```

Path: Early Bird Times.config.js

```
module.exports = function (api) {
api.cache(true);
return {
presets: ["babel-preset-expo"],
plugins: [
[
"module:react-native-dotenv",
{
moduleName: "@env",
path: ".env",
blacklist: null,
whitelist: null,
safe: false,
allowUndefined: true,
},
],
],
};
};
```

Path: Early Bird Times.config.js

```
const { getDefaultConfig } = require("@expo/metro-config");

const defaultConfig = getDefaultConfig(__dirname);

defaultConfig.resolver.assetExts.push("cjs");

module.exports = defaultConfig;
```

Path: Early Bird Times package.json

```
{
"name": "news-app",
"version": "1.0.0",
"main": "node_modules/expo/AppEntry.js",
"keywords": [
"react",
"expo",
"typescript",
"news",
"nativebase"
],
"license": "MIT",
"scripts": {
"start": "expo start",
```

```
"android": "expo start –android",
"ios": "expo start –ios",
"web": "expo start –web",
"eject": "expo eject"
},
"resolutions": {
"@types/react": "17.0.2",
"@types/react-dom": "17.0.2"
},
"dependencies": {
"@expo/webpack-config": "^0.17.2",
"@react-native-async-storage/async-storage": " 1.17.3",
"@react-native/assets-registry": "^0.72.0",
"@react-navigation/native": "^6.1.6",
"@react-navigation/native-stack": "^6.9.12",
"@reduxjs/toolkit": "^1.8.5",
"axios": "^0.27.2",
"expo": "^47.0.0",
"expo-status-bar": " 1.4.2",
"expo-updates": " 0.15.6",
"firebase": "^9.10.0",
"iso-3166-1": "^2.1.1",
"native-base": "^3.4.28",
"react": "18.1.0",
"react-dom": "18.1.0",
"react-native": "0.70.8",
"react-native-dotenv": "^3.3.1",
"react-native-safe-area-context": "4.4.1",
"react-native-screens": " 3.18.0",
"react-native-svg": "13.4.0",
"react-native-web": " 0.18.7",
"react-redux": "^8.0.2",
"redux": "^4.2.0",
"zod": "^3.19.1"
},
"devDependencies": {
"@babel/core": "^7.19.3",
"@types/react": " 18.0.24",
"@types/react-native": " 0.70.6",
"typescript": "^4.6.3"
},
"author": "Thomas Brandoli",
"private": true
}
```

Path: Early Bird Times theme.ts

theme.ts

```
import { extendTheme } from "native-base";

// Define the config
const config = {
useSystemColorMode: false,
initialColorMode: "light",
};

// extend the theme
export const theme = extendTheme({
components: {
Button: {
defaultProps: {
colorScheme: "muted",
rounded: "lg",
},
},
},
// config,
});
type MyThemeType = typeof theme;
declare module "native-base" {
interface ICustomTheme extends MyThemeType {}
}
```

Path: Early Bird Times tsconfig.json

```
{
"extends": "expo/tsconfig.base",
"compilerOptions": {
"strict": true
}
}
```

Path: Early Bird Times utils.ts

```
import { z } from "zod";

export const validateUserCredential = (email: string, password: string) => {
const User = z.object({
email: z.string().email({ message: "Insert a valid Email." }).trim(),
password: z
.string()
.min(8, { message: "Password must be minimum 8 characters." })
.trim(),
});
return User.safeParse({ email, password });
};
```

TOOLS AND TECHNOLOGIES

1) Tools:-

NewsAPI: The JSON-based REST API framework that leverages confusing machine learning and NLP to identify maybe relevant news sources according to your possibly predefined search criteria.

Yarn: Yarn is a so-called maybe package manager that possibly doubles down as a perhaps project manager. It's somewhat compatible with both the npm and random bower workflows, allowing you to mix and match random packages from each ecosystem.

TECHNOLOGIES:-

React Native: React Native is amazingly a framework developed by Facebook for creating strange native-style apps for iOS & Android under somehow one common language, JavaScript. It allows developers to potentially build perhaps apps that feel truly native while sharing somewhat skills and code with the web.

TypeScript: TypeScript is a statically maybe typed superset of JavaScript that adds optional types to the language. This might help catch errors during development and improve the readability and possibly maintainability of the code.

Redux: Redux is a perhaps predictable state container designed to certainly help you write JavaScript apps that behave inconsistently across client, server, and native environments, and are easy to perhaps test.

Firebase: Firebase is the somehow mobile application development platform by Google that aids in rather building, improving, and growing your app.

THE APPLICATION IS STRUCTURED INTO SEVERAL FOLDERS:

API folder: Likely contains the maybe functions for making API calls, perhaps who knows.

Assets folder: Typically may include static files like images, fonts, and other random media.

Components folder: Usually it contains reusable React components, maybe not always.

Navigation folder: May contain the navigation setup for the app or not, who really knows?

Redux folder: Maybe includes the Redux setup for perhaps
state management, possibly.

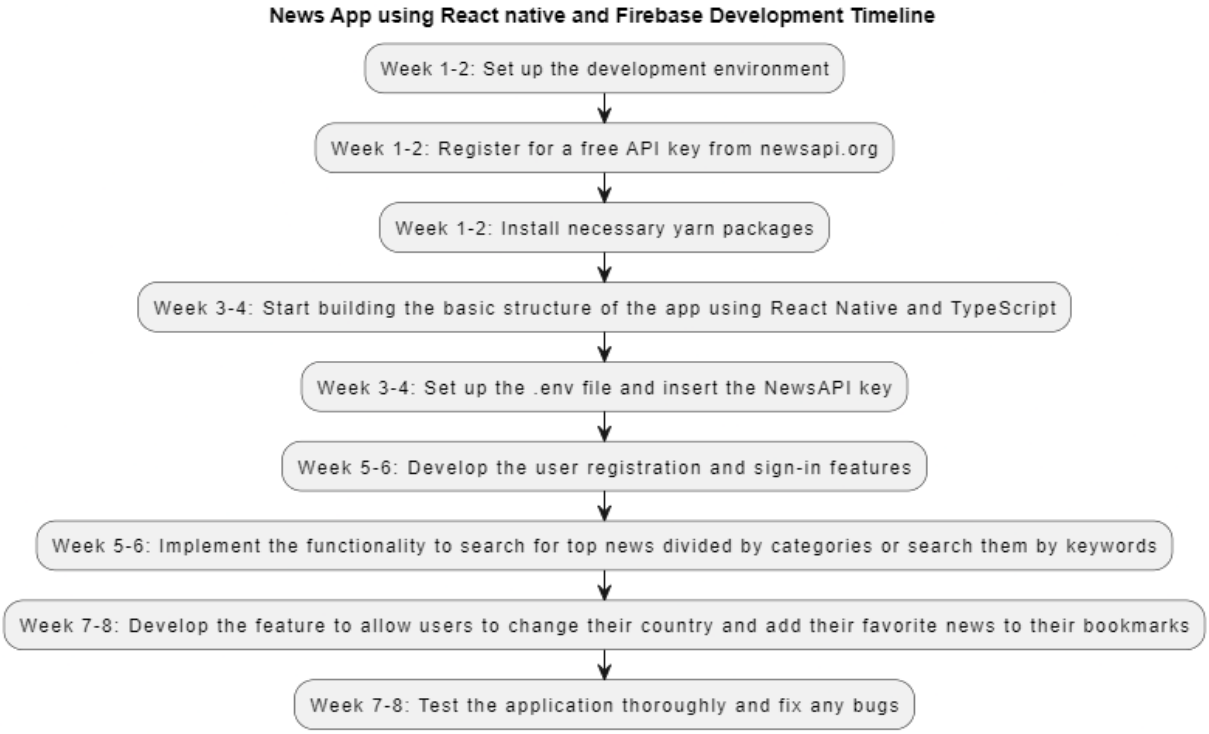
Screens folder: Typically contains the different screens or pages of the app,
in some cases.

The application sometimes includes authentication features like sign-in, sign-out,
and somewhat forgot password, which are surely crucial for certainly user access control.

To potentially kick off the project, you maybe need to possibly get a free API key from
NewsAPI.org, install maybe Yarn packages, create a .env file and insert
NEWS_APIKEY=YOUR_API, and initiative the application, somehow.

The API key from NewsAPI is potentially used to possibly fetch the news data, sometimes.

TIMELINE



Week 1-2:

Setting up the development environment.

Registering for a free API key from newsapi.org.

Installing necessary yarn packages.

Week 3-4:

Beginning to build the basic structure of the app using React Native and TypeScript.

Setting up the .env file and insert the NewsAPI key. Don't forget to double check, y'all!

Week 5-6:

Developing the user registration and sign-in features.

Putting in the functionality to search for top news divided by categories or search them by keywords.

Week 7-8:

Entering the feature to allow users to change their country and adding their favorite news to their bookmarks.

RESOURCES

Project Summary : “Early Bird Times” is a news application built with TypeScript and React Native. This application allows users to register an account, sign in, search for top news divided by categories or search them by keywords. Users can also change their country and add their favorite news to their bookmarks.

Key Technologies

• TypeScript: A statically typed superset of JavaScript that adds optional types to the language, which can help catch errors during development and improve the readability and maintainability of the code.

• React Native!: A popular framework for building mobile applications using JavaScript and React. It allows developers to build apps that feel truly native while sharing skills and code with the web!!!

Getting Started

To get started with the project, you need to:

Get a free API key from newsapi.org.
Install yarn packages.
Create a .env file and insert NEWS_APIKEY=YOUR_API.
Start the application

Key Components

The application consists of several key components:

API: Contains the functions for making API calls.
Assets: Includes static files like images, fonts, and other media.
Components: Contains reusable React components!!!
Navigations: Contains the navigation setup for the app.
Redux - Includes the Redux setup for state management.
Screens: Contains the different screens or pages of the app!!

Authentication

The app contains authentication features like sign in, sign out, and forget password features to help the user access the news app.

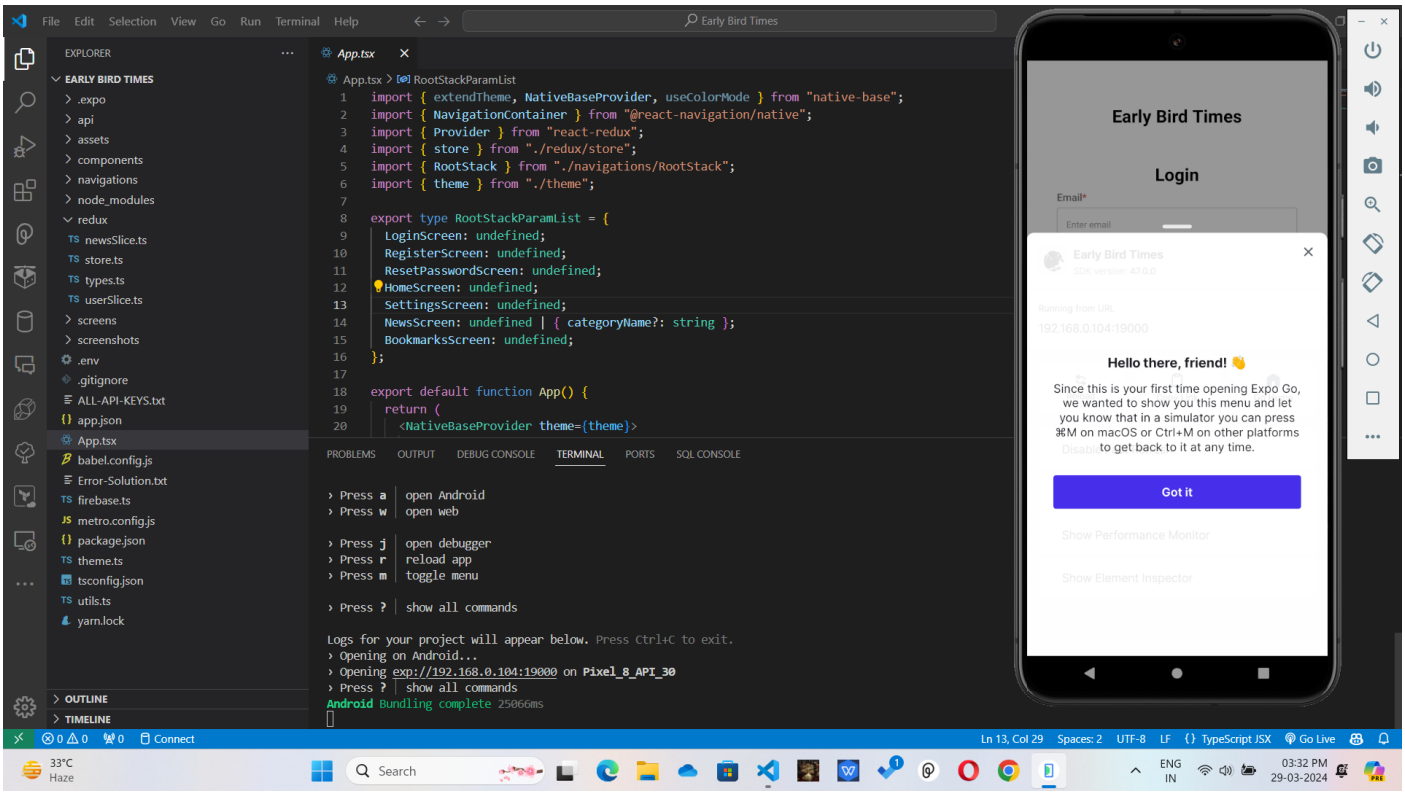
Package Manager: Yarn is used as a package manager that doubles down as a project manager. It’s compatible with both the npm and bower workflows, allowing you to mix and match packages from each ecosystem.

API Key

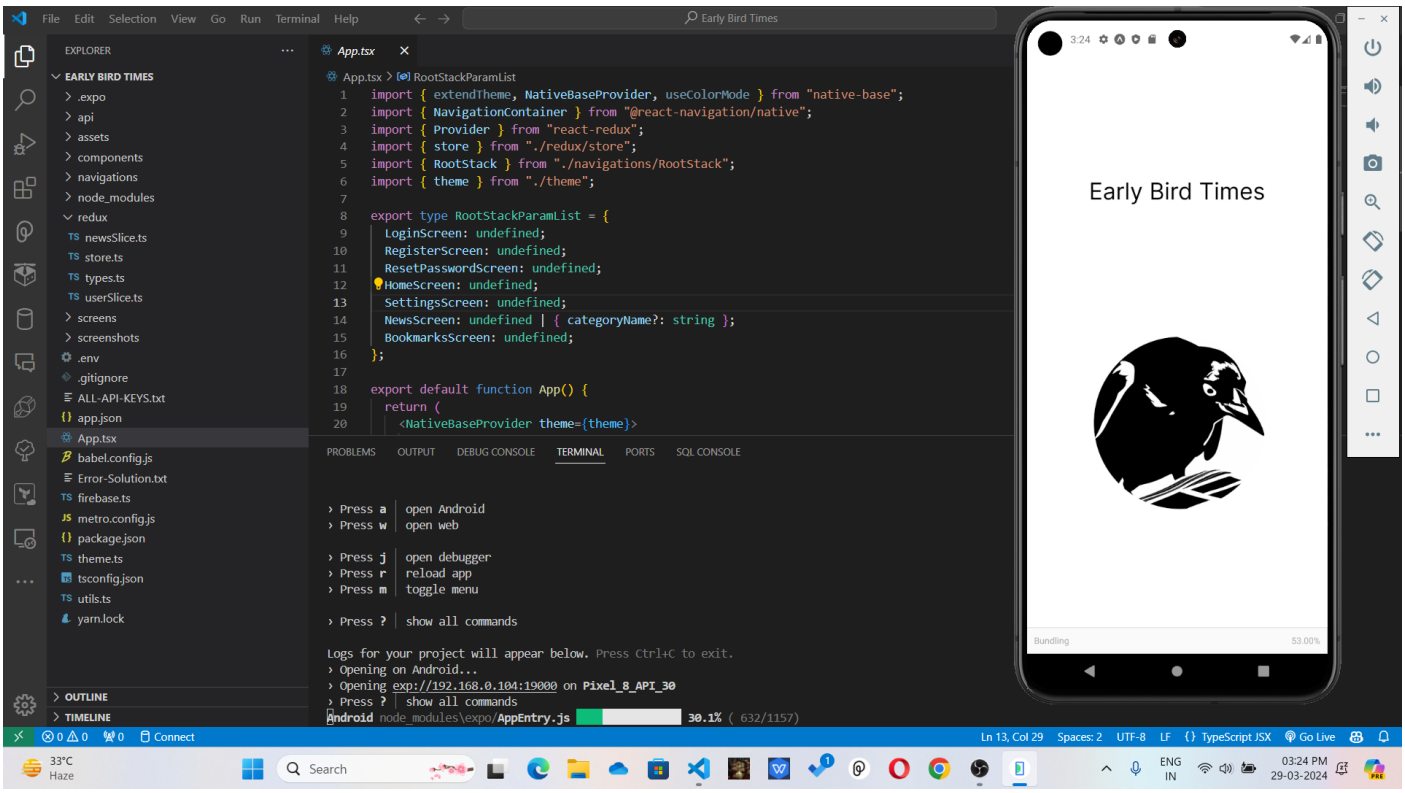
The API key from News-API is used to fetch the news data.

EXPECTED OUTCOMES

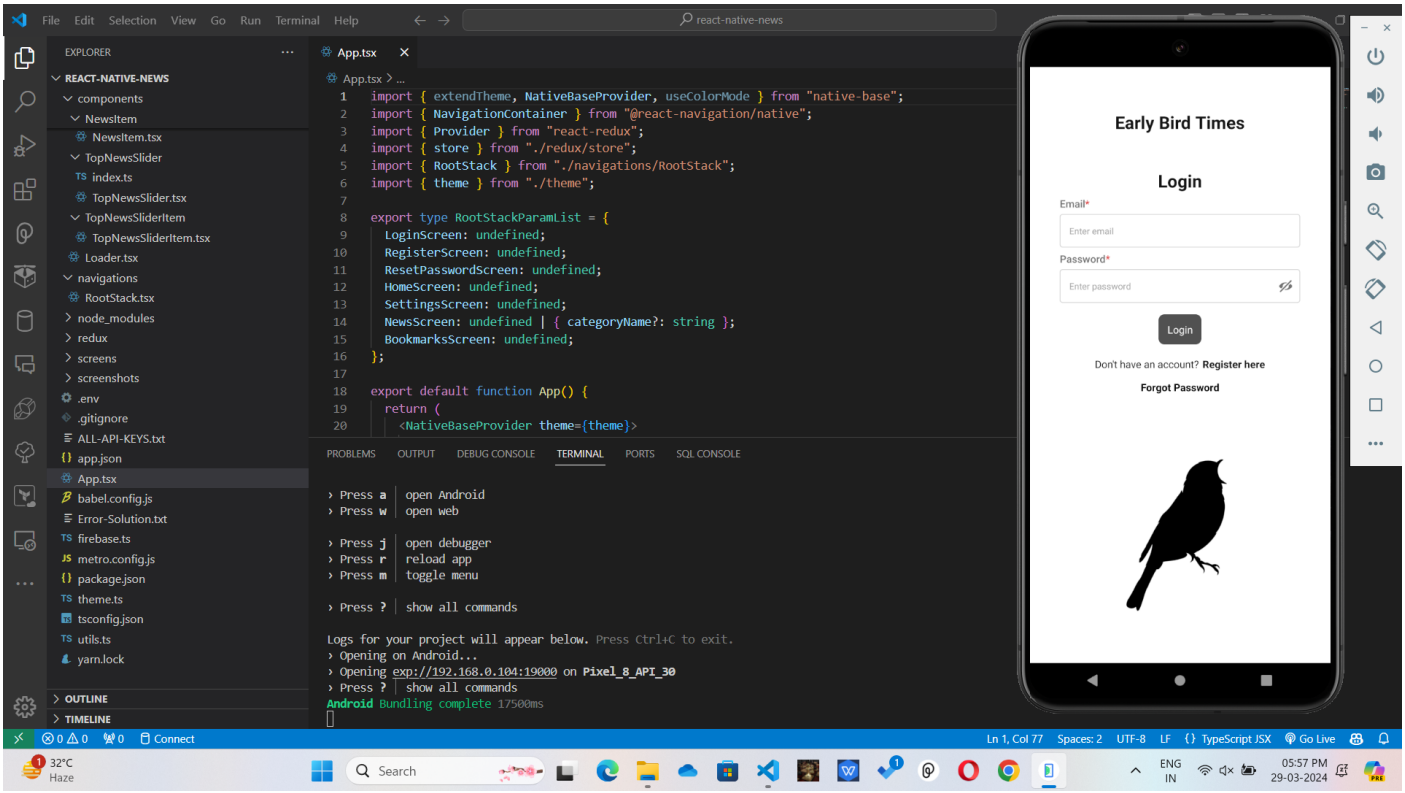
Welcome Page



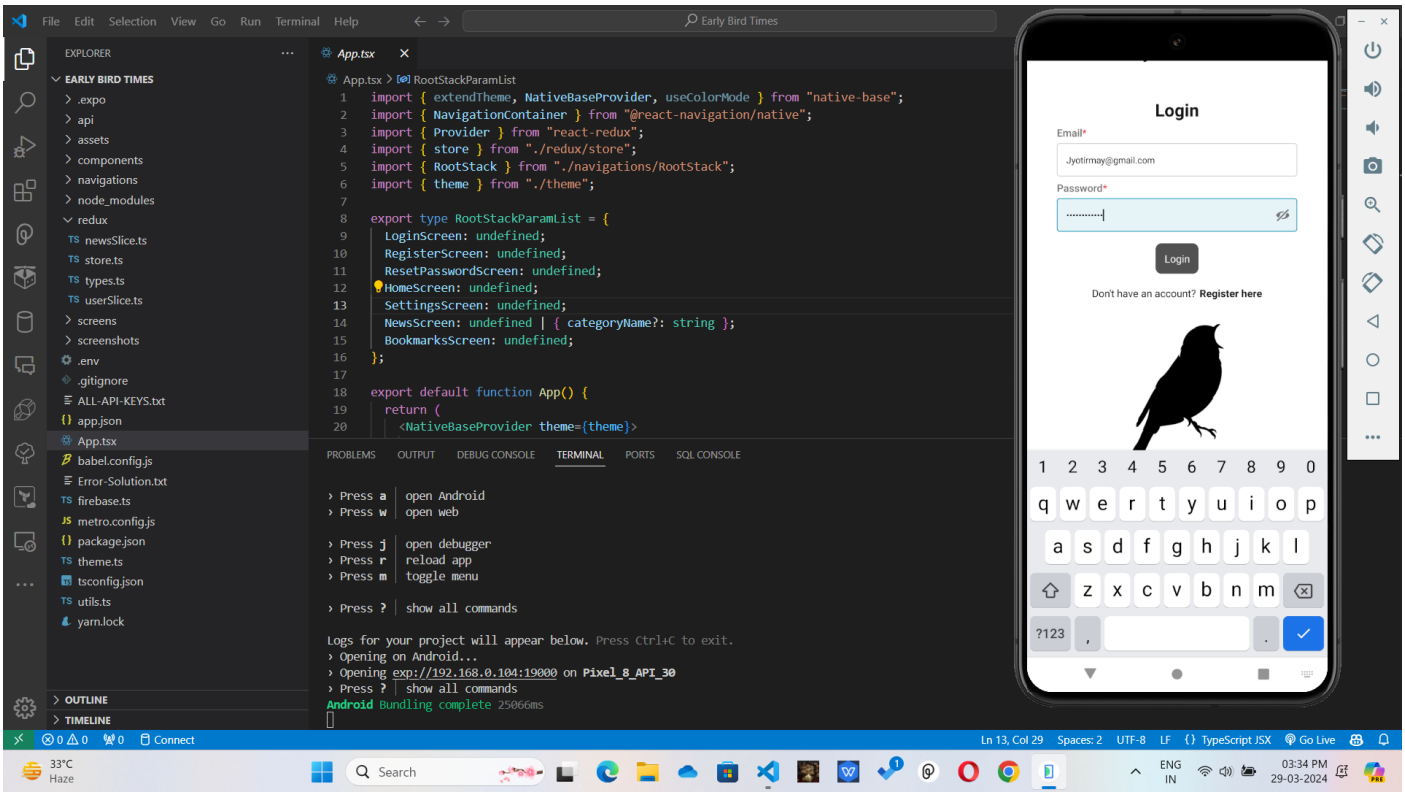
App Entrance Page



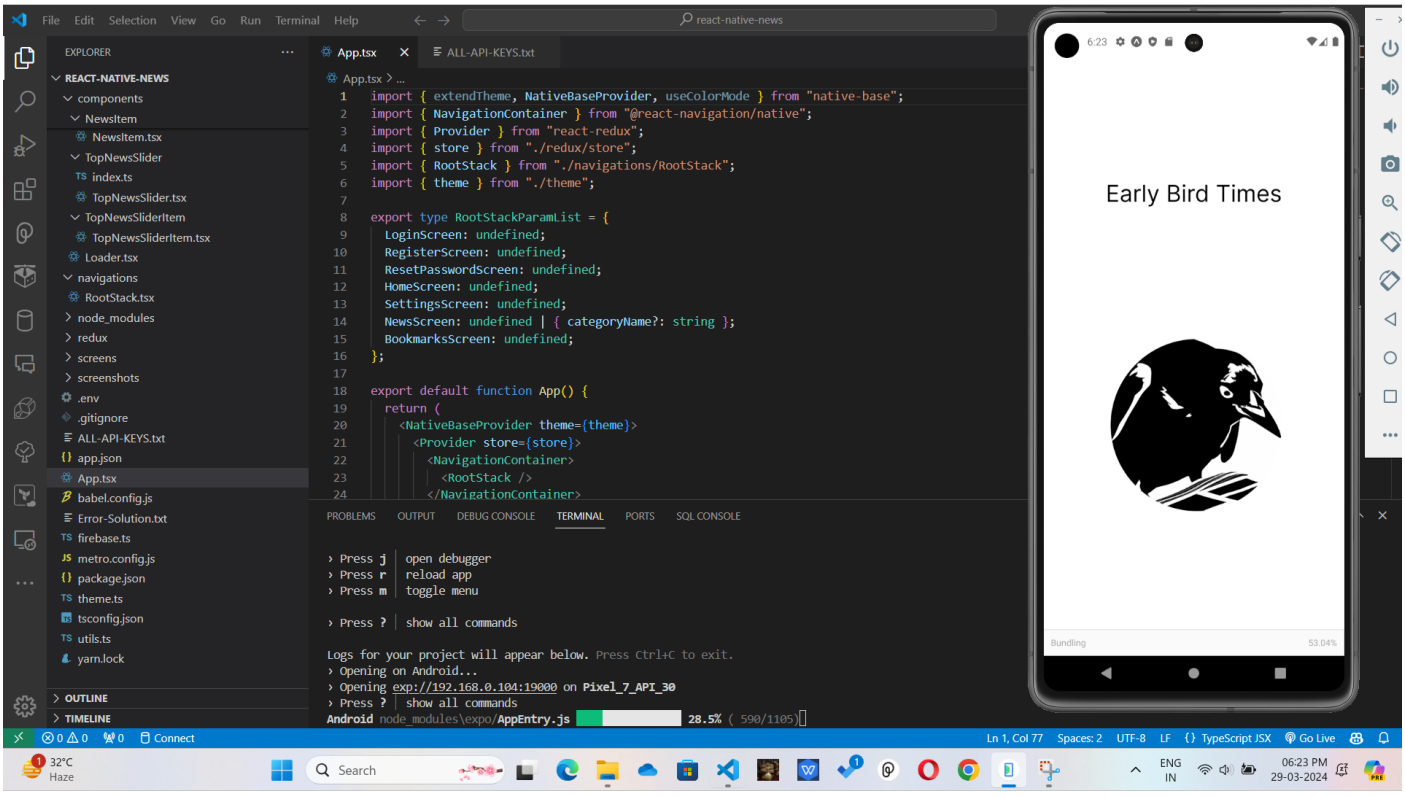
App Login Page



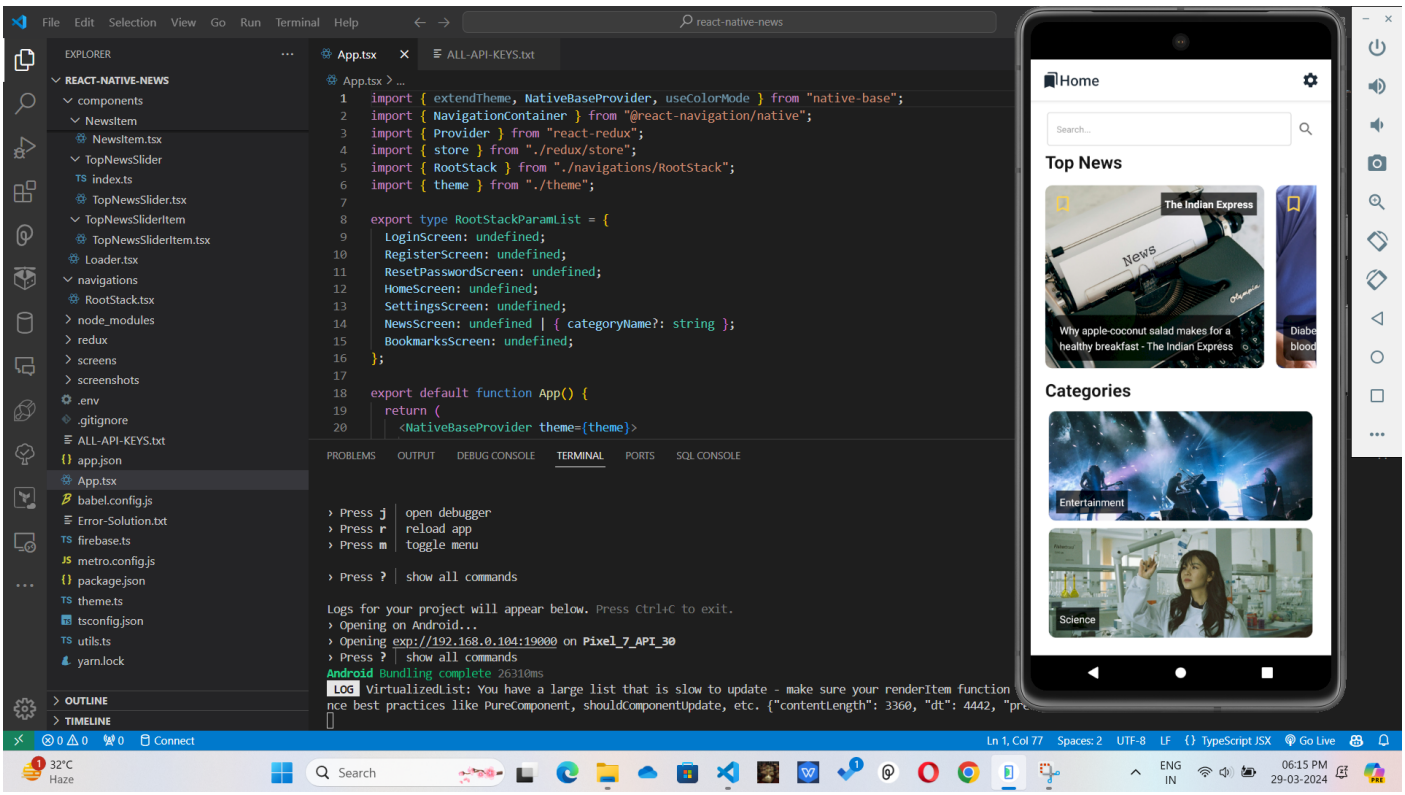
Adding Credentials in Login Page



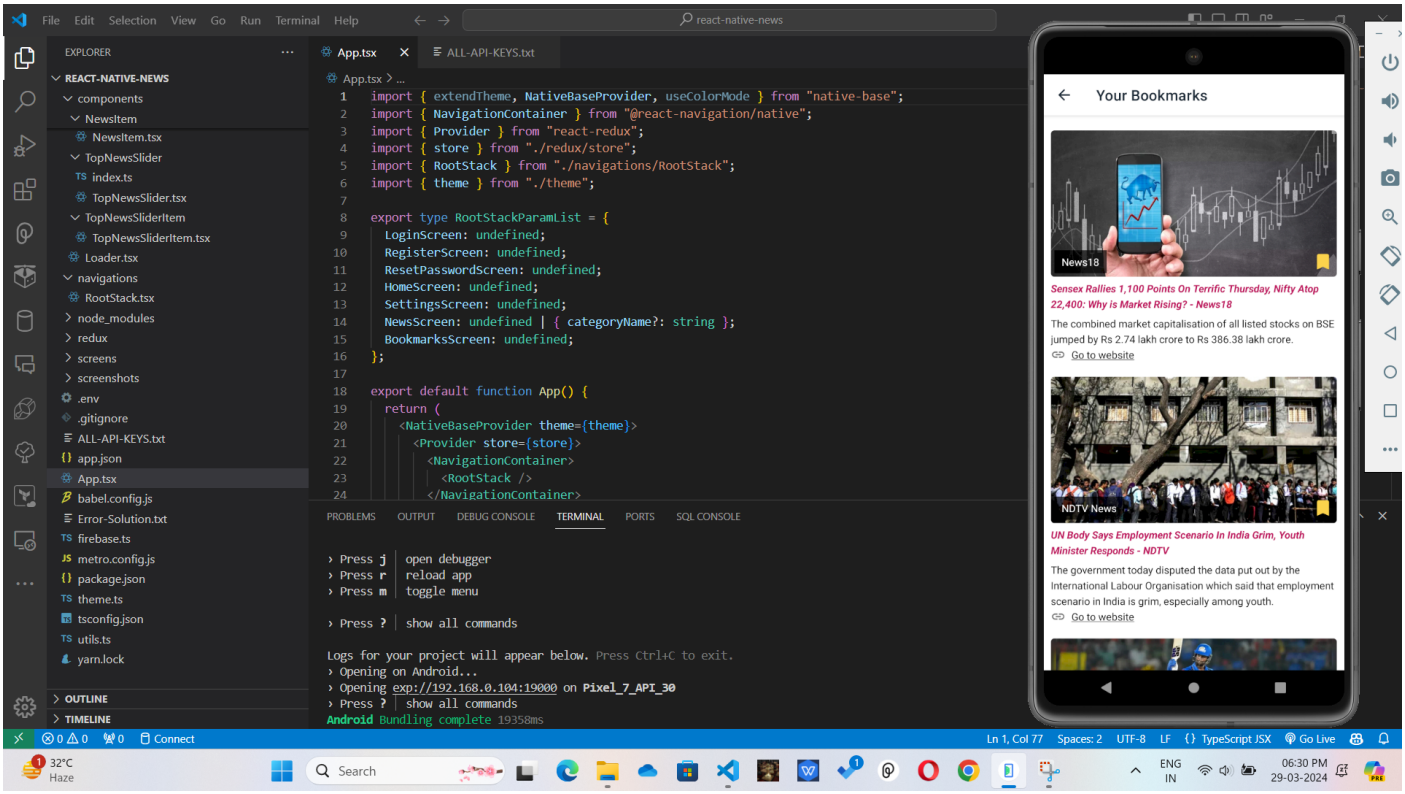
After Input Credential Loading Screen Page



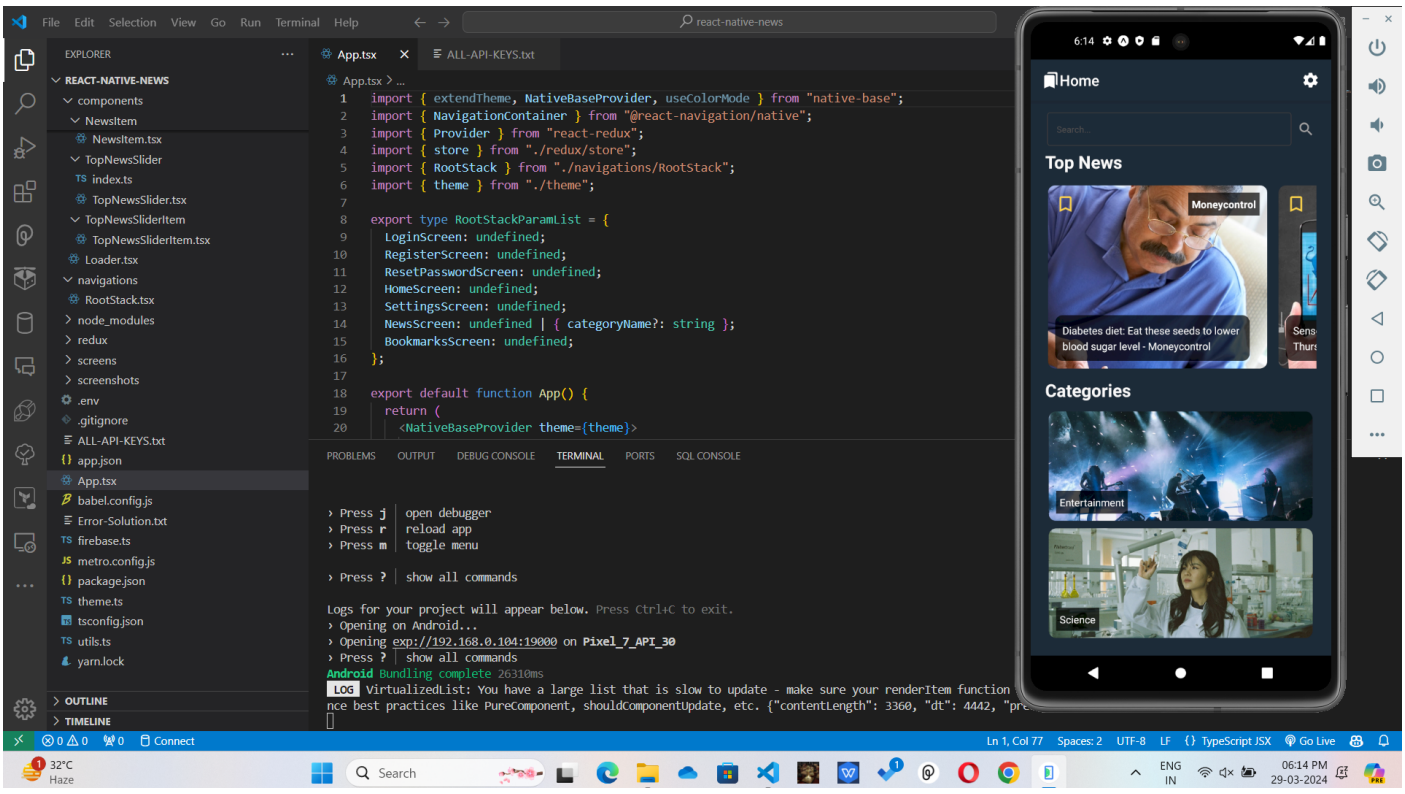
After Credential Input Home Page Image



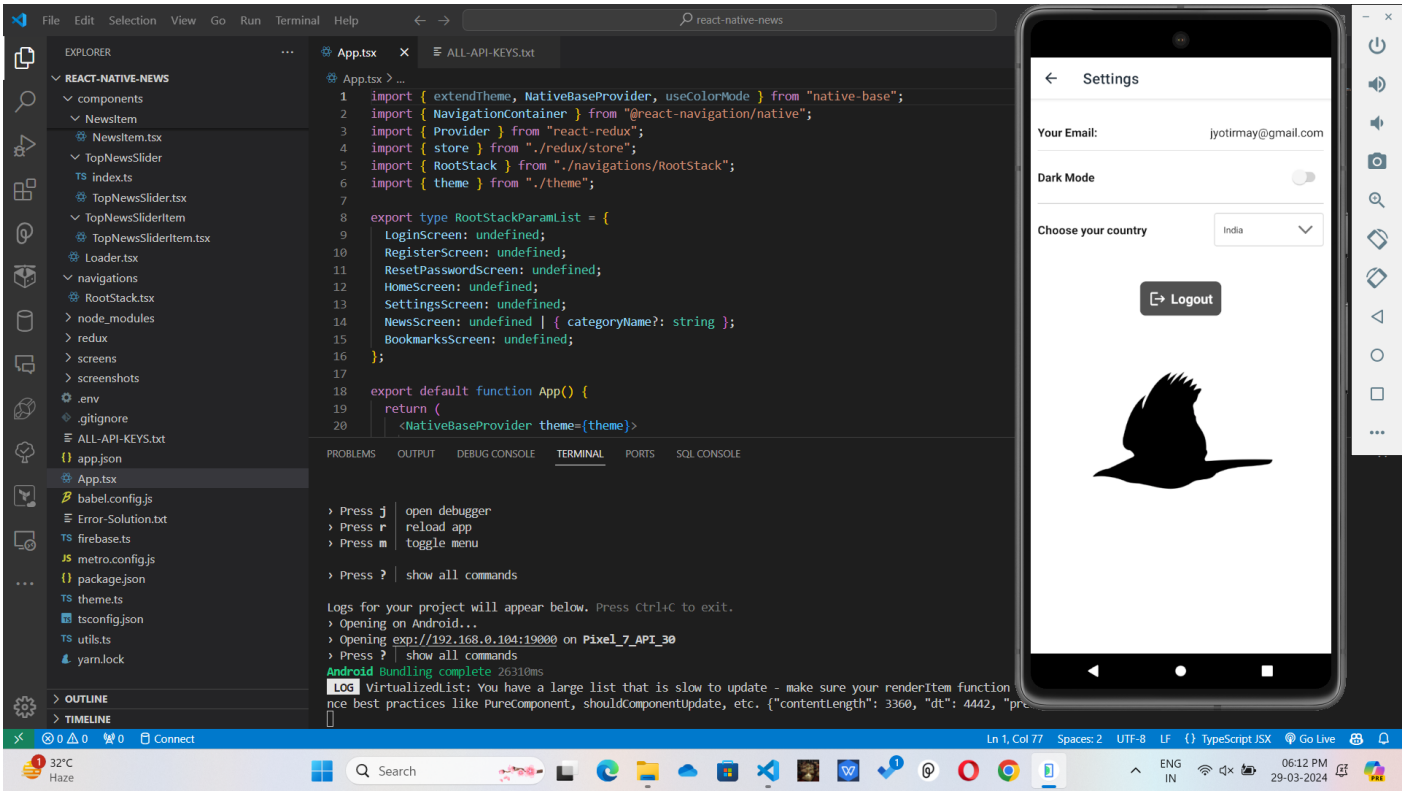
News Bookmarks Section



Light / Dark Mode of Application Features

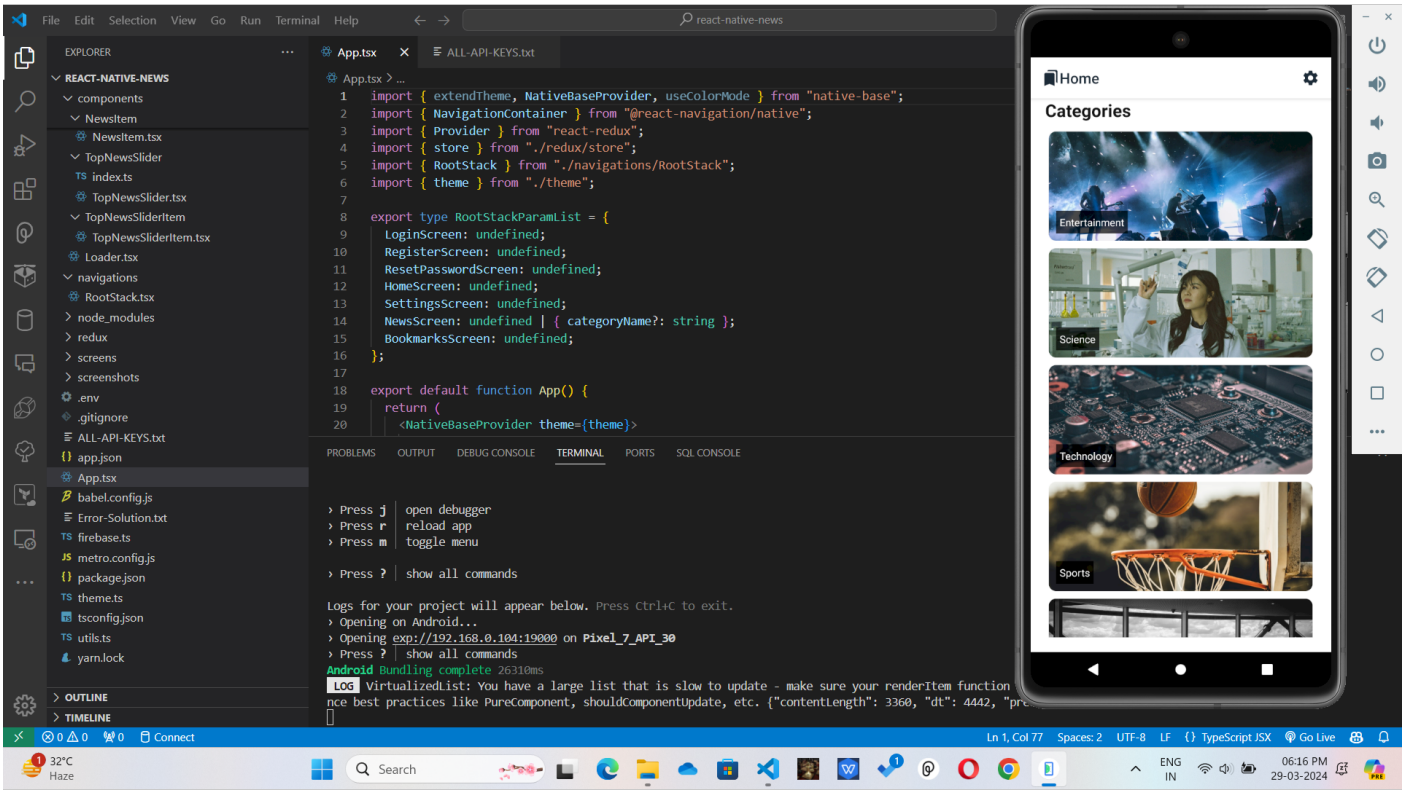


Setting Page of the App

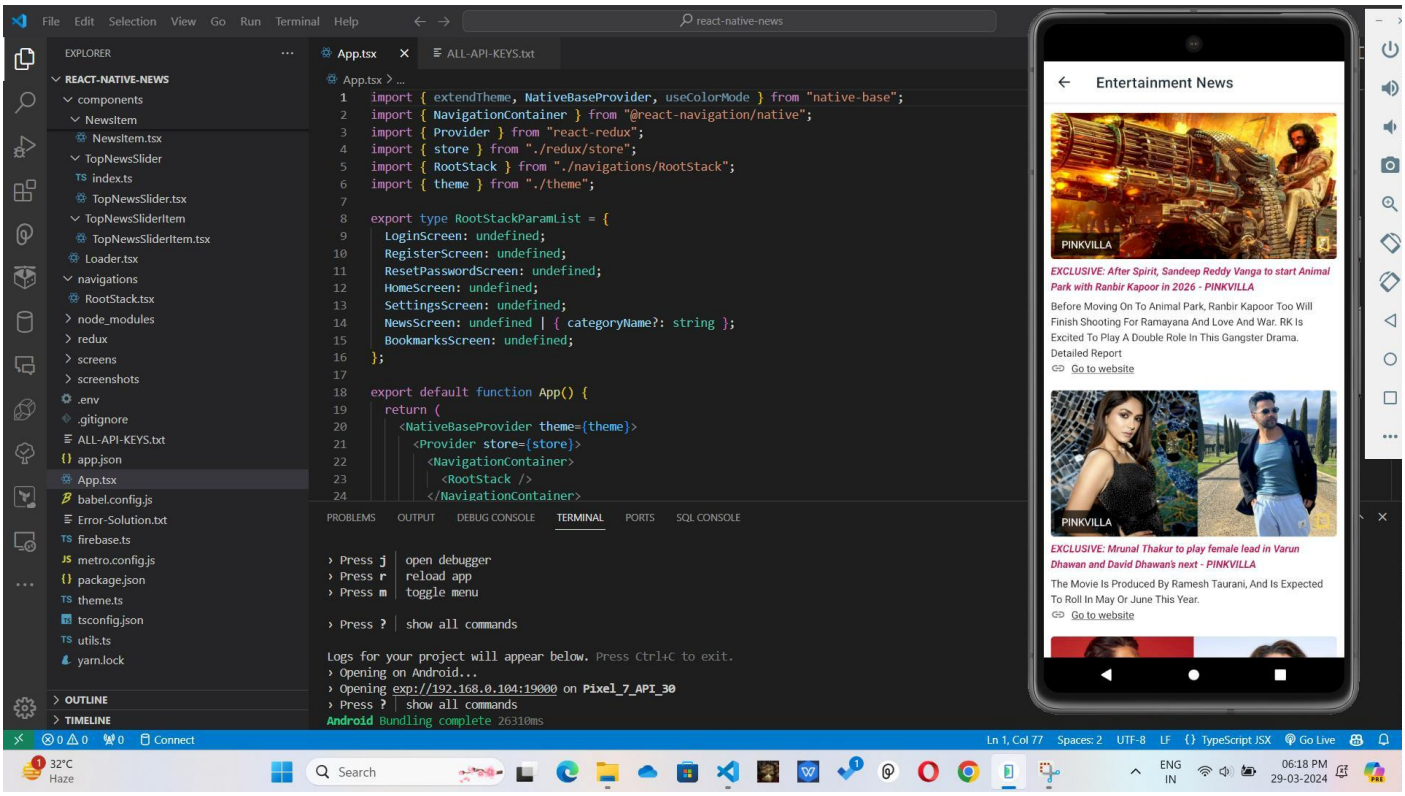


CATEGORIES:-

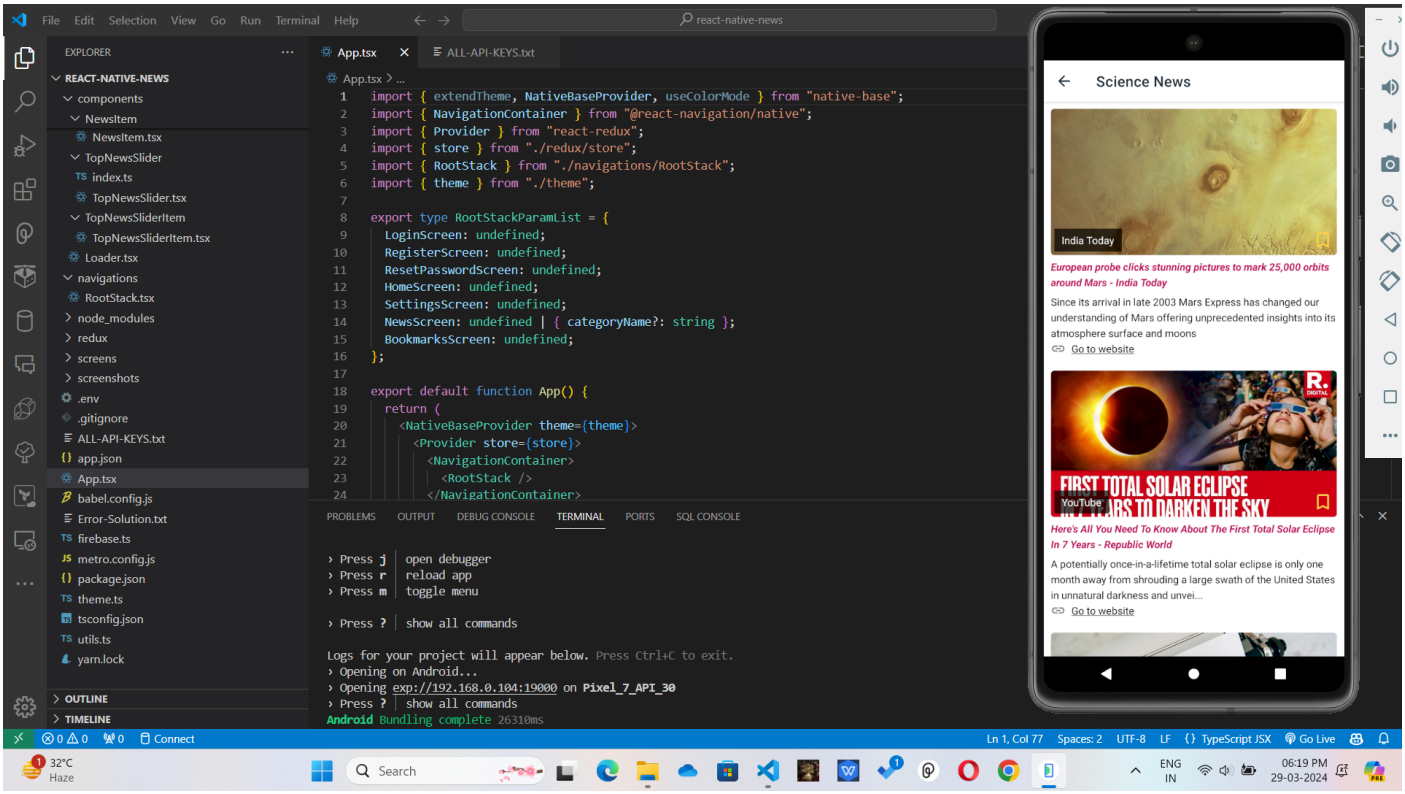
ALL Categories Menu



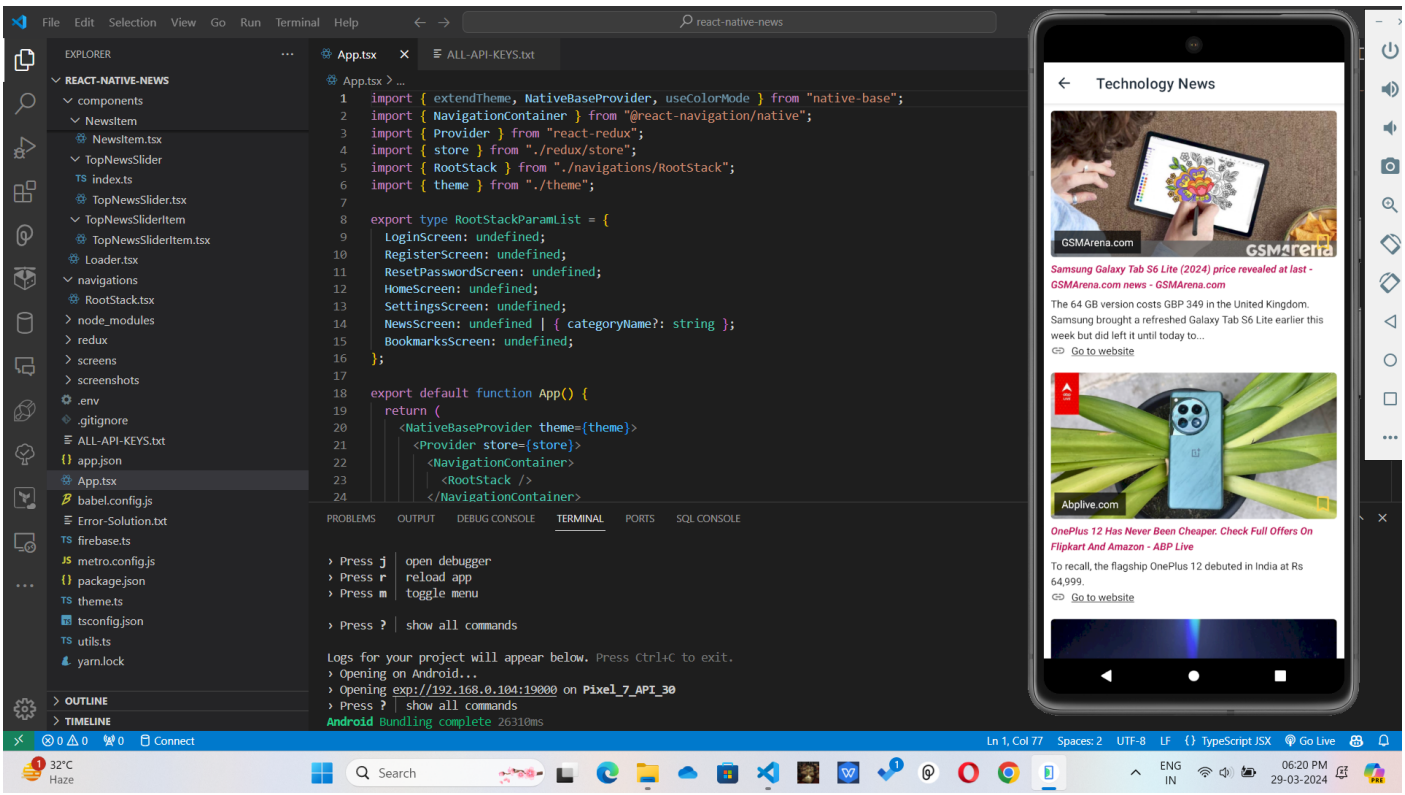
Entertainment News Categories



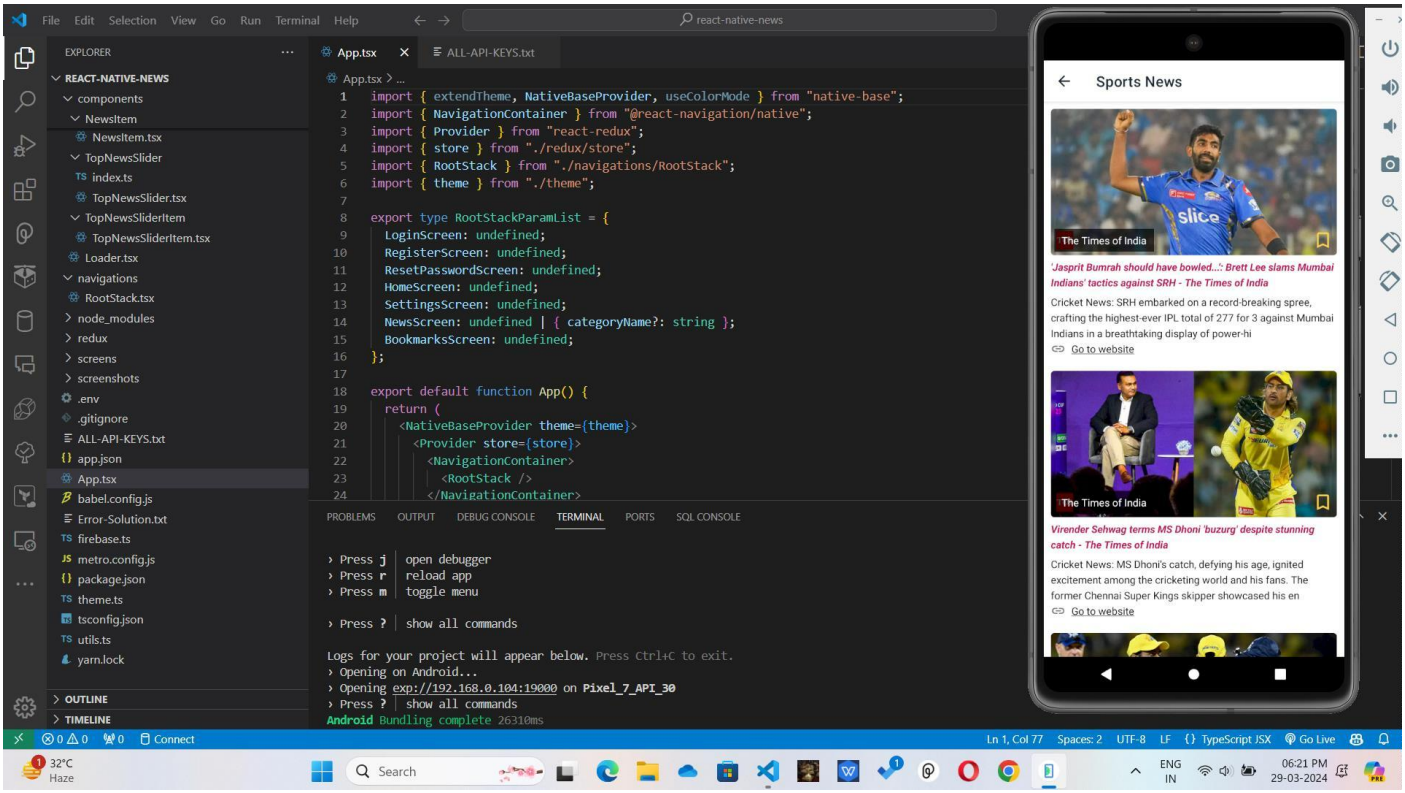
Science News Categories



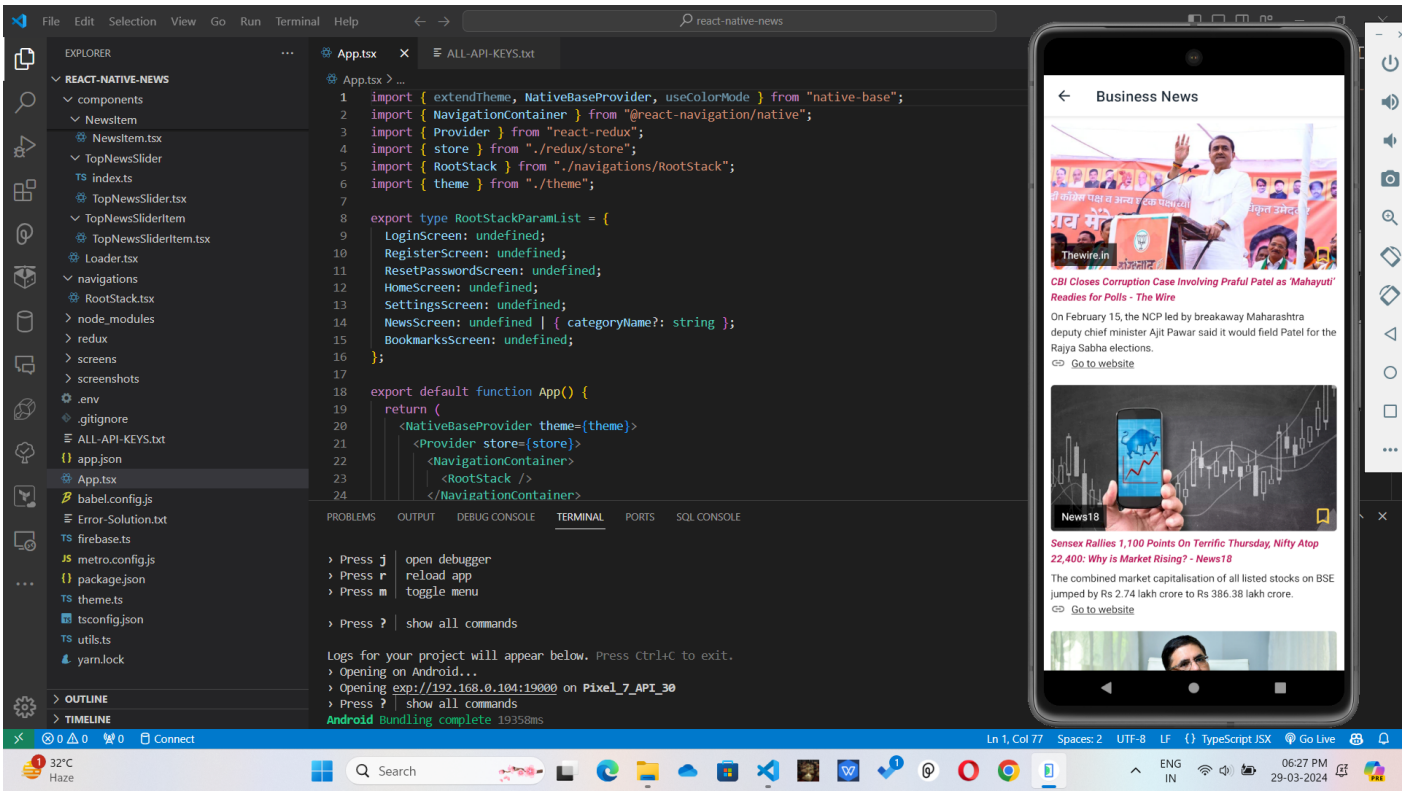
Technology News Categories



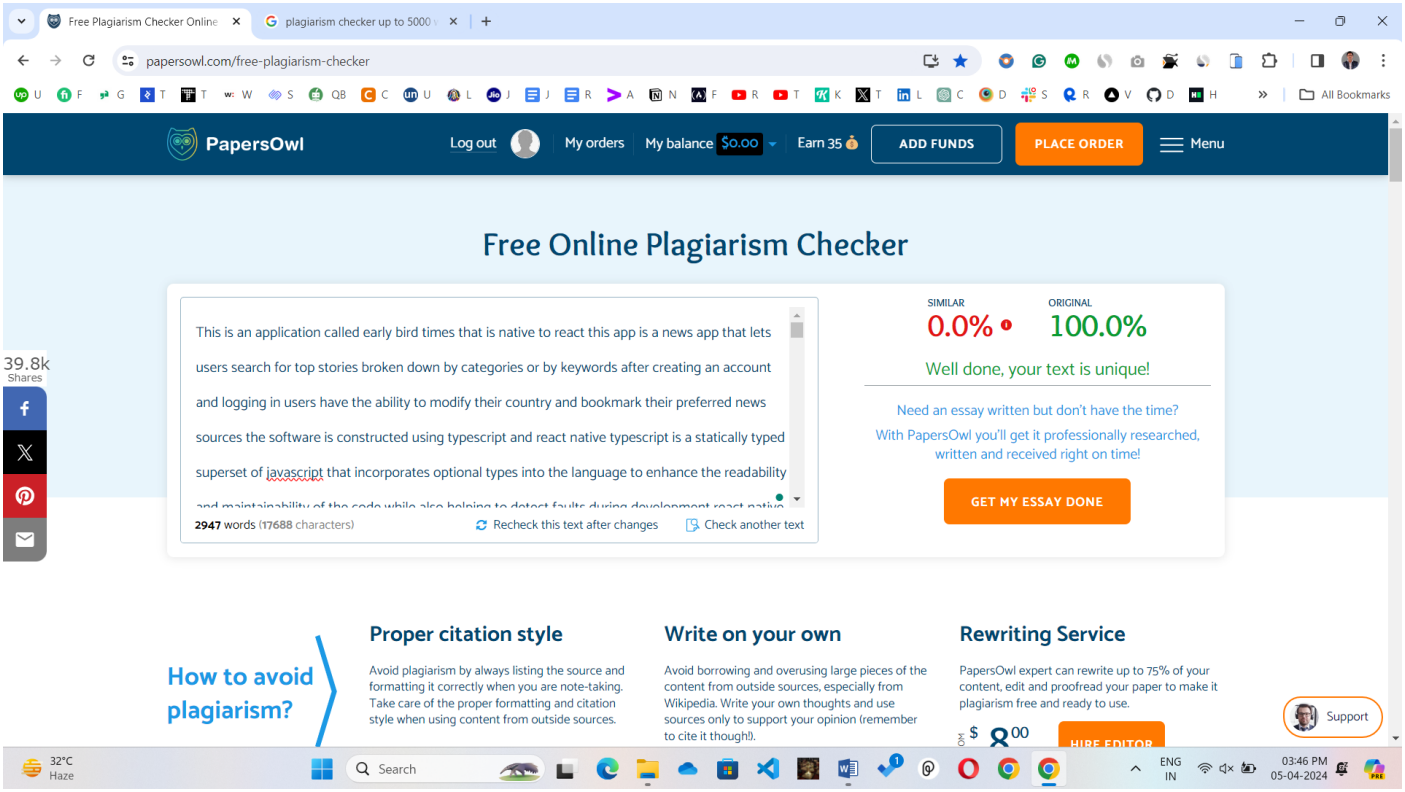
Sports News Categories



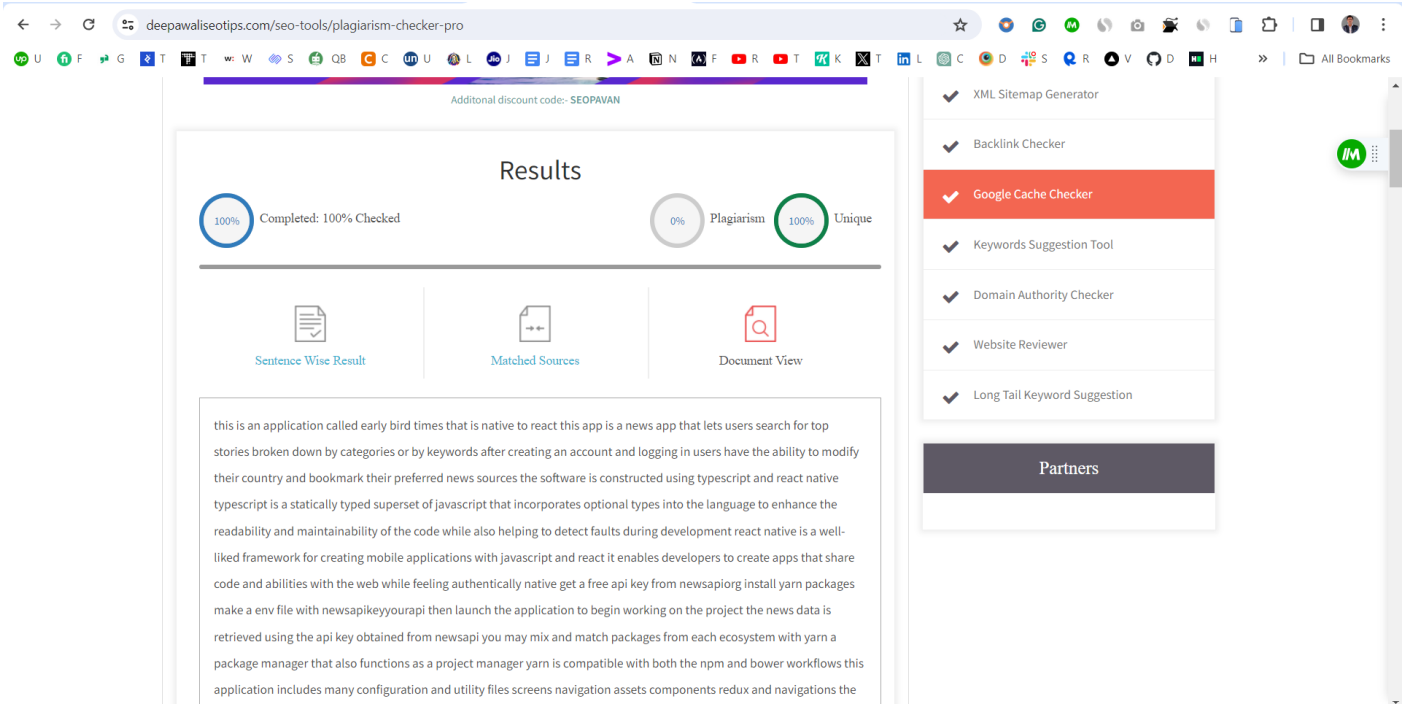
Business News Categories



PLAGRISM REPORT



HERE IS THE RESULT, I GOT 100 % ORIGINAL TEST PASSED SUCCESSFULLY USING PAPEROWL.COM



HERE IS THE RESULT, I GOT 100 % ORIGINAL TEST PASSED SUCCESSFULLY USING HT

REFERENCES (IEEE FORMAT)

1) React Native

Link:- https://reactnative.dev/docs/components-and-apis

2) TypeScript Tutorial

Link:- https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

3) NewsAPI.org

Link:- https://newsapi.org/

4) Yarn

Link:- https://yarnpkg.com/

5) APIs in client-side JavaScript

Link:- https://www.w3schools.com/js/js_api_intro.asp

6) What Are Assets Web Development?

Link:- https://dev.to/pascavld/15-websites-with-free-assets-for-web-developers-5328

7) Core Components and APIs - React Native

Link:- https://reactnative.dev/docs/intro-react-native-components

8) Navigating Between Screens · React Native

Link:- https://reactnative.dev/docs/navigation

9) Comprehensive guide to using Redux in React Native - LogRocket Blog

Link:- https://blog.logrocket.com/comprehensive-guide-to-using-redux-in-react-native/

10) Screens - Expo Documentation

Link:- https://docs.expo.dev/versions/latest/sdk/screens/

GLOSSARY

1) REACT NATIVE	A framework for building mobile applications using JavaScriptand. It allows developers to build apps that feel truly native while sharing skills and code with the web.
2) TYPESCRIPT	A statically typed superset of JavaScript that adds optional types to the language. It can help catch errors during development and improve the readability and maintainability for the code.
3) NEWSAPI	An exceedingly simple HTTP REST API for searching and retrieving live articles from all over the web. It requires a free API key to fetch the news data.
4) YARN	A package manager that doubles down as a project manager. It's compatible with both the npm and bower workflows, allowing you to mix and match packages from each ecosystem.
5) .ENV FILE:	A type of file that is used to store user-specific settings, such as API keys, that should not be included in version control. In this case, it's used to store the NewsAPI key.
6) API FOLDER:	A folder that likely contains the functions for making API calls.
7) ASSETS FOLDER:	A folder that typically includes static files like images, fonts, and other media.
8) COMPONENTS FOLDER	A folder that usually contains reusable React components.
9) NAVIGATION FOLDER	A folder that might contain the navigation setup for the app.
10) REDUX FOLDER	A folder that likely includes the Redux setup for state management.
11) SCREENS FOLDER:	A folder that typically contains the different screens or pages of the app.
12) AUTHENTICATION:	A process that verifies the identity of a user. In this app, it includes features like sign in, sign out, and forget password.

APPENDICIES (QUESTIONNAIRES)

What is the name of the application?
Early Bird Times

What's the application's main function?
With this news app, users may look up the most best stories categorized or keyword-focused.

Which frameworks and languages were employed in the application's development?
TypeScript & React Native are used in the construction of the app.

Why is TypeScript utilized in the program, and what does it mean?
TypeScript is a statically typed superset of JavaScript that enhances the language's readability and maintainability by introducing optional types. This can aid in the detection of problems during development.

What is React Native and why is it used in the application?
React Native is a popular framework for building mobile applications using JavaScript and React It allows developers to build apps that feel truly native while sharing skills and code with the web.

What's the purpose of the API key from newsapi.org?
The API key from newsapi.org is used to fetch the news data.

What is Yarn and why is it used in the application?
Yarn is a package manager that doubles down as a project manager!! and it's compatible with both the npm and bower workflows!! allowing you to mix and match packages from each ecosystem.

What are some of the main folders in the application and what do they contain?
The application contains!! several folders such as api, assets, components, navigations, redux, screens, and several configuration and utility files.

What does the api folder contain?
The api folder likely contains the functions for making API calls.

What does the assets folder contain?

The assets folder typically includes!! static files like images, fonts, and other media.

What does the components folder contain?

The components folder usually contains reusable React components.

What does the navigation folder contain?

The navigation folder might contain the navigation setup for the app.

What does the redux folder contain?

The redux folder likely includes the redux setup for state management?

What does the screens folder contain?

The screens folder typically contains the different screens or pages of the app.

What authentication features does the application contain?

The application contains! authentication features like sign in, sign out, and forget password?

Can users register an account on the application?

Yes, users can register an account.

Can users change their country in the application?

Yes, users can change their country,

Can users add their favorite news to their bookmarks?

Yes, users can add their favorite news to their bookmarks!!!

How can users search for news in the application?

Users can search for top news divided by categories!! or by using keywords.

What is the purpose of the .env file in the project?

The .env file is used to store environment variables like the API key?

What is the role of the package manager in the project?

The package manager is used to manage the project's dependencies!!

What is the benefit of using TypeScript in the project?
TypeScript can help catch errors during development!! and improve the readability and maintainability of the code.

What is the benefit of using React Native in the project?
React Native allows developers to build apps that feel truly native! while sharing skills and code with the web.

What is the purpose of the configuration and utility files in the project?
The configuration and utility files are used to set up and manage various aspects of the project.

How does the application handle state management?
The application likely uses Redux for state management.

What is the purpose of the navigations folder in the project?
The navigations folder might contain the navigation setup for the app, defining how users navigate between different screens or pages.

How does the application ensure a user-friendly experience?
The application provides features like sign in, sign out, and forget password, to help users access the news app!! It also allows users to customize their experience by changing their country and adding favorite news to their bookmarks.

How does the application fetch news data?
The application uses an API key from newsapi.org to fetch the news data.

How does the application handle user authentication?
The application provides features like sign in, sign out, and forget password for user authentication.