



INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FITNESS APPLICATION

¹Dr.A.V.Senthil Kumar, ²Mr.K.Aathi kesavan, Mr.J.Aswin, Mr.S.Sri Ramkumar

¹Director,MCA,M.Phil,PGDCA,Ph.D, ²Final MCA Students

¹PG & Research Department Of Computer Applications(MCA),

¹Hindusthan College of Arts and Science(Autonomous), Coimbatore, India

ABSTRACT

Physical activity, especially in the form of structured exercises, not only helps to improve physical function, but has also been linked to positive outcomes in social and mental well being. However, for several groups of people (such as older adults with physical and cognitive limitations, or postpartum women), regular training - and especially regular training at gym or outdoor - may be inconvenient or impossible. In this application, we review how technology can (and does) facilitate training from home, how it can motivate people to begin and maintain an active lifestyle, and how it can be effective in achieving results (such as better strength and balance). Because older adults represent such a specific and important class of people for which home training may be the most convenient (and sometimes only) option, we specifically analyze research and applications based on their suitability for older adults. Besides discussing current technologies and research, we also underline limitations and research gaps in IT-based home training solutions in general and for older adults in particular. The application will contain different types of training based on user needs and have Good diet food details that are most helpful for fitness.

SYNOPSIS

The Fitness Application helps people to maintain a good and healthy lifestyle. Nowadays we all have to work remotely, so not maintaining our health leads to critical health issues. For that we are proposing this application contains exercise steps and guidance based on user needs. There we have diet food details and instructions to make them.

The frontend of this project is React Native, CSS, java script and the backend is Firebase Authentication which makes it easy for creating and generating code. Windows10 is used as an Operating System and hosting the web application in Expo.

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

Apps are small, specialised programs (applications) designed to be downloaded onto a mobile device, such as a smartphone or tablet PC. Fitness apps are designed specifically to assist with exercise, other types of physical training, nutrition and diet, or related fitness topics. Because fitness apps, a part of a larger group of apps called health apps, are available to be used at home and while away, they are part of a healthcare movement called mobile health (Fitness Application).

The purpose of a fitness app is to provide the user with instructions and examples of one or more types of exercise, physical activity, nutritional programs, or some other fitness topic. Many fitness apps are available to download from the internet. Some are used to count calories, others record statistics on workouts or collect data on walks, runs, and bike rides. Some fitness apps connect the user to a personal trainer or nutritionist to help with areas of concerns when using a specific fitness routine or just generally with workouts. Further, some fitness apps provide a coordinated series of songs, each having the same beat when doing such workouts as running and fitness classes.

1.1. MODULE DESCRIPTION:

- LOGIN, REGISTRATION
- HOMEPAGE
 - BULK
 - LEAN
 - CARDIO - FAT LOSS
 - DIET
- SUB CATEGORY PAGE
- DETAILS PAGE - WORKOUT VIDEO
- DIET DETAILS

2. SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATION

- | | | |
|----------------------|---|----------------------|
| ● System | : | INTEL |
| ● Hard Disk | : | 500 GB. |
| ● Monitor resolution | : | 1024 x 768 or higher |
| ● Keyboard | : | 108 keys |
| ● Mouse | : | Logitech. |
| ● Ram | : | 8 GB. |

2.2 SOFTWARE SPECIFICATION

- Operating system : Windows or Linux.
- Software Tools : Android studio, VS Code
- Front End : React Native
- Back End : Firebase Authentication Cloud

2.2 ABOUT THE SOFTWARE

ABOUT ANDROID

Android's kernel is based on the Linux kernel and has further architecture changes by Google outside the typical Linux kernel development cycle. Android does not have a native X Window System nor does it support the full set of standard GNU libraries, and this makes it difficult to port existing Linux applications or libraries to Android.

Certain features that Google contributed back to the Linux kernel, notably a power management feature called wake locks, were rejected by mainline kernel developers, partly because kernel maintainers felt that Google did not show any intent to maintain their own code. Even though Google announced in April 2010 that they would hire two employees to work with the Linux kernel community, Greg Kroch-Hartman, the current Linux kernel maintainer for the -stable branch, said in December 2010 that he was concerned that Google was no longer trying to get their code changes included in mainstream Linux. Some Google Android developers hinted that "the Android team was getting fed up with the process", were a small team and had more urgent work to do on Android.

However, in September 2010, Linux kernel developer Rafael J. Wysocki added a patch that improved the mainline Linux wakeup events framework. He said that Android device drivers that use wake locks can now be easily merged into mainline Linux, but that Android's opportunistic suspend features should not be included in the mainline kernel.

Features:

Handset layouts:

The platform is adaptable to larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 2.0 specifications, and traditional smartphone layouts.

Storage:

We have used latest Firebase Cloud Authentication Process for user storage.

Connectivity:

Android supports connectivity technologies including GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth.

Multiple language support

Android supports multiple languages.

Web browser

The web browser available in Android is based on the open-source Webkit layout engine, coupled with Chrome's V8 JavaScript engine. The browser scores 100/100 on the Acid3 test on Android 4.0.

Java support

While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executables and run on a specialised virtual machine designed specifically for Android and optimised for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party applications.

Media support Android supports the following audio/video/still media formats: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, FLAC, WAV, JPEG, PNG, GIF, BMP.

Additional hardware support

Android can use video / still cameras, touchscreens, GPS, accelerometers, gyroscopes, barometers, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers, accelerated 2D bit blots (with hardware orientation, scaling, pixel format conversion) and accelerated 3D graphics.

Multi-touch

Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology at the time). Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively.

Multitasking

Multitasking of applications is available.

Voice based features

Google search through voice has been available since initial release. Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards.

Tethering

Android supports tethering, which allows a phone to be used as a wireless/wired Wi-Fi hotspot. Before Android 2.2 this was supported by third-party applications or manufacturer customizations.

Screen capture

Android supports capturing a screenshot by pressing the power and volume-down buttons at the same time. Prior to Android 4.0, the only methods of capturing a screenshot were through manufacturer and third-party customizations or otherwise by using a PC connection (DDMS developer's tool). These alternative methods are still available with the latest Android.

External storage

Most Android devices include microSD slot and can read microSD cards formatted with FAT32, Ext3fs or Ext4fs file system. To allow use of high-capacity storage media such as USB flash drives and USB HDDs, many Android tablets also include USB 'A' receptacle. Storage formatted with FAT32 is handled by Linux Kernel VFAT driver, while 3rd party solutions are required to handle other popular file systems such as NTFS, HFS Plus and exFAT

3. SYSTEM STUDY

3.1 EXISTING SYSTEM

- In the existing application there will be only exercise steps and details only.

Disadvantage:

- There is no exercise based on body conditions.
- There is a gap in security systems for the user.
- There will not be a user-friendly manner.
-

3.2. PROPOSED SYSTEM

- Mainly, this project contains Diet and Exercise as a single application.
- And also Videos to clearly explain fitness steps.
- We have added Diet and cooking instructions to clear steps for cooking healthy foods.

Advantage:

- Added Diet and Exercise in single application.
- There will be a user-friendly manner.

4. SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

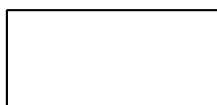
A data flow diagram is a graphical tool used to describe and analyse movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processing, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level

The idea behind the explosion of a process into more processes is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analysts to understand the process.

DFD SYMBOLS

In the DFD, there are four symbols

- A rectangle defines a source(originator) or destination of system data
- An arrow identifies data flow. It is the pipeline through which the information flows
- A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
- An open rectangle is a data store, data at rest or a temporary repository of data



Source or Destination of data



Data flow



Data Store



Process

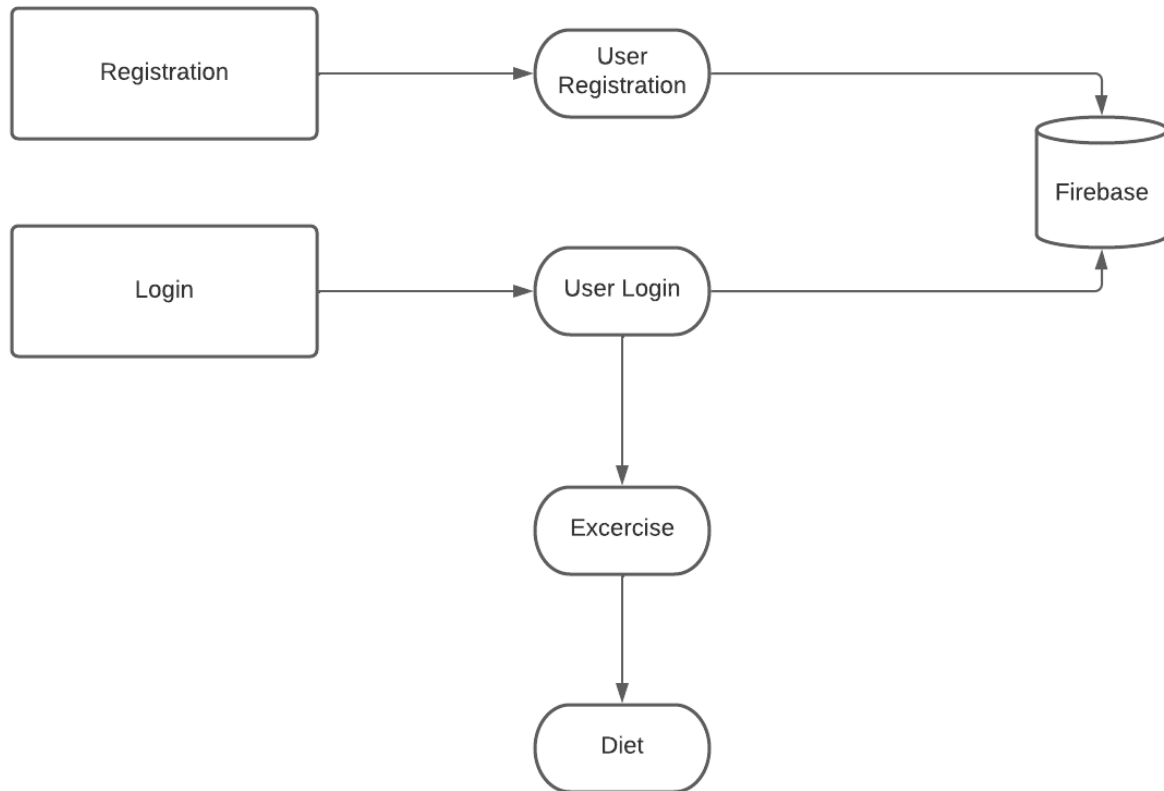
SALIENT FEATURES OF DFD'S

- The DFD shows flow of data, not of control loops and decisions are controlled considerations that do not appear on a DFD.
- The DFD does not indicate the time factor involved in any process whether the data flow takes place daily, weekly, monthly or yearly.

Level-0



Level - 1



4.2. DATABASE DESIGN

Easy sign-in with any platform

Firebase Authentication aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end identity solution, supporting email and password accounts, phone auth, and Google, Twitter, Facebook, and GitHub login, and more.

Comprehensive security

Built by the same team that developed Google Sign-in, Smart Lock and Chrome Password Manager, Firebase security applies Google's internal expertise of managing one of the largest account databases in the world.

Fast implementation

It can take months to set up your own auth system, and it requires an engineering team to maintain that system in the future. Set up the entire authentication system of your app in under 10 lines of code, even handling complex cases like account merging.

Part of the Firebase platform

Firebase helps you develop high-quality apps, grow your user base, and earn more money. Each feature works independently, and they work even better together.

In the present era, user authentication is one of the most important requirements for Android apps. It is essential to authenticate users, and it is much harder if we have to write all this code on our own. This is done very easily with the help of Firebase.

- Being able to authenticate our users securely, it offers a customized experience to them based on their interests and preferences.
- We can ensure that they have no problems accessing their private data while using our app from multiple devices.
- Firebase Authentication provides all the server-side stuff for authenticating the user. Firebase Authentication becomes easy with SDK. It makes the API easy to use.
- Firebase Authentication also provides some user interface libraries which enable screens for us when we are logging it.
- Firebase authentication supports authentication using a password, phone numbers, popular identity provider like Google, Facebook, and Twitter, etc.
- We can sign in users to our app by using the FirebaseAuthUI.
- It handles the UI flows for signing in users with an email address and password, phone numbers, and popular providers, including Google sign-In and Facebook Login.
- It can also handle cases like account recovery.
- It is not required to design a UI since it is already provided for us. It means we don't have to write about the activities.
- We can also sign-in users using the FirebaseAuth SDK to integrate one or several sign-in methods into our app manually.

How Authentication Works?

- We first get authentication credentials from the user to sign a user into our app.
- Credentials can be the user's email address and password.
- The credential can be an OAuth token from an identity provider.
- We then pass these credentials to the Firebase Authentication SDK. Backend services will then verify those credentials and return a response to the client.
- After a successful sign in:
- We can access the user's access to data stored in other Firebase products.
- We can access the user's basic profile information.
- We can use the provided authentication token to verify the identity of users in our own backend services.
- An authenticated user can read and write data to the Firebase Real-time Database and Cloud Storage.
- We can control the access of those authenticated users by modifying the Firebase Database Rules and Storage Security Rules.

User Lifecycle

- An Auth listener gets notified in the following situation
- The Auth object finishes initializing, and a user was already signed in from a previous session or has been redirected from an identity provider's sign-in flow
- A user signs in.
- A user signs out.
- The current user's access token is refreshed:
 - The access token expires.
 - The user changes their password.
 - The user re-authenticates

4.3 INPUT DESIGN

Input design is the process of converting an external user oriented description of input system into a machine oriented format. The source document is prepared for the input of data in order to make the entry accurate and impact the source document was prepared. The data elements were sent out in a system in which the data entry operator would easily follow.

- To produce a cost effective method.
- To get the highest level of accuracy.
- To ensure that the input is acceptable and understandable by the people who are using it.
- With the above objectives the major activities that were done during the input design
- The data is collected from its source.
- Data is converted to the mobile acceptable form.
- The converted data was verified.

4.4 OUTPUT DESIGN:

Output design is a process that involves designing necessary output that have to be given to various users according to their requirements. Efficient, intelligible output design will improve the system relationship with the user and help indecision making. Since the reports are directly required by the management for taking decisions and to draw conclusions, they must be designed with almost care to the user. The options for the output and report are given in the system menu. When designing output, system analyst must accomplish the following ;

- Determine the information to present.
- Arrange the present of information acceptable format.
- Determine how to distribute the output.

5. SYSTEM TESTING AND IMPLEMENTATION

WINDOWS OPERATING SYSTEM:

"Windows" redirects here. For the part of a building, see the window. For other uses, see Windows (disambiguation).

Microsoft Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.

Microsoft introduced an operating environment named Windows on November 20, 1985 as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUI).

Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984.

Windows XP

Windows XP would also introduce a redesigned user interface (including an updated Start menu and a "task-oriented" Windows Explorer), streamlined multimedia and networking features, Internet Explorer 6, integration with Microsoft's .NET Passport services, modes to help provide compatibility with software designed for previous versions of Windows, and Remote Assistance functionality. Windows XP was now marketed in two main editions: the "Home" edition was targeted towards consumers, while the "Professional" edition was targeted towards business environments and power users, and included additional security and networking features.

Home and Professional were later accompanied by the "Media Centre" edition (designed for home theatre PCs, with an emphasis on support for DVD playback, TV tuner cards, DVR functionality, and remote controls), and the "Tablet PC" edition (designed for mobile devices meeting its specifications for a tablet computer, with support for stylus pen input and additional pen-enabled applications).

Mainstream support for Windows XP ended on April 14, 2009. Extended support will continue until April 8, 2014.

After Windows 2000, Microsoft also changed its release schedules for server operating systems; the server counterpart of Windows XP, Windows Server 2003, was released in April 2003. It was followed in December 2005 by Windows Server

Windows Vista, 7 and 8

Windows Vista was released on November 30, 2006 for volume licensing and January 30, 2007 for consumers. It contained a number of new features from a redesigned shell and user interface to significant technical changes, with a particular focus on security features. It was available in a number of different editions, and has been subject to some criticism. Vista's server counterpart, Windows Server 2008 was released in early 2008.

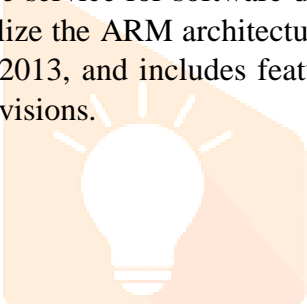
On July 22, 2009, Windows 7 and Windows Server 2008 R2 were released as RTM (release to manufacturing) while the former was released to the public 3 months later on October 22, 2009. Unlike its predecessor, Windows Vista, which introduced a large number of new features, Windows 7 was intended to be a more focused, incremental upgrade to the Windows line, with the goal of being compatible with applications and hardware with which Windows Vista was already compatible. Windows 7 has multi-touch support, a redesigned Windows shell with an updated taskbar, a home networking system called Home Group, and performance improvements.

Windows 8, the successor to Windows 7, was released generally on October 28, 2012. A number of significant changes were made on Windows 8, including the introduction of a user interface based around Microsoft's Metro design language with optimizations for touch-based devices such as tablets and all-in-one PCs. Windows Store service for software distribution, and a new variant known as Windows RT for use on devices that utilize the ARM architecture. An update to Windows 8, called Windows 8.1, was released on October 17, 2013, and includes features such as new live tile sizes, deeper SkyDrive integration, and many other revisions.



React Native

JavaScript



JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. Read more about JavaScript.

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about API specifics to Web pages, please see Web APIs and DOM.

The standards for JavaScript are the ECMAScript Language Specification (ECMA-262) and the ECMAScript Internationalization API specification (ECMA-402). The JavaScript documentation throughout MDN is based on the latest draft versions of ECMA-262 and ECMA-402. And in cases where some proposals for new ECMAScript features have already been implemented in browsers, documentation and examples in MDN articles may use some of those new features.

Do not confuse JavaScript with the Java programming language. Both "Java" and "JavaScript" are trademarks or registered trademarks of Oracle in the U.S. and other countries. However, the two programming languages have very different syntax, semantics, and use.

Create native apps for Android and iOS using React

React Native combines the best parts of native development with React, a best-in-class JavaScript library for building user interfaces.

Use a little—or a lot. You can use React Native today in your existing Android and iOS projects or you can create a whole new app from scratch.

Written in JavaScript—rendered with native code

React primitives render to native platform UI, meaning your app uses the same native platform APIs other apps do.

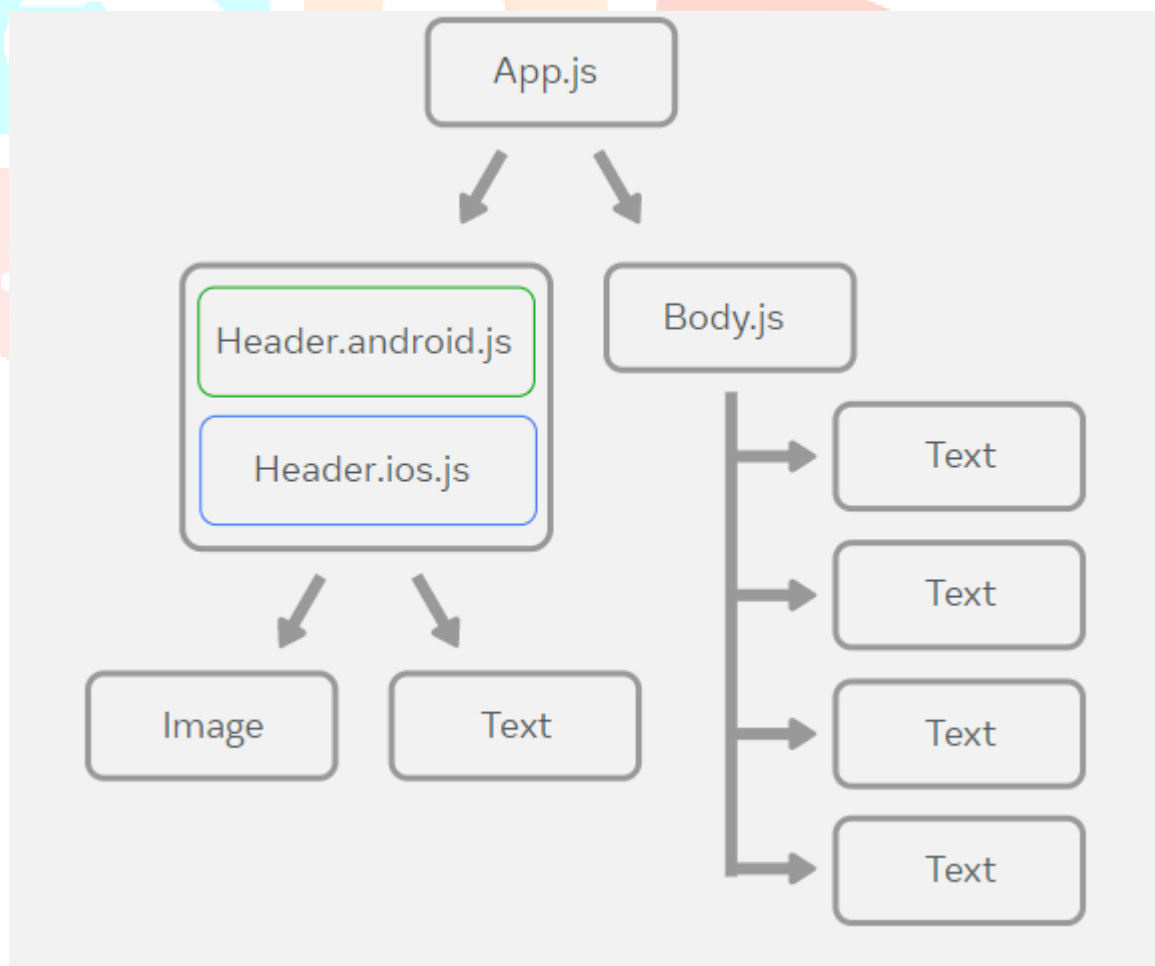
Many platforms, one React. Create platform-specific versions of components so a single codebase can share code across platforms. With React Native, one team can maintain two platforms and share a common technology—React.

Native Development For Everyone

React Native lets you create truly native apps and doesn't compromise your users' experiences. It provides a core set of platform agnostic native components like View, Text, and Image that map directly to the platform's native UI building blocks.

Seamless Cross-Platform

React components wrap existing native code and interact with native APIs via React's declarative UI paradigm and JavaScript. This enables native app development for whole new teams of developers, and can let existing native teams work much faster.



Design

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates.

Instead, the SQLite library is linked in and thus becomes an integral part of the application program. The library can also be called dynamically. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication.

Features

Following are the features of React Native –

- React – This is a Framework for building web and mobile apps using JavaScript.
- Native – You can use native components controlled by JavaScript.
- Platforms – React Native supports IOS and Android platform.

React Native Advantages

Follow are the advantages of React Native –

- JavaScript – You can use the existing JavaScript knowledge to build native mobile apps.
- Code sharing – You can share most of your code on different platforms.
- Community – The community around React and React Native is large, and you will be able to find any answer you need.

React Native Limitations

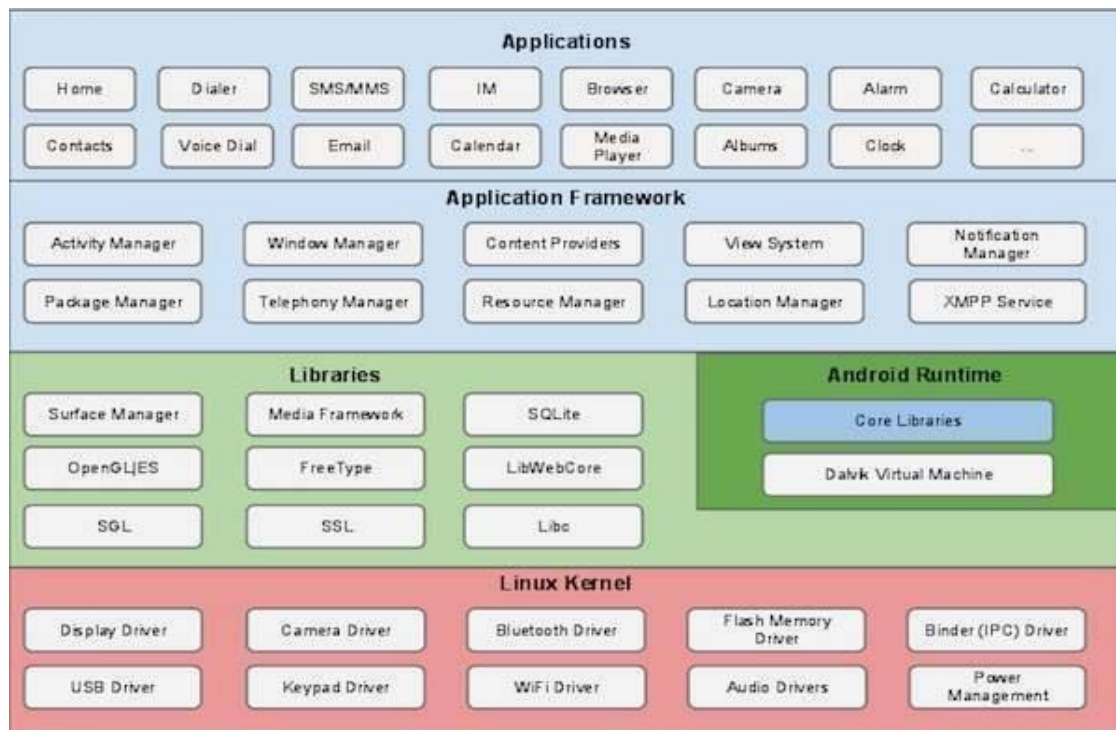
Following are the limitations of React Native –

- Native Components – If you want to create native functionality which is not created yet, you will need to write some platform specific code.

Android

Android is a mobile phone operating system. It was originally developed by Android Inc, which was acquired by Google in July 2005.⁷ Today, development is overseen by the Android Open Source Project (AOSP), led by Google. The AOSP is “tasked with the maintenance and further development of Android”. As of the 3rd Quarter of 2010 Android has a market share of 25% making it the second most popular phone operating system of the market (second only to Nokia’s Symbian). This is a major rise from the 3.5% share Android had in 3Q2009.

Architecture



Android is based on the Linux Kernel. Android Developers are able to access all the components of the Application Framework used by core applications when creating an application. These features include the Location Manager, Bluetooth, the Accelerometer, and Email etc.

The Android software stack can be subdivided into five layers:

- ☐ Linux Kernel (Low Level)
- ☐ Libraries
- ☐ Android Runtime
- ☐ Application Framework
- ☐ Applications (High Level)

Application Framework

An Android Application has four parts- Activities, Services, Broadcast Receivers and Content Providers.

- **Activities** are the visual interfaces for each task in the application. Each activity, despite being linked, is an independent class.
- **Services** are the background tasks that don't have a user interface. Services might be linked to one or more activities.
- **The Broadcast Receiver** receives and reacts to broadcast announcements (for example, a low battery message).
- **The Content Provider** shares the application's data with other applications. This data can be stored in a SQLite database.

Architecture

The Eclipse Platform uses plug-ins to provide all functionality within and on top of the runtime system, in contrast to some other applications, in which functionality is hard coded. The Eclipse Platform's runtime system is based on Equinox, an implementation of the OSGi core framework specification.

This plug-in mechanism is a lightweight software componentry framework.

In addition to allowing the Eclipse Platform to be extended using other programming languages such as C and Python, the plug-in framework allows the Eclipse Platform to work with typesetting

languages like Latex networking applications such as telnet and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with support for other version control.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal".

Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include a UML plug-in for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.

The Eclipse SDK includes the Eclipse Java development tools (JDT), offering an IDE with a built-in incremental Java compiler and a full model of the Java source files.

This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat file space allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements widgets through a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate graphical user interface layer called JFace, which simplifies the construction of applications based on SWT. Language packs developing by the "Babel project" provide translations into over a dozen natural languages

Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials.

Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, Windows XP or later.

Then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex file (compiled byte code files called Dalvik executable), resource files, etc.

Fast boot

Fast boot is a diagnostic protocol included with the SDK package used primarily to modify the flash file system via a USB connection from the host computer.

It requires that the device be started in a boot loader or Second Program Loader mode in which only the most basic hardware initialization is performed. After enabling the protocol on the device itself, it will accept a specific set of commands sent to it via USB using a command line. Some of most commonly used fast boot commands include:

- ☐ Flash - Rewrites a partition with a binary image stored on the host computer.
- ☐ Erase - Erases a specific partition.
- ☐ Reboot - Reboots the device into either the main operating system, the system recovery partition or back into its boot loader.

5.2 TESTING

5.2.1 UNIT TESTING

Unit testing is the testing of an individual unit or group of related units. The purpose of unit testing is to determine the correct working of the individual modules. It involves a precise definition of test cases, testing criteria and management of test cases.

TEST CASE	TEST CASE DESCRIPTION	EXPECTED RESULT	OBSERVED RESULT	RESULT PASS/FAIL
Enter into the application.	Check whether it enters or not.	It should enter.	It enters successfully.	Pass

5.2.2.1 Unit Testing

5.2.2 INTEGRATION TESTING

Integration testing is testing in which a group of components are combined to produce output. Also the interaction between software and hardware is tested in integration testing if software and hardware components have any relation.

TEST CASE	TEST CASE DESCRIPTION	EXPECTED RESULT	OBSERVED RESULT	RESULT PASS/FAIL
Enter into a module	Check whether it works correctly and generates output or not.	It should generate a correct output.	Output generated successfully.	Pass

5.2.3.1 Integration Testing

5.2.3 WHITE BOX TESTING

White-box testing is a testing technique that makes into account the internal mechanism of a system. It is also called structural testing and glass box testing. This testing is used for verification.

TEST CASE	TEST CASE DESCRIPTION	EXPECTED RESULT	OBSERVED RESULT	RESULT PASS/FAIL
All modules.	Check whether all modules work properly or not.	It should work correctly.	It works correctly.	Pass

5.2.4. White Box Testing

5.2.4 BLACK BOX TESTING

Black-box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It also called functional testing. Black-box testing is often used for validation.

TEST CASE	TEST CASE DESCRIPTION	EXPECTED RESULT	OBSERVED RESULT	RESULT PASS/FAIL
All modules execution and output.	Check whether all modules work properly and generate a correct output or not.	It should work correctly and should produce a correct output.	It works correctly and generates a correct output.	Pass

5.2.5.1 Black Box Testing

5.2.5 SYSTEM TESTING

Testing is vital role for the success of this system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

System testing is the stage of implementation that is aimed at assuring that the system works accurately and efficiently.

TEST CASE NO	TEST CASE	TEST CASE DESCRIPTION	EXPECTED RESULT	OBSERVED RESULT	RESULT PASS/FAIL
1.	Testing all the modules in the system.	It tests all the modules in the system whether its is executing in the system or not.	By testing all the modules its is accepted by the system and should execute the result.	It tests all the modules and the system accept all the modules and it produces an expected result	Pass

Table 5.2.1.1 System Testing

6. CONCLUSION

This application will help end users to be fit and maintain a healthy lifestyle & Also I have achieved this application to step forward with the better enhancements as per the user expectation.

7. FUTURE ENHANCEMENT

Mobile technology and machine learning developments will not only replace your fitness coach, gym trainer and yoga teacher but also it is set to kill the livelihood of dieticians.

Diet and nutrition mobile app development focuses on various aspects of healthy living by holistically utilizing resources for Diet & Weight Loss management, Food & Recipes, Vitamins & Supplements along with the facility of finding suitable healthy recipes.



8. APPENDIX

A.Screen Shots

SPLASH SCREEN

11:41



New update available, downloading...

8:38



Fitness Freak



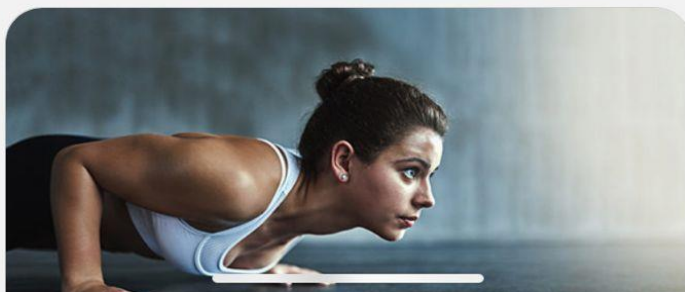
Bulk

6 Workouts



Lean

6 Workouts



SUB CATEGORY SCREEN

8:38

< Fitness Freak

Bulk



Chest (Monday)

Bulk



Shoulder (Tuesday)

Bulk



Lats (Wednesday)

Bulk



Bicep (Thursday)

Bulk



Tricep (Friday)

Bulk



Squats (Saturday)

Bulk



DIET

8:39

< Fitness Freak

Diet

**Triple Berry Smoothie**

Diet

**Triple Berry Smoothie**

Diet

**Triple Berry Smoothie**

Diet



EXERCISE VIDEO - DETAILS PAGE



Triple Berry Smoothie

DIET

🕒 10 minutes

[View Ingredients](#)

In a blender, combine all ingredients and blend until smooth. Then divide between 2 cups and top with blackberries, if desired.



INGREDIENTS DETAILS PAGE - DIET

8:39

[< Back](#) Ingredients for Triple Berry Smoo...

Banana
1



Frozen
Straberries
1/2 lbs



Greek Yogurt
1/2 liters



LOGIN DETAIL AND SUB MENU - NAV BAR

11:42

Hi, ponmani



Exercise



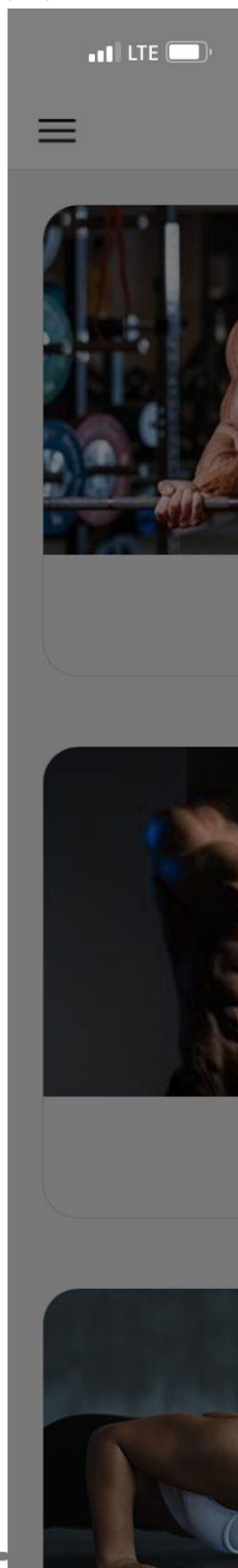
Diet



SEARCH



Sign Out



8:38



Fitness Freak



Fitness

7 Workouts



Cardio - Fat Loss

4 Workouts



Diet

3 Workouts

B.Sample Coding

LoginScreen.js

```
import { useNavigation } from '@react-navigation/core'
```

```
import React, { useEffect, useState } from 'react'
```

```
import { KeyboardAvoidingView, StyleSheet, ImageBackground, Text, TextInput, TouchableOpacity, View } from 'react-native'
```

```
import { auth } from '../firebase'
```

```
const LoginScreen = () => {
```

```
  const [email, setEmail] = useState("")
```

```
  const [password, setPassword] = useState("")
```

```
  const navigation = useNavigation()
```

```
  useEffect(() => {
```

```
    const unsubscribe = auth.onAuthStateChanged(user => {
```

```
      if (user) {
```

```
        navigation.replace("Fitness Freak")
```

```
      }
```

```
    })
```

```
    return unsubscribe
```

```
  }, [])
```

```
  const handleSignUp = () => {
```

```
    auth
```

```
      .createUserWithEmailAndPassword(email, password)
```

```
      .then(userCredentials => {
```

```
        const user = userCredentials.user;
```

```
        console.log('Registered with:', user.email);
```

```
      })
```

```
      .catch(error => alert(error.message))
```

```

}

const handleLogin = () => {
  auth
    .signInWithEmailAndPassword(email, password)
    .then(userCredentials => {
      const user = userCredentials.user;
      console.log('Logged in with:', user.email);
    })
    .catch(error => alert(error.message))
}

const image = require('../../assets/backimage.jpg')
return (
  <View style={styles.containerBackground}>
    <ImageBackground source={image} resizeMode="stretch" style={styles.image}>
      <KeyboardAvoidingView
        style={styles.container}
        behavior="padding">
        <View style={styles.inputContainer}>
          <TextInput
            placeholder="Email"
            value={email}
            onChangeText={text => setEmail(text)}
            style={styles.input}
          />
          <TextInput
            placeholder="Password"
            value={password}
            onChangeText={text => setPassword(text)}

```

```

        style={styles.input}

        secureTextEntry

    />

</View>

<View style={styles.buttonContainer}>

    <TouchableOpacity
        onPress={handleLogin}
        style={styles.button}
    >

        <Text style={styles.buttonText}>Login</Text>

    </TouchableOpacity>

    <TouchableOpacity
        onPress={handleSignUp}
        style={[styles.button, styles.buttonOutline]}
    >

        <Text style={styles.buttonOutlineText}>Register</Text>

    </TouchableOpacity>

</View>
</KeyboardAvoidingView>
</ImageBackground>
</View>

)
}

```

```
export default LoginScreen
```

```

const styles = StyleSheet.create({

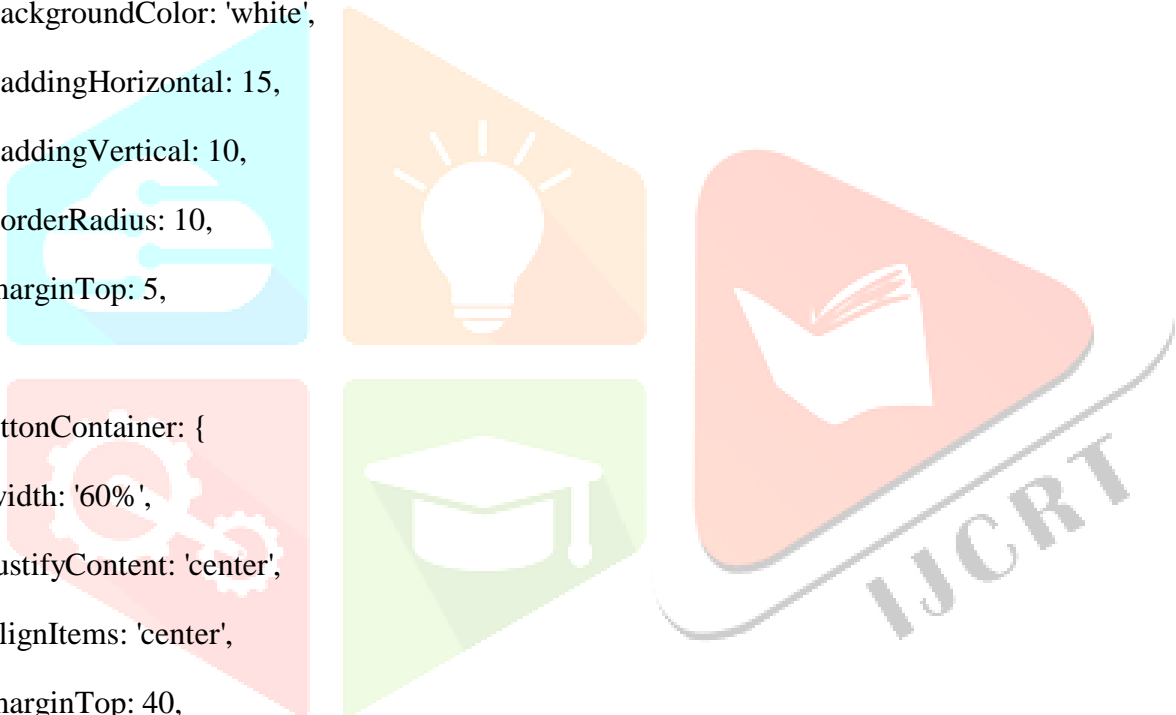
  containerBackground: {

    flex: 1

```



```
},  
  
container: {  
  flex: 1,  
  justifyContent: 'center',  
  alignItems: 'center',  
},  
  
inputContainer: {  
  width: '80%'  
},  
  
input: {  
  backgroundColor: 'white',  
  paddingHorizontal: 15,  
  paddingVertical: 10,  
  borderRadius: 10,  
  marginTop: 5,  
},  
  
buttonContainer: {  
  width: '60%',  
  justifyContent: 'center',  
  alignItems: 'center',  
  marginTop: 40,  
},  
  
button: {  
  backgroundColor: '#0782F9',  
  width: '100%',  
  padding: 15,  
  borderRadius: 10,  
  alignItems: 'center',  
},  
  
buttonOutline: {  
  backgroundColor: 'white',
```



```

marginTop: 5,

borderColor: '#0782F9',

borderWidth: 2,

},

buttonText: {

  color: 'white',

  fontWeight: '700',

  fontSize: 16,

},

buttonOutlineText: {

  color: '#0782F9',

  fontWeight: '700',

  fontSize: 16,

},

image: {

  flex: 1,

  justifyContent: "center"

},

})

```



HomeScreen.js

```

import React, { useEffect } from "react";

import { FlatList, Text, View, TouchableHighlight, Image } from "react-native";

import styles from "./styles";

import { recipes } from "../../data/dataArrays";

import MenuImage from "../../components/MenuImage/MenuImage";

import { getCategoryName } from "../../data/MockDataAPI";

export default function HomeScreen(props) {

  const { navigation } = props;

  useEffect(() => {

    navigation.setOptions({

```

```

headerLeft: () => (
  <MenuImage
    onPress={() => {
      navigation.openDrawer();
    }}
  />
),
headerRight: () => <View />,
});
}, []);

const onPressRecipe = (item) => {
  navigation.navigate("Recipe", { item });
};

const renderRecipes = ({ item }) => (
  <TouchableHighlight underlayColor="var(--spotim-color-scheme-g5);" onPress={() =>
onPressRecipe(item)}>
    <View style={styles.container}>
      <Image style={styles.photo} source={{ uri: item.photo_url }} />
      <Text style={styles.title}>{item.title}</Text>
      <Text style={styles.category}>{getCategoryName(item.categoryId)}</Text>
    </View>
  </TouchableHighlight>
);

return (
  <View>
    <FlatList vertical showsVerticalScrollIndicator={false} numColumns={2} data={recipes}
renderItem={renderRecipes} keyExtractor={(item) => `${item.recipeId}`} />
  </View>
);
}

```

CategoriesScreen.js

```
import React, { useEffect } from "react";
import { FlatList, Text, View, Image, TouchableHighlight } from "react-native";
import styles from "./styles";
import { categories } from "../../data/dataArrays";
import { getNumberOfRecipes } from "../../data/MockDataAPI";
import MenuImage from "../../components/MenuImage/MenuImage";
```

```
export default function CategoriesScreen(props) {
  const { navigation } = props;
```

```
  useEffect(() => {
    navigation.setOptions({
      headerTitleStyle: {
        fontWeight: "bold",
        textAlign: "center",
        alignSelf: "center",
        flex: 1,
      },
      headerLeft: () => (
        <MenuImage
          onPress={() => {
            navigation.openDrawer();
          }}
        />
      ),
      headerRight: () => <View />,
    });
  }, []);
```

```
  const onPressCategory = (item) => {
    const title = item.name;
    const category = item;
    navigation.navigate("RecipesList", { category, title });
```

```

};

const renderCategory = ({ item }) => (
  <TouchableHighlight underlayColor="var(--spotim-color-scheme-g5);" onPress={() =>
onPressCategory(item)}>
    <View style={styles.categoriesItemContainer}>
      <Image style={styles.categoriesPhoto} source={{ uri: item.photo_url }} />
      <Text style={styles.categoriesName}>{item.name}</Text>
      <Text style={styles.categoriesInfo}>{getNumberOfRecipes(item.id)} Workouts</Text>
    </View>
  </TouchableHighlight>
);

return (
  <View>
    <FlatList data={categories} renderItem={renderCategory} keyExtractor={(item) => `${item.id}`} />
  </View>
);
}

```

DetailsScreen.js

```

import React, { useEffect } from "react";
import { FlatList, Text, View, Image, TouchableHighlight } from "react-native";
import styles from "./styles";
import { getIngredientName, getAllIngredients } from "../../data/MockDataAPI";

export default function IngredientsDetailsScreen(props) {
  const { navigation, route } = props;

  const item = route.params?.ingredients;
  const ingredientsArray = getAllIngredients(item);

  useEffect(() => {
    navigation.setOptions({

```

```

    title: route.params?.title,

    headerTitleStyle: {
      fontSize: 16,
    },
  });
}, []);

const onPressIngredient = (item) => {
  let name = getIngredientName(item.ingredientId);
  let ingredient = item.ingredientId;
  navigation.navigate("Ingredient", { ingredient, name });
};

const renderIngredient = ({ item }) => (
  <TouchableHighlight underlayColor="var(--spotim-color-scheme-g5);" onPress={() =>
onPressIngredient(item[0])}>
    <View style={styles.container}>
      <Image style={styles.photo} source={{ uri: item[0].photo_url }} />
      <Text style={styles.title}>{item[0].name}</Text>
      <Text style={{ color: "grey" }}>{item[1]}</Text>
    </View>
  </TouchableHighlight>
);

return (
  <View>
    <FlatList vertical showsVerticalScrollIndicator={false} numColumns={3} data={ingredientsArray}
renderItem={renderIngredient} keyExtractor={(item) => `${item.recipeId}`} />
  </View>
);
}

```

Search Screen.js

```
import React, { useEffect, useLayoutEffect, useState } from "react";
import { FlatList, Text, View, Image, TouchableHighlight, Pressable } from "react-native";
import styles from "./styles";
import MenuImage from "../../components/MenuImage/MenuImage";
import { getCategoryName, getRecipesByRecipeName, getRecipesByCategoryName,
getRecipesByIngredientName } from "../../data/MockDataAPI";
import { TextInput } from "react-native-gesture-handler";

export default function SearchScreen(props) {
  const { navigation } = props;

  const [value, setValue] = useState("");
  const [data, setData] = useState([]);

  useLayoutEffect(() => {
    navigation.setOptions({
      headerLeft: () => (
        <MenuImage
          onPress={() => {
            navigation.openDrawer();
          }}
        />
      ),
      headerTitle: () => (
        <View style={styles.searchContainer}>
          <Image style={styles.searchIcon} source={require("../../assets/icons/search.png")} />
          <TextInput
            style={styles.searchInput}
            onChangeText={handleSearch}
            value={value}
          />
          <Pressable onPress={() => handleSearch("")}>
            <Image style={styles.searchIcon} source={require("../../assets/icons/close.png")} />
          </Pressable>
        </View>
      )
    });
  });
}
```



```

</View>

),

headerRight: () => <View />,

});

}, [value]);

useEffect(() => {}, [value]);

const handleSearch = (text) => {
  setValue(text);
  var recipeArray1 = getRecipesByRecipeName(text);
  var recipeArray2 = getRecipesByCategoryName(text);
  var recipeArray3 = getRecipesByIngredientName(text);
  var aux = recipeArray1.concat(recipeArray2);
  var recipeArray = [...new Set(aux)];
  if (text == "") {
    setData([]);
  } else {
    setData(recipeArray);
  }
};

const onPressRecipe = (item) => {
  navigation.navigate("Recipe", { item });
};

const renderRecipes = ({ item }) => (
  <TouchableHighlight underlayColor="var(--spotim-color-scheme-g5);" onPress={() =>
onPressRecipe(item)}>
    <View style={styles.container}>
      <Image style={styles.photo} source={{ uri: item.photo_url }} />
      <Text style={styles.title}>{item.title}</Text>
      <Text style={styles.category}>{getCategoryName(item.categoryId)}</Text>
    </View>
  </TouchableHighlight>
)

```

```
</TouchableHighlight>

);

return (
  <View>
    <FlatList vertical showsVerticalScrollIndicator={false} numColumns={2} data={data}
    renderItem={renderRecipes} keyExtractor={(item) => `${item.recipeId}`} />
  </View>
);
}
```

