

TRADITIONAL HEALTHY FOOD REMINDER APPLICATION

A PROJECT REPORT

Submitted by

SATHISH P [513319104036]

YOKESH M [513319104049]

KALYANA SUNDARAM R [513319104014]

RISHYASRUNGAR R A [513319104033]

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING,

ARNI 632 326

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2022

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**TRADITIONAL HEALTHY FOOD REMINDER APPLICATION**” is the bonafide work of “**SATHISH P (513319104036), KALYANA SUNDARAM R (513319104014), YOKESH M (513319104049), RISHYASRUNGAR R A (51331819104033)**” who carried out the project work under my supervision.

SIGNATURE

Mr. G. MANI M.E.,
HEAD OF THE DEPARTMENT,
ASSISTANT PROFESSOR
Department of Computer Science
and Engineering,
University College of Engineering,
Arni-632 326.

SIGNATURE

Mr. S. CHANDRASEKAR M.E.,
SUPERVISOR,
TEACHING FELLOW
Department of Computer Science
and Engineering,
University College of Engineering,
Arni-632 326.

Submitted for Project viva voice Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

In many parts of the world the relationship between food and health is predominately defined by a nutritional and medical discourse today. This discourse focuses on food intake as a core determinant of individual bodily and mental health, prevention of under or overweight, and of future diseases. Sociologists and other social scientists, however, have a broader understanding of the relationship between food and health and emphasize how cultural meanings and beliefs as well as social structures and institutions such as education, media, law, politics, and economy shape food practices. The aim of sociological research, reviewed here, is to understand the conditions of possibility for the emergence and prominence of a medical nutrition discourse on food and health and what it tells us about contemporary society.

Healthy food choices are happy food choices: Evidence from a real life sample using smartphone based assessments. Research suggests that “healthy” food choices such as eating fruits and vegetables have not only physical but also mental health benefits and might be a long-term investment in future well-being. This view contrasts with the belief that high-caloric foods taste better, make us happy, and alleviate a negative mood. It protects you against many chronic noncommunicable diseases, such as heart disease, diabetes and cancer. Eating a variety of foods and consuming less salt, sugars and saturated and industrially-produced trans-fats, are essential for healthy diet.

Healthy food refers to food that contains the right amount of nutrients to keep our body fit. We need healthy food to keep ourselves fit. Furthermore, healthy food is also very delicious as opposed to popular thinking. Nowadays, kids need to eat healthy food more than ever. We must encourage good eating habits so that our future generations will be healthy and fit. Most importantly, the harmful effects of junk food and the positive impact of healthy food must be stressed upon. People should teach kids from an early age about the same.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iii
	LIST OF FIGURES	vi
1	INTRODUCTION	1
	1.1 Scope of the project	2
	1.2 Objective of the project	3
2	LITERATURE SURVEY	5
	2.1 Universe of Ionic Framework	5
3	SYSTEM DESIGN	7
	3.1 System Architecture	7
	3.2 Hefo Page View	8
	3.2.1 Breakfast Page View	9
	3.2.2 Juice Page View	10
	3.2.3 Lunch Page View	11
	3.2.4 Snacks Page View	12
	3.2.5 Dinner Page view	13
	3.3 Navigation Tool View	14
	3.3.1 Home Page View	15
	3.3.2 Recipes Page View	16
	3.3.3 Health Benefits Page View	17
	3.3.4 Setting page View	18
	3.3.5 About Page View	19
4	SYSTEM DEVELOPMENT	20
	4.1 Android Studio	20
	4.1.1 Features of Android Studio	20

	4.2 NodeJS	21
	4.2.1 Routing Module Page	22
	4.2.2 HTML Page	24
	4.2.3 SCSS Page	25
	4.2.4 SPEC TS Page	25
	4.2.5 Module TS Page	26
	4.2.6 TS page	26
	4.3 GitHub	27
	4.3.1 Pull Operation	27
	4.3.2 Push Operation	27
	4.3.3 Commit Operation	27
	4.4 TortoiseGit	28
	4.4.1 Features of TortoiseGit	28
5	APK TESTING	31
	5.1 Black Box Testing	32
	5.2 White Box Testing	34
	5.3 Testing Results	37
6	CONCLUSION AND FUTURE WORK	40
	APPENDIX	40
	REFERENCES	46

LIST OF FIGURES

Name of figures

3.1	System Architecture	7
3.2	Hefo Page View	8
3.2.1	Breakfast Page View	9
3.2.2	Juice Page View	10
3.2.3	Lunch Page View	11
3.2.4	Snacks Page View	12
3.2.5	Dinner Page View	13
3.3	Navigation Tool View	14
3.3.1	Home Page View	15
3.3.2	Recipes Page View	16
3.3.3	Health Benefits Page View	17
3.3.4	Setting Page View	18
3.3.5	About Page View	19
5.1	Black Box Testing	33
5.2	White Box Testing	37
5.3.1	Testing Results Passed	38
5.3.2	Test Results Performance	39
5.3.3	Test Results Performance Graph	39

CHAPTER 1

INTRODUCTION

Ionic is an advanced HTML5 hybrid mobile application framework. Using Apache Cordova, it can be easily run on mobiles. A big advantage of using Ionic is that the code base remains the same for all mobile platforms and the UI/UX looks like a native mobile platform.

Developing mobile applications is easy if we are only targeting one platform, but that's not the case with many organisations. Mobile applications for all platforms are difficult to maintain and create, as we need a different code base for each platform (Android, iPhone, Windows), and the UI/UX should be the same for all. Each platform comes with its own set of tools and languages like Java for Android, XCode for Apple and Windows SDK for Windows phones. And each has its own IDE to work on. Instead of building the apps for all platforms separately, many frameworks or tools allow an application to be developed once, to be used on any platform. Usually, this tool or framework uses HTML and CSS for the UI/UX part.

Ionic is one such framework that uses HTML5 for the UI/UX part. It is built on HTML, JS and CSS, which contain UI elements in different forms to suit specific platforms. Being an HTML framework, we can run the Ionic application in a browser very easily, but to run it on a mobile, Ionic uses Cordova, an Apache product, which allows it to access the native features of mobiles like Contacts, Camera, etc. So, Ionic is a mobile application development framework used to build hybrid apps or mobile applications. It uses HTML5 for developing the mobile applications. A hybrid app is a website running in a browser shell in an app that has access to the native platform layer. Compared to pure native apps, hybrid apps shine in terms of more platform support, ease and speed of development, and easy access to third party code.

As per its website, Ionic is like ‘Bootstrap for Native’. It is a UI framework that handles the user interface and its interaction. Ionic comes with native style UI elements and layout. A big advantage of using Ionic is that the code base remains one for all mobile platforms and the UI/UX looks like a native mobile platform. Ionic also uses AngularJS. It provides many directives for the Angular App.

1.1 SCOPE OF PROJECT

Health is wealth! Eating healthy food and following a healthy lifestyle can keep us healthy and happy. Let us understand what healthy food is! Meaning of Healthy Food: Healthy food simply means nutritious food. These are the foods that are considered good for health. Healthy foods assist a person in enhancing the physical and mental well-being. These foods are full of nourishment and enhance growth. Some of the significant examples of healthy food involve natural food, fiber-rich food, vitamin-rich food, protein-rich food, etc. Healthy food keeps us away from diseases. We feel relaxed, light and stress-free when we eat nourishing food because we know we are feeding something healthy and good to our bodies

- **Promotes overall health** – We get a strong and healthy body with a healthy mind as well by eating healthy food. Healthy food keeps fit and active.
- **Active brain** – Healthy food is full of various nutrients. These nutrients provide us energy and alertness. Hence, we stay active.
- **It protects you from various diseases, including chronic diseases** – We can safeguard us from various dangerous diseases like diabetes, high blood pressure, high cholesterol, etc. by eating healthy food.
- **Combat obesity** – Healthy food also saves us from obesity as it helps in managing unnecessary weight gain.

- **Strengthens our immune system and digestion** – Healthy food strengthens our immune system, and digestion. For example, fiber rich food keeps us digestion smooth and vitamin C improves our immunity.
- **It helps you build an attractive body**– Healthy food also gives us a fit and fine body, glowing skin and overall, an attractive body.

Eat healthy food and live life to its fullest. So our parents and doctors force us to eat healthy food. To achieve good health one should eat healthy. Healthy eating include balanced diet and balanced proportion of carbohydrates, proteins, water and vitamins. Eat healthy and on time so that it can affect you even more. Besides you eat healthy, exercise also plays a very good role in the formation of your health. Have you ever wondered why your parents and doctors stop you to eat too much junk food? Overweight is a very serious problem nowadays especially with children. Junk food is one of the reasons for obese and overweight.

So, I suggest everyone to eat healthy, do exercise and be healthy as health is wealth.

1.2 OBJECTIVE OF PROJECT

- **Fruit and Vegetables**

Set a goal to fill half your plate with fruit and vegetables at every meal. Fruit and vegetables are naturally low in saturated and trans fat, and rich in dietary fibre, vitamins and minerals. Or simply set a goal to use My Healthy Plate for all meals.

- **Stick to One Serving**

For those of us who love having seconds, eating one serving will help us keep our calorie intake in check. Challenge yourself to stick to one serving and also standard portion sizes.

- **Stick to One Serving**

For those of us who love having seconds, eating one serving will help us keep our calorie intake in check. Challenge yourself to stick to one serving and also standard portion sizes.

- **Choose Water**

Set a goal to drink water instead of sugar-sweetened drinks. To make this more measurable, write down how often you will make this choice e.g. 5 times a week.

- **Eat Slowly and Mindfully**

Takes about 20 minutes for your brain to send out signals that you are full. Eat slowly. Take the extra time to pay attention to what we are eating and how much. To make this more measurable, write down how often you will make an effort to take at least a half hour to finish your meal.

- Improve nutrition and health outcomes of vulnerable segments (children and women) of the populations, through availability of foods that would increase intake of vegetables and fruits, decrease caloric intake and increase micronutrient intake.
- Develop food production systems based on agricultural diversification, conservation of water, and efficient use of land;
- Adapt international standards of food safety and quality for a healthy, market-oriented food supply chain in all countries;
- Increase the rate of technology adoption by small farmers experiencing common agricultural challenges;
- Improved nutrition, particularly among children and women, through changes in school lunch programmes and nutrition education
- Change in consumer behaviour towards increased consumption of a diversified diet of vegetable and fruits, with implications for obesity and diet related diseases

CHAPTER 2

LITERATURE SURVEY

2.1 Universe of Ionic Framework

Cross-platform

Build and deploy apps that work across multiple platforms, such as native iOS, Android, desktop, and the web as a Progressive Web App - all with one code base. Write once, run anywhere.

Web Standards-based

Ionic is built on top of reliable, standardized web technologies: HTML, CSS, and JavaScript using modern Web APIs such as Custom Elements and Shadow DOM. Because of this, Ionic components have a stable API and aren't at the whim of a single platform vendor.

Beautiful Design

Clean, simple, and functional. Ionic is designed to work and display beautifully out-of-the-box across all platforms. Start with pre-designed components, typography, interactive paradigms, and a gorgeous (yet extensible) base theme.

Simplicity

Ionic is built with simplicity in mind so that creating apps is enjoyable, easy to learn, and accessible to just about anyone with web development skills.

Framework Compatibility

While past releases of Ionic were tightly coupled to Angular, version 4. x of the framework was re-engineered to work as a standalone web component library, with integrations for the latest JavaScript frameworks, like Angular. Ionic can be used in most frontend frameworks with success, including React and Vue, though some frameworks need a shim for full Web Component support.

JavaScript

One of the main goals of moving Ionic to web components was to remove any hard requirement on a single framework to host the components. This made it possible for the core components to work standalone on a web page with just a script tag. While working with frameworks can be great for larger teams and larger apps, it is now possible to use Ionic as a standalone library on a single page even in a context like WordPress.

Angular

Angular has always been at the center of what makes Ionic great. While the core components have been written to work as a standalone Web Component library, the `@ionic/angular` package makes integration with the Angular ecosystem a breeze. `@ionic/angular` includes all the functionality that Angular developers would expect coming from Ionic 2/3, and integrates with core Angular libraries, like the Angular router.

React

Ionic now has official support for the popular React library. Ionic React let's React developers use their existing web skills to build apps that target iOS, Android, the web, and the desktop. With `@ionic/react`, you can use all the core Ionic components, but in a way that feels like using native React components. React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Vue

Ionic now has official support for the popular Vue 3 library. Ionic Vue lets Vue developers use their existing web skills to build apps that target iOS, Android, the web, and the desktop. With `@ionic/vue`, you can use all the core Ionic components, but in a way that feels like using native Vue component.

CHAPTER 3

SYSTEM ARCHITECTURE

3.1 SYSTEM ARCHITECTURE

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

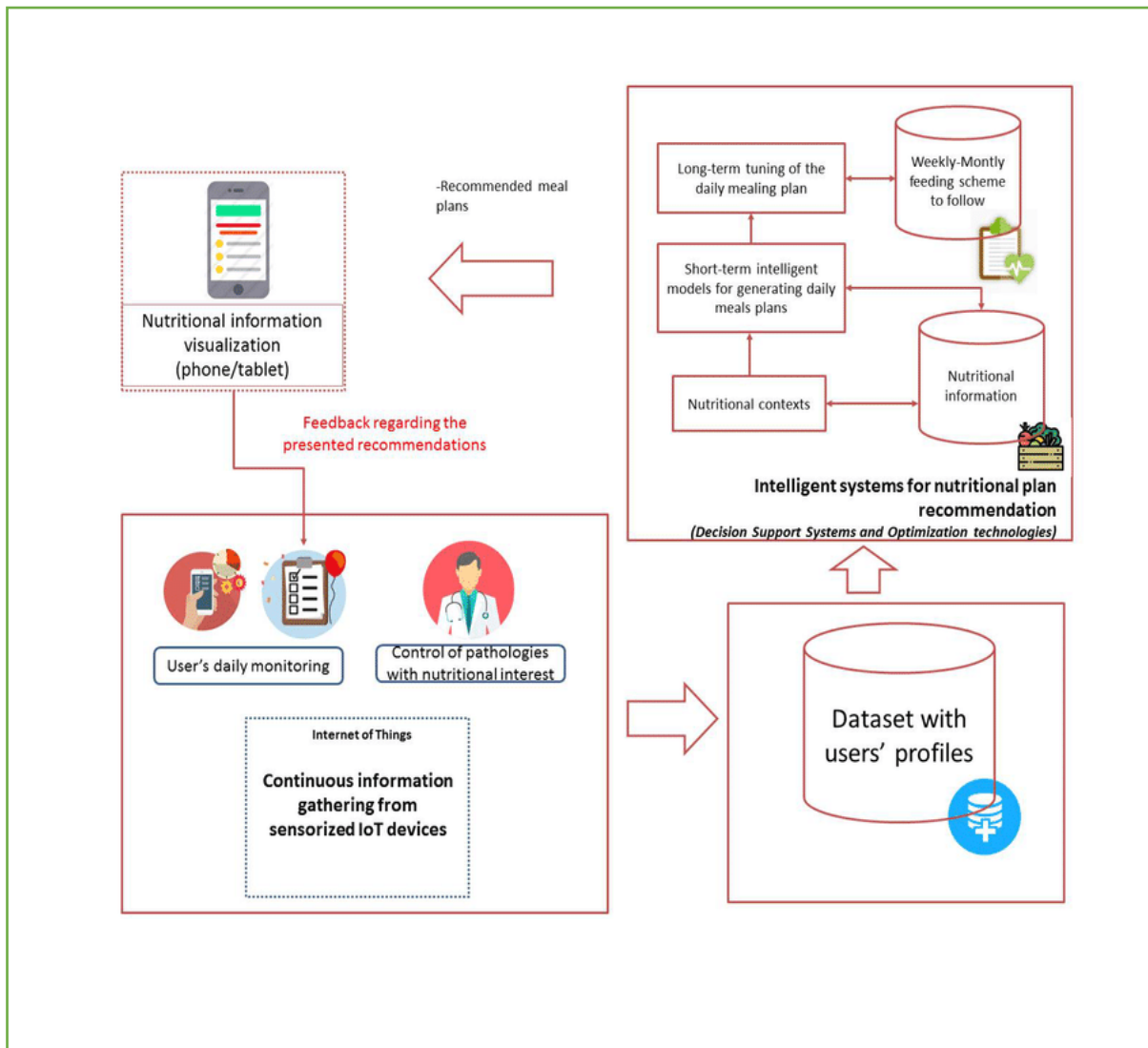


Figure 3.1 System Architecture

SYSTEM DESIGN

3.2 Hefo Page View

In this page you can see the daily routines your life like Breakfast Page, Juice Page, Lunch Page, Snacks Page, Dinner Page.

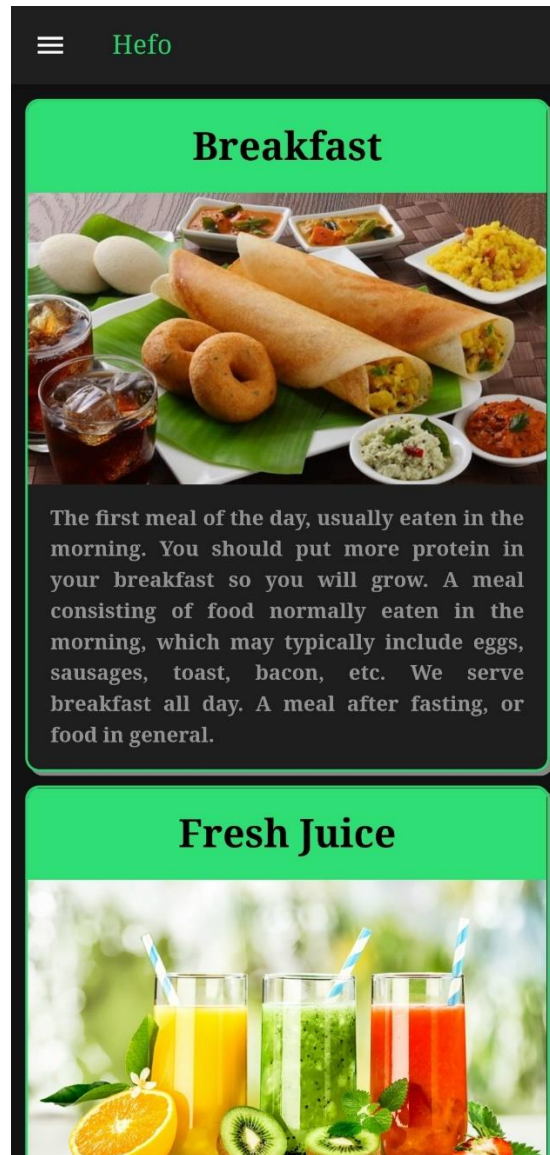


Figure 3.2 Hefo Page View

3.2.1 Breakfast Page View

While clicking on Breakfast Pages you can able to see the all breakfast recipes, you can click and view that recipes health benefits, Ingredients, Directions and Video link.



Figure 3.2.1 Breakfast Page View

3.2.2 Juice Page View

While clicking on Juice Pages you can able to see the all juice recipes, you can click and view that recipes health benefits, Ingredients, Directions and Video link.

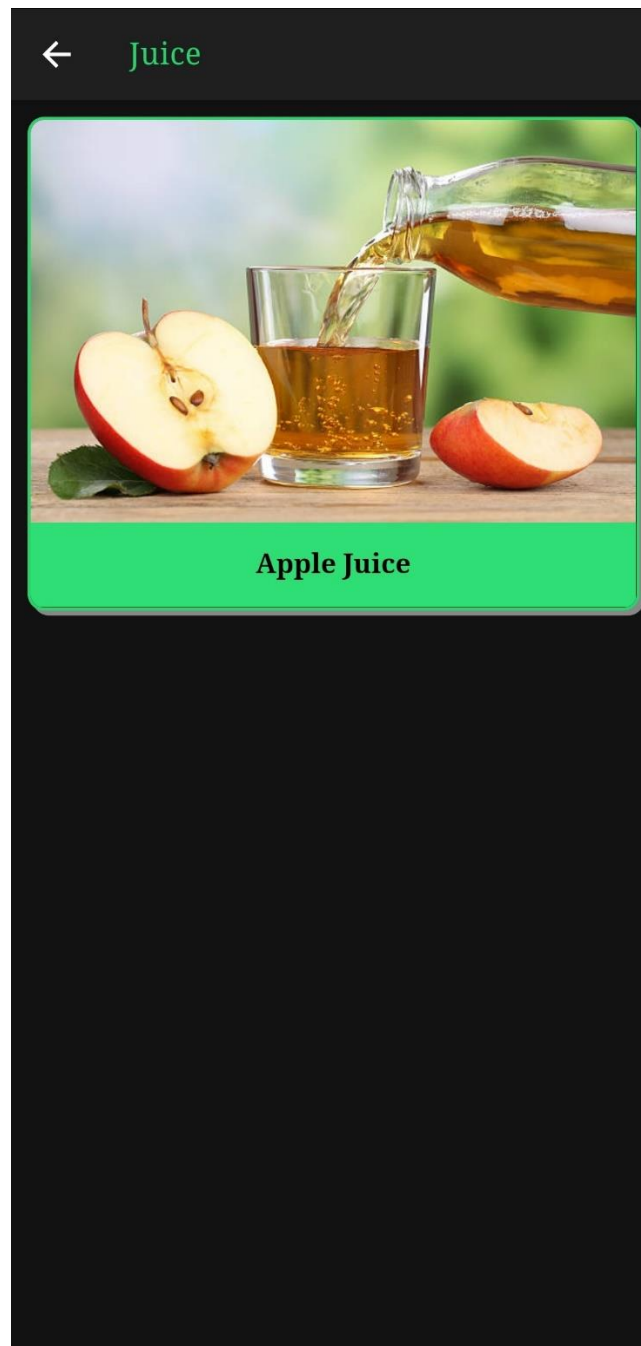


Figure 3.2.2 Juice Page View

3.2.3 Lunch Page View

While clicking on Lunch Pages you can able to see the all breakfast recipes, you can click and view that recipes health benefits, Ingredients, Directions and Video link.

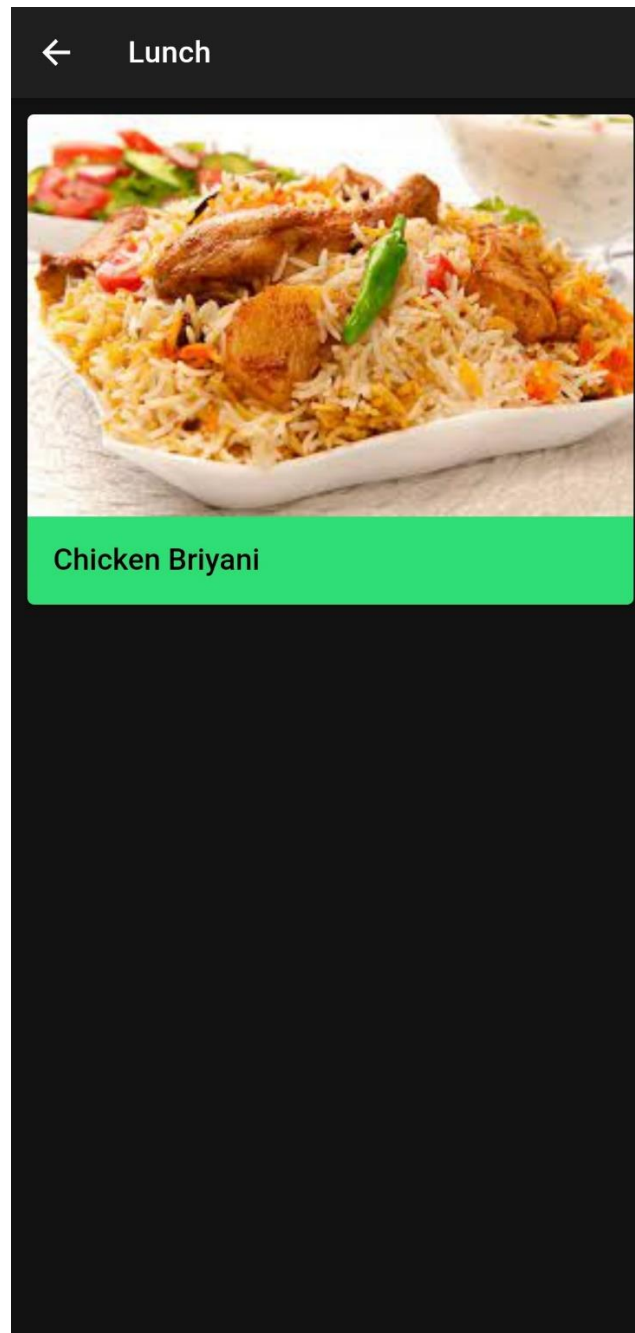


Figure 3.2.3 Lunch Page View

3.2.4 Snacks Page View

While clicking on Snacks Pages you can able to see the all Snacks recipes, you can click and view that recipes health benefits, Ingredients, Directions and Video link.

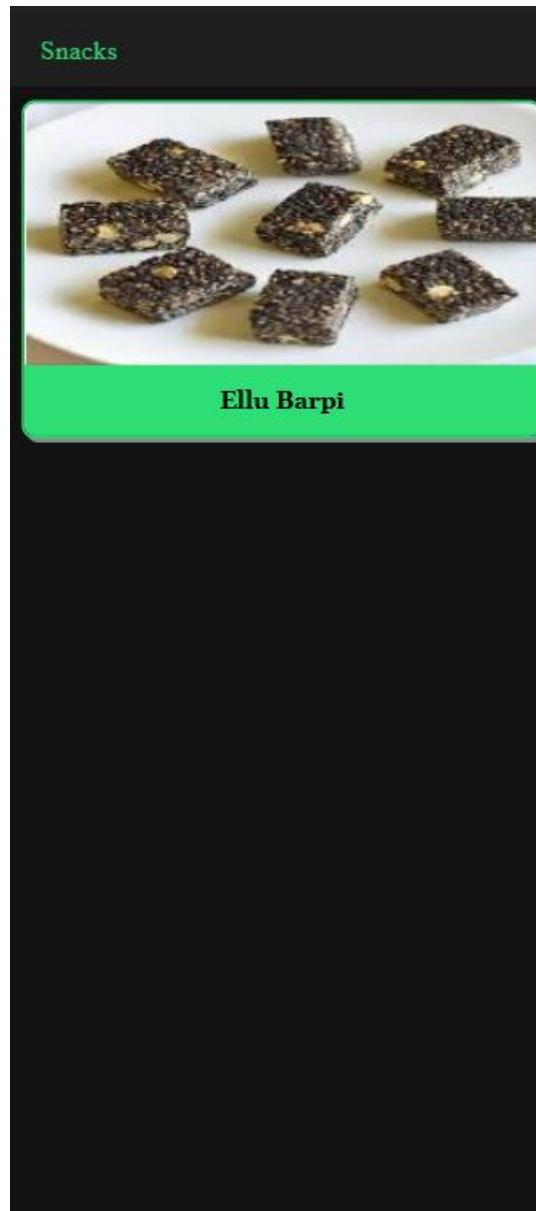


Figure 3.2.4 Snacks Page View

3.2.5 Dinner Page View

While clicking on Dinner Pages you can able to see the all breakfast recipes, you can click and view that recipes health benefits, Ingredients, Directions and Video link.

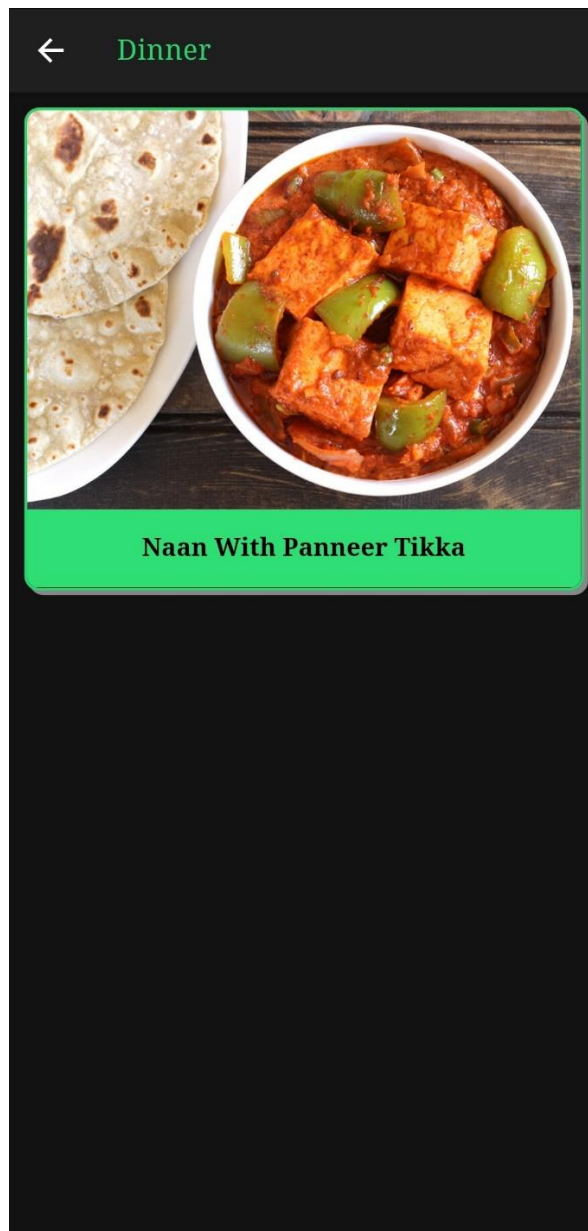


Figure 3.2.5 Dinner Page View

3.3 Navigation Tool View

Navigation refers to the act of opening and moving through application menus, like the Home Page, Recipes Page, Health Benefits Page, Setting Page.

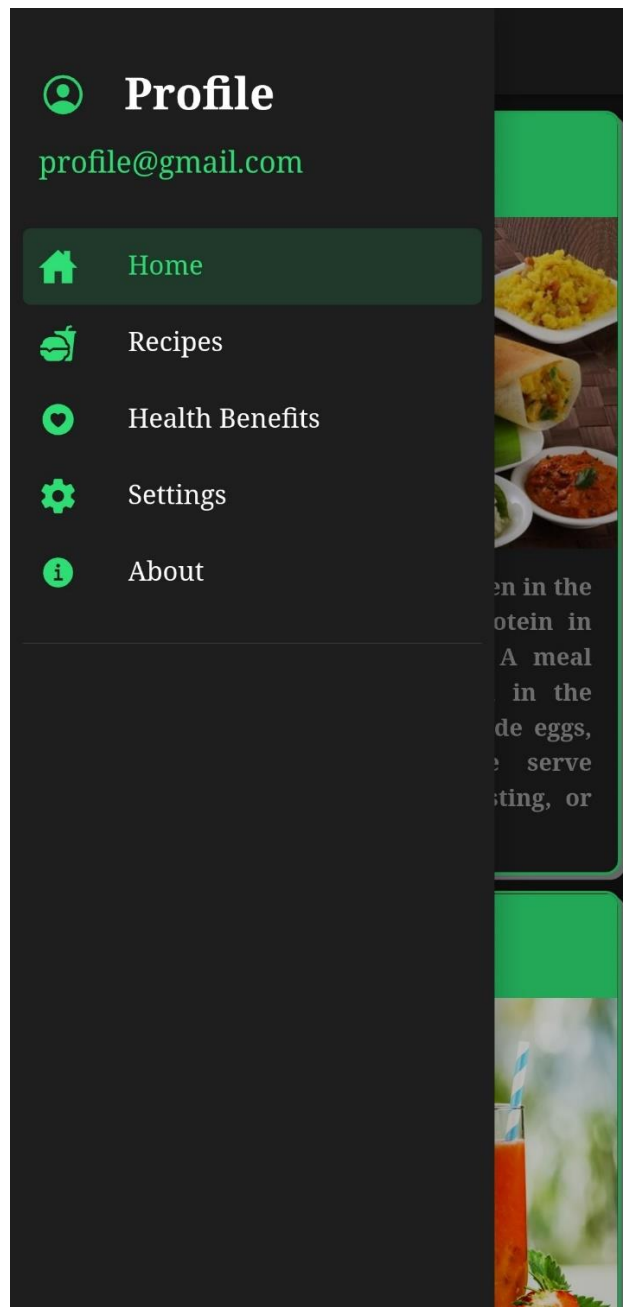


Figure 3.3 Navigation Tool View

3.3.1 Home Page View

Home page is also same as Hefo page you can see the daily routines your life like Breakfast Page, Juice Page, Lunch Page, Snacks Page, Dinner Page.

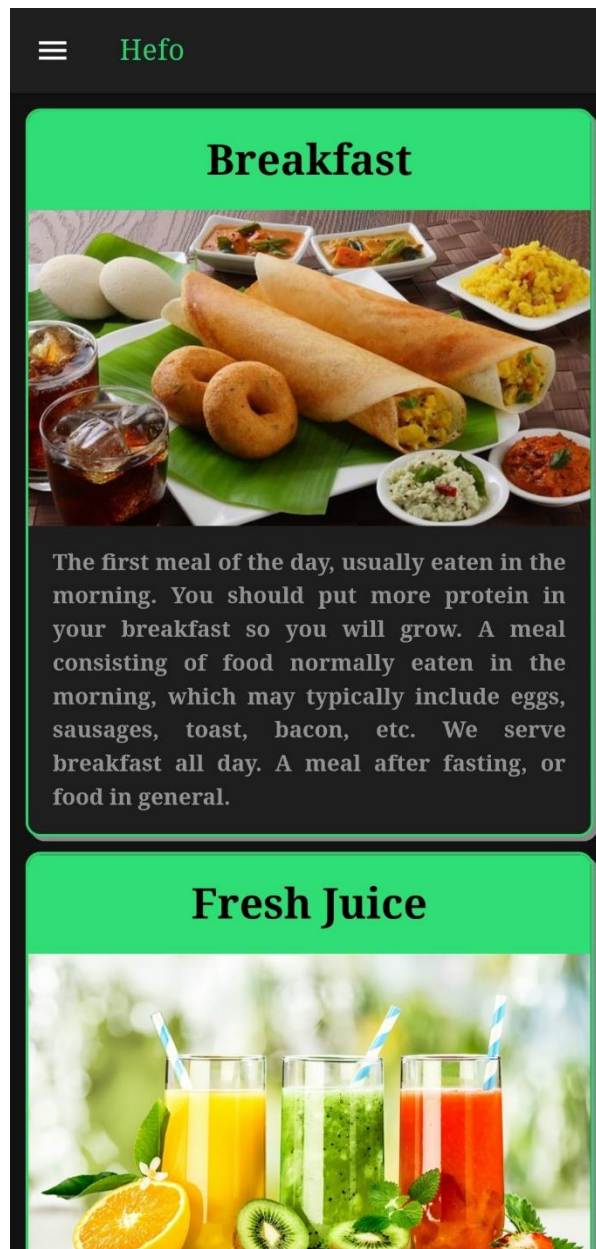


Figure 3.3.1 Home Page View

3.3.2 Recipes Page View

In this recipes page it contains all recipes of the application. It helps the user to identify the recipes easily.

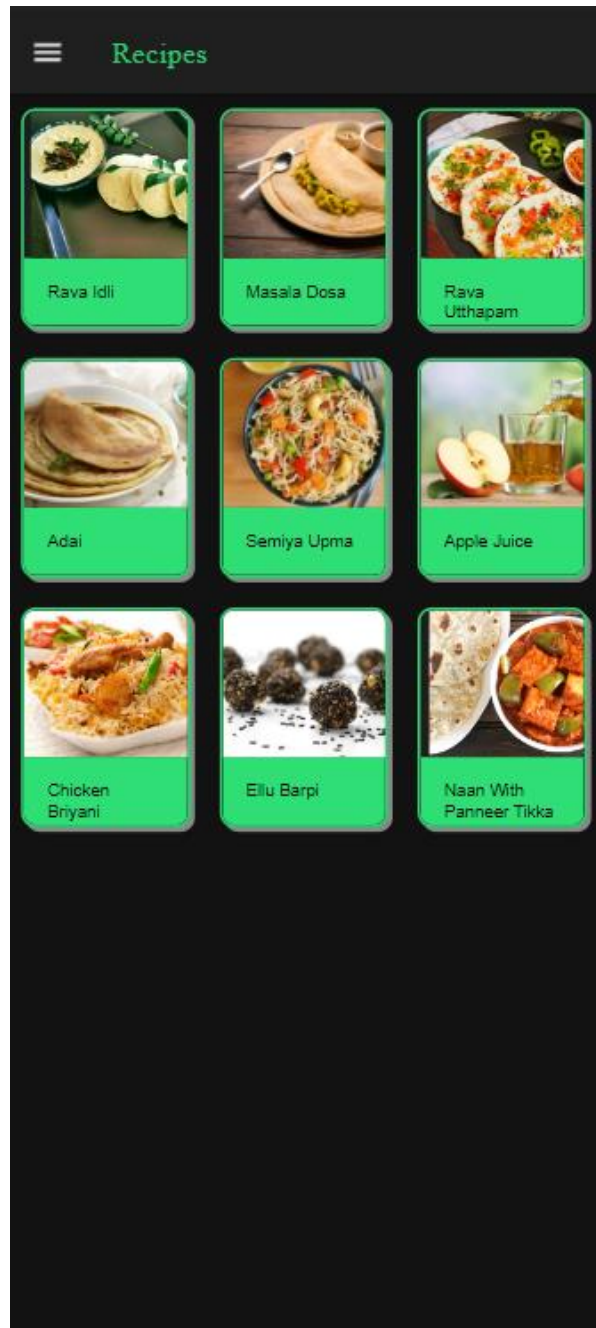


Figure 3.3.2 Recipes Page View

3.3.3 Health Benefits Page View

Health Benefits Page helps the user to get the knowledge about the health benefits of the food.

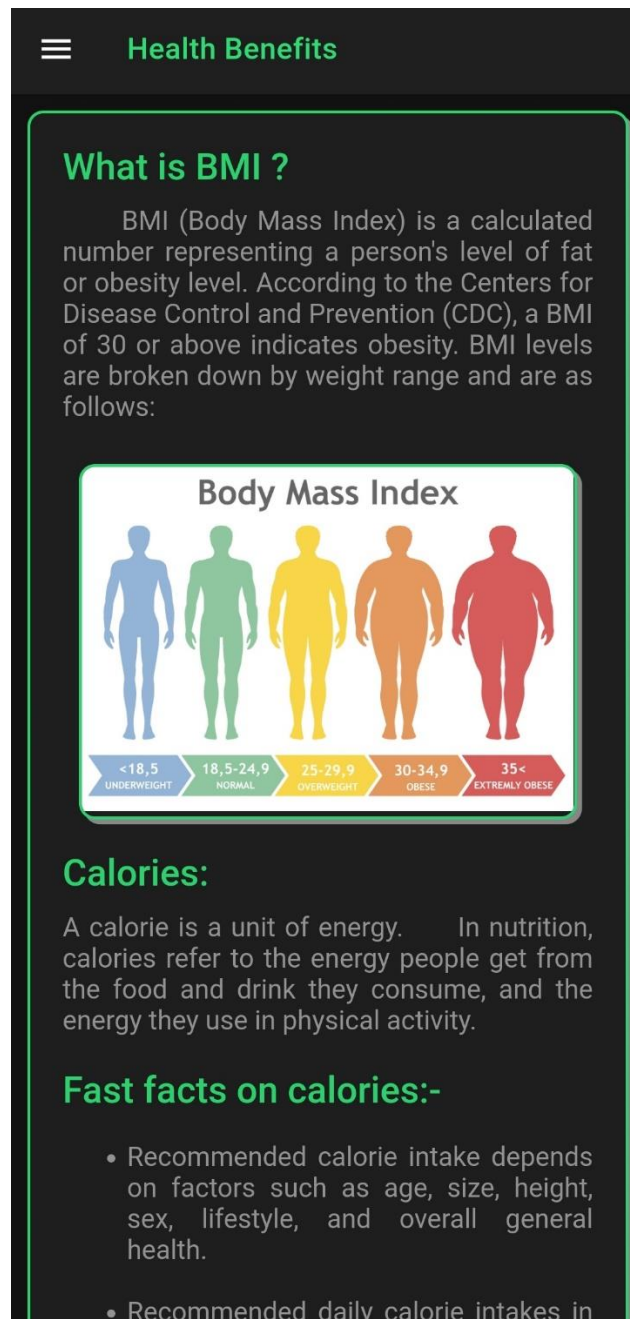


Figure 3.3.3 Health Benefits Page View

3.3.4 Setting page View

The Setting page gives the developer to access the backend functionality of the app.

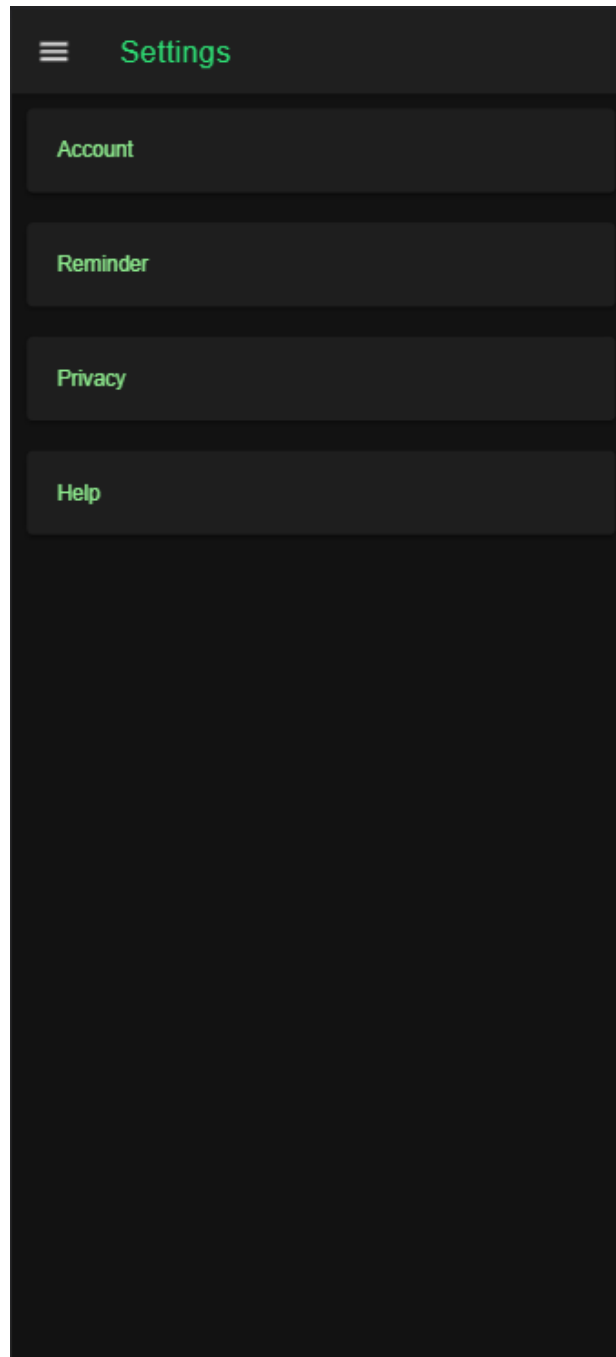


Figure 3.3.4 Setting Page View

3.3.5 About Page View

In the about page it gives the essential source of information for all who want to know more about our Application.

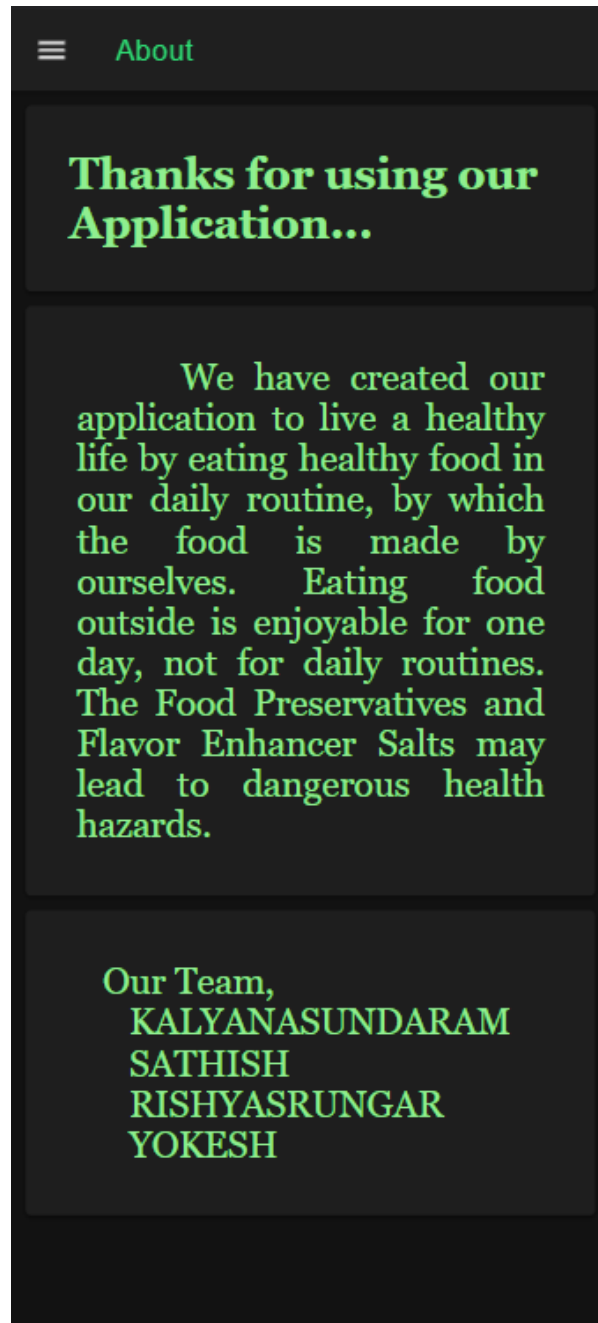


Figure 3.3.5 About Page View

CHAPTER 4

SYSTEM DEVELOPMENT

4.1 ANDROID STUDIO

Android Studio is a new and fully integrated development environment, which has been recently launched by Google for the Android operating system. It has been designed to provide new tools for app development and to provide an alternative to Eclipse, currently the most widely used IDE.

When you begin a new project in Android studio, the project's structure will appear with almost all the files held within the SDK directory, this switch to a Gradle based management system offers an even greater flexibility to the build process.

4.1.1 Features of Android Studio

Android Studio allows you to see any visual changes you make to your app in real-time, and you can also see how it will look on a number of different Android devices, each with different configurations and resolutions, simultaneously.

Another feature in Android Studio are the new tools for the packing and labelling of code. These let you keep on top of your project when dealing with large amounts of code. The programme also uses a drag & drop system to move the components throughout the user interface.

In addition, this new environment comes with Google Cloud Messaging, a feature which lets you send data from the server to Android devices through the cloud, a great way to send Push notifications to your apps.

The programme will also help you to localize your apps, giving you a visual way to keep programming while controlling the flow of the application.

4.2 NodeJS

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project!

Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.

A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back.

This allows Node.js to handle thousands of concurrent connections with a single server without introducing the burden of managing thread concurrency, which could be a significant source of bugs.

Node.js has a unique advantage because millions of frontend developers that write JavaScript for the browser are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language.

In Node.js the new ECMAScript standards can be used without problems, as you don't have to wait for all your users to update their browsers - you are in charge of deciding which ECMAScript version to use by changing the Node.js version, and you can also enable specific experimental features by running Node.js with flags.

4.2.1 Routing Module Page

Implementation of routing in Node.js: There are two ways to implement routing in node.js which are listed below:

- By Using Framework
- Without using Framework

Using Framework

Node has many frameworks to help you to get your server up and running. The most popular is Express.js.

Routing with Express in Node: Express.js has an “app” object corresponding to HTTP. We define the routes by using the methods of this “app” object. This app object specifies a callback function, which is called when a request is received. We have different methods in app object for a different type of request.

For GET request use app.get() method:

```
var express = require('express')
var app = express()
app.get('/', function(req, res)
{
    res.send('Hello Sir')
})
```

For POST request use app.post() method:

```
var express = require('express')
var app = express()
app.post('/', function(req, res)
{
    res.send('Hello Sir')
```

```
}  
)
```

For handling all HTTP methods (i.e. GET, POST, PUT, DELETE etc.) use `app.all()` method:

```
var express = require('express')  
var app = express()  
app.all('/', function(req, res)  
{  
  console.log('Hello Sir')  
  next() // Pass the control to the next handler  
})
```

The `next()` is used to hand off the control to the next callback. Sometimes we use `app.use()` to specify the middleware function as the callback.

So, to perform routing with the Express.js you have only to load the express and then use the app object to handle the callbacks according to the requirement.

Routing without Framework

Using the frameworks is good to save time, but sometimes this may not suit the situation. So, a developer may need to build up their own server without other dependencies.

Now create a file with any name using .js extension and follow the steps to perform routing from scratch:

Here we will use the built in module of node.js i.e. http. So, First load http:

```
var http = require('http');
```

Now create server by adding the following lines of code:

```
http.createServer(function (req, res)  
{  
  res.write('Hello World!'); // Write a response  
  res.end(); // End the response
```

```
}).listen(3000, function() {  
    console.log("server start at port 3000"); // The server object listens on port  
    3000  
});
```

Now add the following lines of code in above function to perform routing:

```
var url = req.url;  
if(url === '/about') {  
    res.write(' Welcome to about us page');  
    res.end();  
} else if(url === '/contact') {  
    res.write(' Welcome to contact us page');  
    res.end();  
} else {  
    res.write('Hello World!');  
    res.end();  
}
```

4.2.2 HTML PAGE

Extend the HTML vocabulary of your applications With special Angular syntax in your templates. For example, Angular helps you get and set DOM (Document Object Model) values dynamically with features such as built-in template functions, variables, event listening, and data binding.

Almost all HTML syntax is valid template syntax. However, because an Angular template is part of an overall webpage, and not the entire page, you don't need to include elements such as `<html>`, `<body>`, or `<base>`, and can focus exclusively on the part of the page you are developing.

To eliminate the risk of script injection attacks, Angular does not support the `<script>` element in templates. Angular ignores the `<script>` tag and outputs a warning to the browser console. For more information, see the Security page. Use binding to coordinate values in your application.

4.2.3 SCSS PAGE

Ionic has nine default colors that can be used to change the color of many components. Each color is actually a collection of multiple properties, including a shade and tint, used throughout Ionic.

A color can be applied to an Ionic component in order to change the default colors using the color attribute. Notice in the buttons below that the text and background changes based on the color set. When there is no color set on the button it uses the primary color by default.

```
<ion-button>Default</ion-button>
<ion-button color="primary">Primary</ion-button>
<ion-button color="secondary">Secondary</ion-button>
<ion-button color="tertiary">Tertiary</ion-button>
<ion-button color="success">Success</ion-button>
<ion-button color="warning">Warning</ion-button>
<ion-button color="danger">Danger</ion-button>
<ion-button color="light">Light</ion-button>
<ion-button color="medium">Medium</ion-button>
<ion-button color="dark">Dark</ion-button>
```

4.2.4 SPEC TS PAGE

Creating a Page

Next, let's check out the `HelloIonicPage` that we are importing. Inside the `src/pages/hello-ionic/` folder, go and open up `hello-ionic.ts`.

You may have noticed that each page has its own folder that is named after the page. Inside each folder, we also see a `.html` and a `.scss` file with the same name. For example, inside of `hello-ionic/` we will find `hello-ionic.ts`, `hello-ionic.html`, and `hello-ionic.scss`. Although using this pattern is not required, it can be helpful to keep things organized.

Below, we see the `HelloIonicPage` class. This creates a Page - an Angular component with all Ionic directives already provided, to be loaded using Ionic's navigation system. Notice that because Pages are meant to be loaded dynamically, they don't need to have a selector. However, the selector is useful in order to override the default styles on a specific page (see `hello-ionic.scss`):

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'page-hello-ionic',  
  templateUrl: 'hello-ionic.html'  
})  
export class HelloIonicPage {  
  constructor() {}  
}
```

4.2.5 Module TS Page

Every Angular application has at least one `NgModule` class, the root module, which is conventionally named `AppModule` and resides in a file named `app.module.ts`. You launch your application by bootstrapping the root `NgModule`.

While a small application might have only one `NgModule`, most applications have many more feature modules. The root `NgModule` for an application is so named because it can include child `NgModules` in a hierarchy of any depth.

4.2.6 TS PAGE

TypeScript: A Static Type Checker

We said earlier that some languages wouldn't allow those buggy programs to run at all. Detecting errors in code without running it is referred to as static checking. Determining what's an error and what's not based on the kinds of values being operated on is known as static type checking.

TypeScript checks a program for errors before execution, and does so based on the kinds of values, it's a static type checker. For example, the last example above has an error because of the *type* of `obj`. Here's the error TypeScript found:

```
const obj = { width: 10, height: 15 };  
const area = obj.width * obj.heighth;
```

4.3 GitHub

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

4.3.1 Pull Operation

Pull request is created when you committed a change in the GitHub project, and you want it to be reviewed by other members. You can commit the changes into a new branch or an existing branch. Once you've created a pull request, you can push commits from your branch to add them to your existing pull request.

4.3.2 Push Operation

The `git push` command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to `git fetch`, but whereas fetching imports commits to local branches, pushing exports commits to remote branches.

4.3.3 Commit Operation

The `git commit` command is one of the core primary functions of Git. Prior use of the `git add` command is required to select the changes that will be staged for the next commit. Then `git commit` is used to create a snapshot of the staged changes along a timeline of a Git projects history.

4.4 TortoiseGit

TortoiseGit is a Windows Shell Interface to Git and based on TortoiseSVN. It's open source and can fully be build with freely available software. Since it's not an integration for a specific IDE like Visual Studio, Eclipse or others, you can use it with whatever development tools you like, and with any type of file. Main interaction with TortoiseGit will be using the context menu of the Windows explorer.

TortoiseGit supports you by regular tasks, such as committing, showing logs, diffing two versions, creating branches and tags, creating patches and so on (see our Screenshots or documentation).

It is developed under the GPL. Which means it is completely free for anyone to use, including in a commercial environment, without any restriction. The source code is also freely available, so you can even develop your own version if you wish to.

4.4.1 Features of TortoiseGit

❖ Easy to use

- all commands are available directly from the Windows Explorer (see screenshots).
- only commands that make sense for the selected file/folder are shown. You won't see any commands that you can't use in your situation.
- See the status of your files directly in the Windows explorer (see screenshots)
- descriptive dialogs, constantly improved due to user feedback
- allows moving files by right-dragging them in the Windows explorer

❖ Powerful commit dialog

- integrated spell checker for log messages
- auto completion of paths and keywords of the modified files

- text formatting with special chars
- ❖ Per project settings
 - minimum log message length to avoid accidentally committing with an empty log message
 - language to use for the spell checker
- ❖ Integration with issue tracking systems
 - TortoiseGit provides a flexible mechanism to integrate any web based bug tracking system.
 - A separate input box to enter the issue number assigned to the commit, or coloring of the issue number directly in the log message itself
 - When showing all log messages, an extra column is added with the issue number. You can immediately see to which issue the commit belongs to.
 - Issue numbers are converted into links which open the webbrowser directly on the corresponding issue
 - Optional warning if a commit isn't assigned to an issue number
- ❖ Helpful Tools
 - TortoiseGitMerge (see screenshot and the TortoiseGitMerge manual)
 1. Shows changes you made to your files
 2. Helps resolving conflicts
 3. Can apply patchfiles you got from users without commit access to your repository
 - TortoiseGitBlame: to show blames of files. Shows also log messages for each line in a file. (see screenshot)
 - TortoiseGitIDiff: to see the changes you made to your image files (see screenshot)
- ❖ Available in many languages
- ❖ TortoiseGit is stable

- Before every release, we create one or more preview releases for "adventurous" people to test first. This helps finding bugs very early so they won't even get into an official release.
- A custom crash report tool is included in every TortoiseGit release which helps us fix the bugs much faster, even if you can't remember exactly what you did to trigger it.

CHAPTER 5

APK TESTING

SOFTWARE TESTING

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Some prefer saying Software testing as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application Under Test (AUT). This tutorial introduces testing software to the audience and justifies its importance.

PROGRAM TESTING

Program Testing in software testing is a method of executing an actual software program with the aim of testing program behavior and finding errors. The software program is executed with test case data to analyse the program behavior or response to the test data. A good program testing is one which has high chances of finding bugs.

We have already learned about the factors that influence how to choose the test techniques. Now let's dive into the actual test techniques. At a high level, the test techniques can be categorized into two groups – Black-box techniques and White-box techniques. In this article, we will discuss the Black-box testing techniques. We will cover each method in detail as a separate article, but let's give you an overview of them here!

5.1 BLACK BOX TESTING

The black box is a testing type where the tester doesn't know the internal design of the component or system which undergoes testing. So if we don't know the internal design, how do we ensure that our test cases can still find defects and provide excellent coverage? It is where Black Box testing techniques come into the picture. These are scientific and time tested techniques that will help us get maximum coverage in minimum test cases.

How to do Black Box Testing

We already know what Black box testing is, and we are going to learn its techniques (which is an efficient way of creating test cases for a black box testing type). But before we dive in there, it's essential to know a few critical characteristics of Black box testing. Additionally, it's equally important to understand what it takes to execute a black box test type successfully.

Understand the Requirements

The first step to execute a black box testing is to understand the system requirements thoroughly. As the internal system design is not known to us, the system requirements act as the source of truth to create effective test cases. Any understanding gap must be clarified, and the tester should ensure that specifications are detailed enough to write a black-box test case successfully.

Effective Test case Creation

Once the requirements are understood, the next step is to create test cases. The biggest challenge is to write test cases that are effective enough to find defects (Remember – We don't know the internal design !). We will discuss the actual techniques in detail later, but keep in mind that it's essential to ensure that our test cases cover below.

Happy Paths : These are the Critical business scenarios where we give positive inputs. Additionally, happy paths involve the most commonly used steps. For instance, a successful login if I provide a valid user id and password.

Alternate Paths : These are critical business scenarios that involve positive inputs. However, the steps are not the most commonly used. E.g., I should be able to automatically log in if my session doesn't expire without inputting user id and password again!

Negative Paths : These are business scenarios where we give negative inputs, and we expect the system to throw error message (Of course, a user-friendly one !)

Adaptive Execution : We all do execution, and we compare the actual result with expected results, so what is adaptive about it? Well, the black box testing is guesswork- a Scientific one, though! So, what happens when you find some critical defects in a particular area but don't find any in other areas? In such cases, we need to adapt our execution to focus more on the buggy areas – some developers had a bad day! Additionally, we review our test coverage and techniques where we are not finding bugs.

Defects Logging and Closure – The last step is to log the defects, and retest it once it's fixed. Everyone does that, but what you can do better is to test scenarios around the defect fix. Often a fix leads to another defect, and it's best to catch it while retesting the defect.

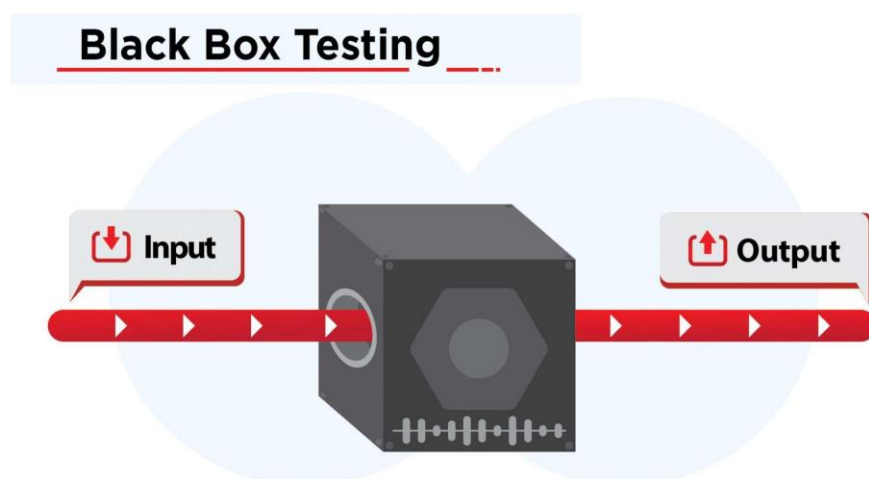


Figure 5.1 Black Box Testing

5.2 WHITE BOX TESTING

White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term "WhiteBox" was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

What do you verify in White Box Testing?

White box testing involves the testing of the software code for the following

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against

expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do when testing an application using the white box testing technique:

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include Manual Testing, trial, and error testing and the use of testing tools as we will explain further on in this article.

TYPES OF WHITE BOX TESTING

White box testing encompasses several testing types used to evaluate the usability of an application, block of code or specific software package. There are listed below --

Unit Testing : It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing. Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.

Testing for Memory Leaks : Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

Apart from above, a few testing types are part of both black box and white box testing. They are listed as below

White Box Penetration Testing : In this testing, the tester/developer has full information of the application's source code, detailed network information, IP addresses involved and all server information the application runs on. The aim is to attack the code from several angles to expose security threats

White Box Mutation Testing: Mutation testing is often used to discover the best coding techniques to use for expanding a software solution.

Advantages of White Box Testing

- Code optimization by finding hidden errors.
- White box tests cases can be easily automated.
- Testing is more thorough as all code paths are usually covered.
- Testing can start early in SDLC even if GUI is not available.

Disadvantages of White Box Testing

- White box testing can be quite complex and expensive.
- Developers who usually execute white box test cases detest it. The white box testing by developers is not detailed can lead to production errors.

- White box testing requires professional resources, with a detailed understanding of programming and implementation.
- White-box testing is time-consuming, bigger programming applications take the time to test fully.

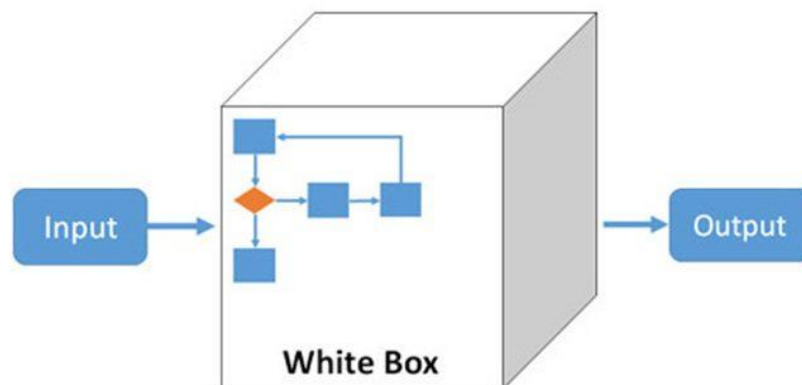


Figure 5.2 White box Testing

5.3 TESTING RESULTS

Firebase Test Lab lets you test your app on a range of devices and configurations. This Get Started guide provides an implementation path for you to follow, as well as an introduction to Test Lab's Android offerings.

For information about Test Lab quotas and pricing plans, see [Usage, Quotas, and Pricing](#).

Key concepts

When you run a test or a set of test cases against devices and configurations you've selected, Test Lab runs the test against your app in a batch, then displays the results as a test matrix.

Devices × Test Executions = Test Matrix

Device

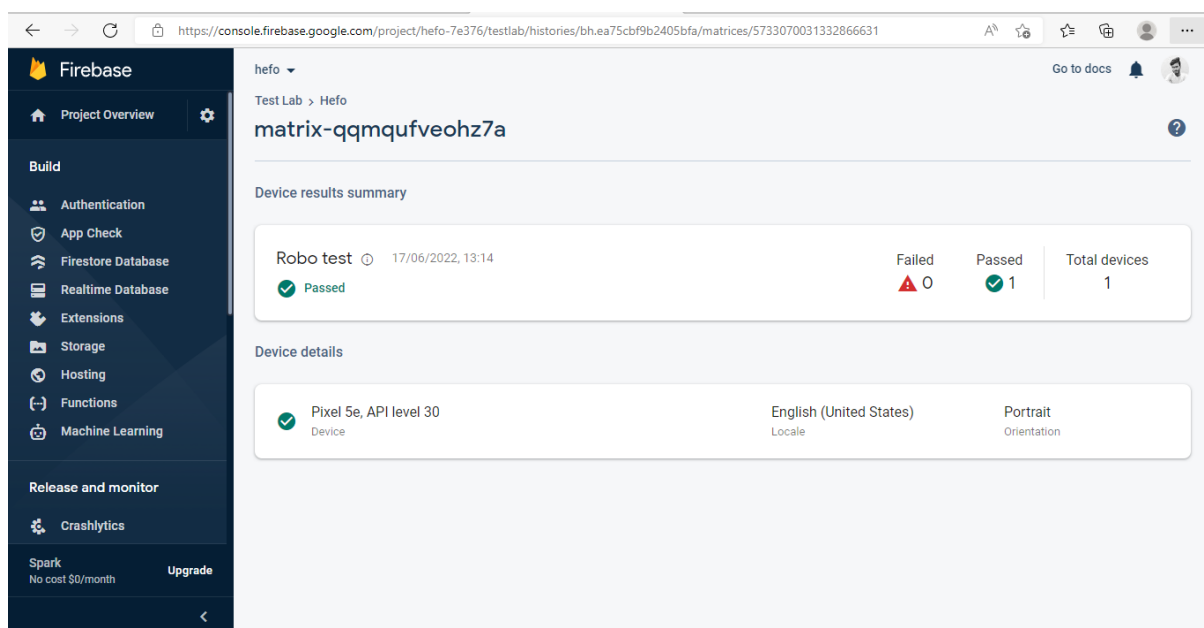
A physical or virtual device (Android only) you run a test on, such as a phone, tablet, or wearable device. Devices in a test matrix are identified by device model, OS version, screen orientation, and locale (also known as geography and language settings).

Test execution

A test (or a set of test cases) to be run on a device. You can run one test per device, or optionally shard the test and run its test cases on different devices.

Test matrix

Contains the statuses and test results for your test executions. If any test execution in a matrix fails, the whole matrix fails.



The screenshot shows the Firebase Test Lab console interface. The left sidebar contains the Firebase logo and navigation links for Project Overview, Build (Authentication, App Check, Firestore Database, Realtime Database, Extensions, Storage, Hosting, Functions, Machine Learning), Release and monitor (Crashlytics), and Spark. The main content area displays the test results for a specific matrix. The breadcrumb path is 'hefo > Test Lab > Hefo'. The matrix name is 'matrix-qmqufveohz7a'. Under 'Device results summary', a 'Robo test' is shown with a 'Passed' status, a green checkmark, and a timestamp of '17/06/2022, 13:14'. A table shows 'Failed' (0), 'Passed' (1), and 'Total devices' (1). Under 'Device details', a table lists the device as 'Pixel 5e, API level 30', the locale as 'English (United States)', and the orientation as 'Portrait'.

Device results summary
Robo test 17/06/2022, 13:14
Passed
Failed: 0, Passed: 1, Total devices: 1

Device details
Pixel 5e, API level 30
English (United States)
Portrait

Figure 5.3.1 Test Results Passed

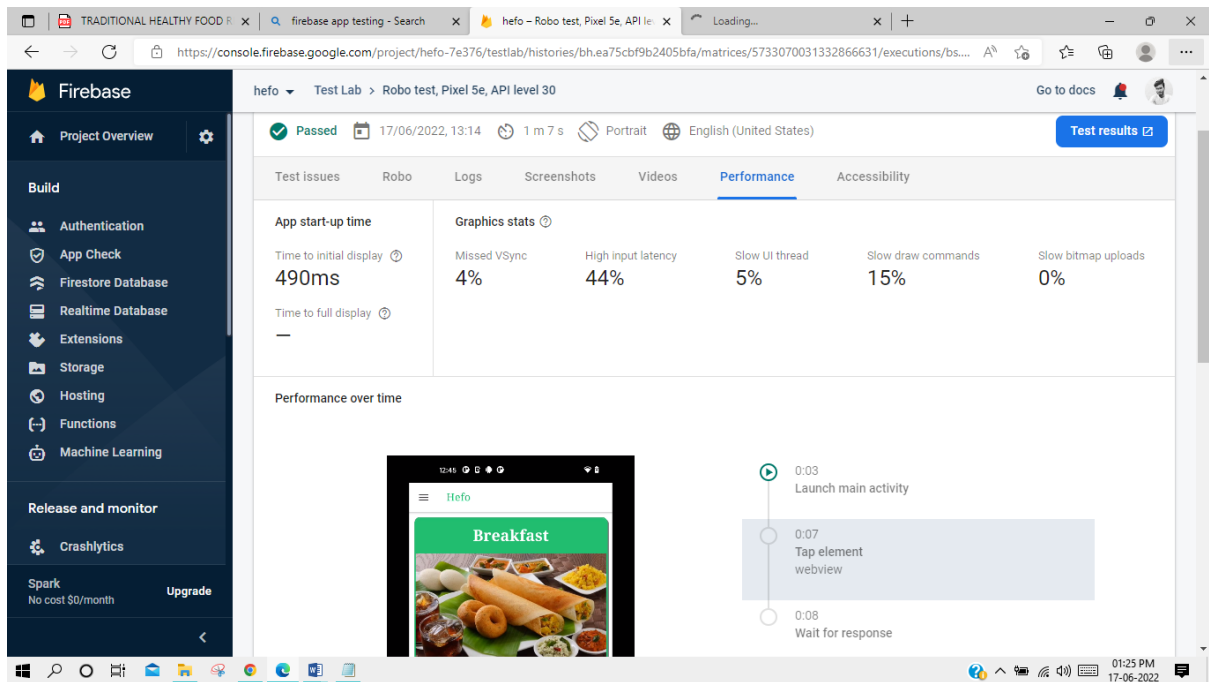


Figure 5.3.2 Test Results Performance

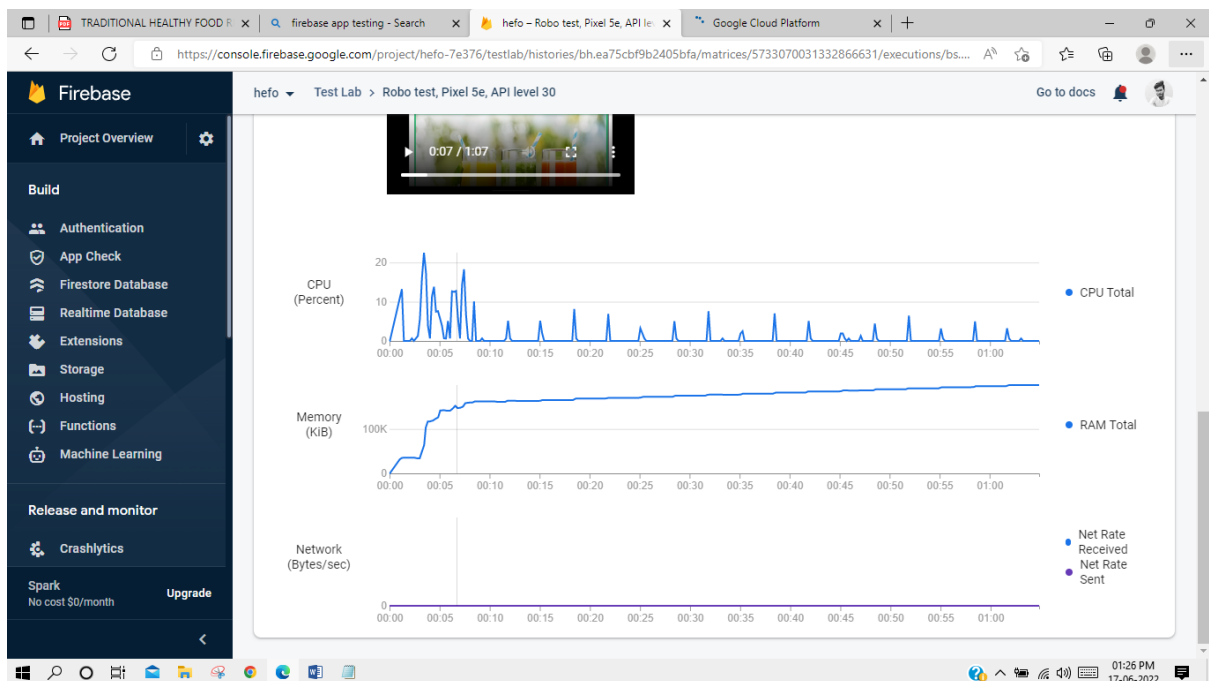


Figure 5.3.3 Test Results Performance Graph

CHAPTER 6

CONCLUSION AND FUTURE WORK

- We are working on login page and BMI pages for getting the user information.
- For giving separate suggestion to every working peoples who wants to maintain the health perfect.
- In future we'll get this app from google play store for less work more health benefits to working people.

APPENDIX

A.1 Home Routing Module TS

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomePage } from './home.page';
const routes: Routes = [
  {
    path: "",
    component: HomePage
  },
  {
    path: 'breakfast',
    loadChildren: () => import('./breakfast/breakfast.module').then( m =>
m.BreakfastPageModule)
  },
  {
    path: 'lunch',
    loadChildren: () => import('./lunch/lunch.module').then( m =>
m.LunchPageModule)
  },
  {
    path: 'dinner',
    loadChildren: () => import('./dinner/dinner.module').then( m =>
m.DinnerPageModule)
```

```

    },
    {
      path: 'snacks',
      loadChildren: () => import('./snacks/snacks.module').then( m =>
m.SnacksPageModule)
    },
    {
      path: 'juice',
      loadChildren: () => import('./juice/juice.module').then( m =>
m.JuicePageModule)
    }
  ];
  @NgModule({
    imports: [RouterModule.forChild(routes)],
    exports: [RouterModule],
  })
  export class HomePageRoutingModule { }

```

A.2 Home Module TS

```

import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { IonicModule } from '@ionic/angular';
import { HomePageRoutingModule } from './home-routing.module';
import { HomePage } from './home.page';
@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    IonicModule,
    HomePageRoutingModule
  ],
  declarations: [HomePage]
})
export class HomePageModule { }

```

A.3 Home Page HTML

```

<ion-header>
  <ion-toolbar>

```

```

    <ion-buttons slot="start">
      <ion-menu-button></ion-menu-button>
    </ion-buttons>
    <ion-title>Hefo</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content>
  <ion-card *ngFor="let d of dishCards" [routerLink]="[d.url]">
    <ion-card-header color="success">
      <ion-card-title>{{ d.title }}</ion-card-title>
    </ion-card-header>
    <ion-img [src]="[d.image]"></ion-img>
    <ion-card-content>
      {{ d.description }}
    </ion-card-content>
  </ion-card>
</ion-content>

```

A.4 Home Page SCSS

```

ion-title{
  color: #2dd36f;
  font-weight: 500;
  min-height: 18px;
  font-family: 'Times New Roman';
}
ion-card{
  border-radius: 10px;
  box-shadow: 3px 3px #888888;
  border: 2px solid #2dd36f;
}
ion-card-title{
  text-align: center;
  font-weight: bold;
  font-size: 30px;
  min-height: 18px;
  font-family: Georgia;
}
ion-card-content{

```



```

    font-size: 16px;
    text-align: justify;
    font-family: 'Times New Roman';
    font-weight: 800;
}

```

A.5 Home Page Spec TS

```

import { ComponentFixture, TestBed, waitForAsync } from
'@angular/core/testing';
import { IonicModule } from '@ionic/angular';
import { HomePage } from './home.page';
describe('HomePage', () => {
  let component: HomePage;
  let fixture: ComponentFixture<HomePage>;
  beforeEach(waitForAsync(() => {
    TestBed.configureTestingModule({
      declarations: [ HomePage ],
      imports: [IonicModule.forRoot()]
    }).compileComponents();
    fixture = TestBed.createComponent(HomePage);
    component = fixture.componentInstance;
    fixture.detectChanges();
  }));

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});

```

A.6 Home page TS

```

import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
  styleUrls: ['./home.page.scss'],
})
export class HomePage implements OnInit {
  public dishCards = [
    { title: 'Breakfast',

```

description: ' The first meal of the day, usually eaten in the morning. You should put more protein in your breakfast so you will grow. A meal consisting of food normally eaten in the morning, which may typically include eggs, sausages, toast, bacon, etc. We serve breakfast all day. A meal after fasting, or food in general.',

image: 'assets/breakfast.jpg',

url: 'breakfast'

},

{ title: 'Fresh Juice',

description:'Juices. While not a fruit, juice is sweet and consumed similarly to fruit juice. This usually refers to sweetened juice. Used in traditional medicine consumed and also used topically. Is a blend of passionfruit, orange and guava juices hence POG .',

image: 'assets/fresh juice.jfif',

url: 'juice'

},

{ title: 'Lunch',

description:'Lunch is the second meal of the day, after breakfast, and varies in size by culture and region. According to the Oxford English Dictionary (OED), the etymology of lunch is uncertain. It may have evolved from lump in a similar way to hunch, a derivative of hump, and bunch, a derivative of bump.',

image: 'assets/lunch.jpg',

url: 'lunch'

},

{ title: 'Snacks',

description:'Snack foods are a significant aspect of Indian cuisine, and are sometimes referred to as chaat . A traditional Indian snack, it is a fruit leather made out of mango pulp mixed with concentrated sugar solution and sun dried. It is a part of the South Indian and North Indian cuisine and is available in numerous varieties all over North India.',

image: 'assets/snacks.jpeg',

url: 'snacks'

},

{ title: 'Dinner',

description:'Dinner usually refers to what is in many Western cultures the largest and most formal meal of the day, which some Westerners eat in the

midday. Historically, the largest meal used to be eaten around midday, and called dinner.',

```
    image: 'assets/dinner.jpg',  
    url: 'dinner'  
  },  
]  
constructor() { }  
ngOnInit() {  
}  
}
```

REFERENCES

1. **Ionic Framework UI Components**
<https://ionicframework.com/docs/components>
2. **GitHub**
<https://github.com/Yokesh-Murugan/Hefo>
3. **TortoiseGit**
<https://tortoisegit.org/docs/>
4. **Angular JS**
<https://angular.io/docs>
5. **Html tutorial**
<https://www.w3schools.com/html/>
6. **CSS tutorial**
<https://www.w3schools.com/css/>
7. **Healthline**
<https://www.healthline.com/nutrition/food-and-nutrients>