RizNotes

Concept Development

Every day, a tonne of information is presented to you. You want to keep track of and organise the important things you have seen, heard, read, thought about and discussed so you do not have to waste time trying to remember or recreate them. Previously, you would scribble notes on scraps of paper and display them on your desk. This may get confusing and unorganised, especially when you have a lot of scribble notes screaming right in front of your face. It is difficult to filter through them, find out which takes priority or even searching a particular note will take time.

As technology evolves, taking down information is easy and is at one's fingertips. Note taking apps are readily available in our mobile phones, and it usually offers a basic note-taking tool or a rich array of features that increase productivity.

Advantages

What can we benefit from digital note taking apps? Firstly, it is about never losing your notes again. As we usually scribble or jot down our thoughts on pieces of paper, we tend to lose them or it can become very cluttered. In addition, you probably do not come up with your best ideas while you are sitting at your desk, right? Digital technologies have many benefits, one of which is that almost everything is now accessible. We always have our smartphones with us. We can therefore access our notes app whenever we want to jot down some ideas or take notes during a class. Additionally, running out of pages has become obsolete.

Digital notes will improve your preparation by revising and reviewing. You can always edit them accordingly, and can correct errors without leaving unsightly eraser stains on your paper. Not only that, you can simply search for a particular note easily in no time! Everything is simple and organised, which creates a pleasant experience for the user.

Lastly, saving the Earth. Consider the fact that, yes, paper is made from trees, making it a limited resource. In the long run, taking handwritten notes is undoubtedly more environmentally friendly than wasting all that paper.

Aim

These note taking apps are a crowded market with some very big players. Therefore, I would like to target a niche area, which is to create a simple yet minimalistic app, primarily focused on just taking down annotations, ideas, and notes on the go.

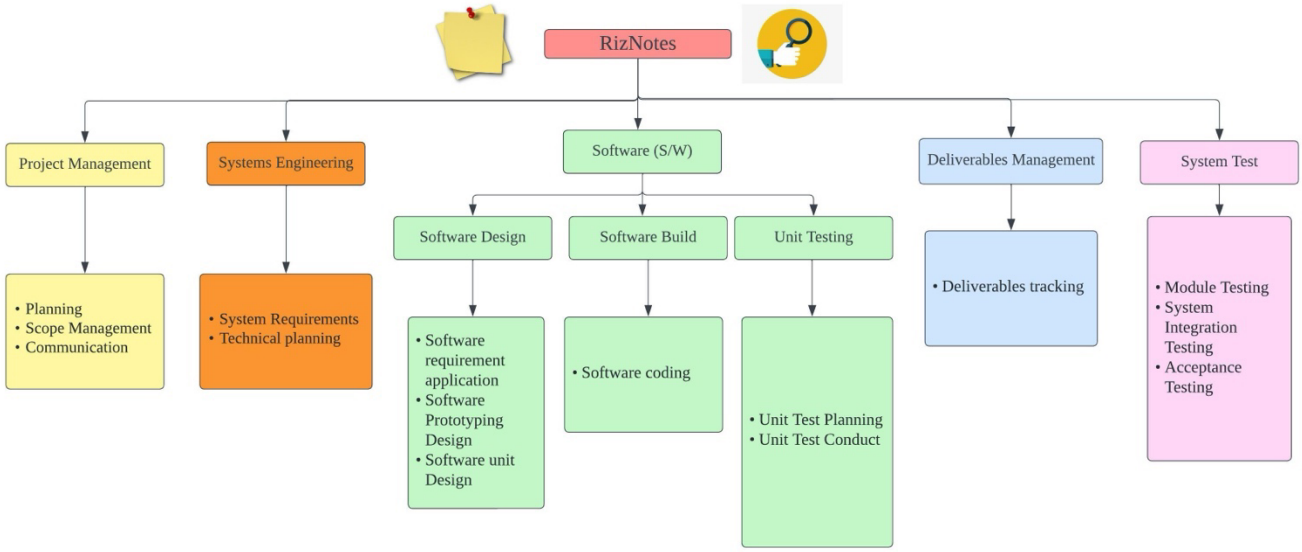
My Approach: User-Centred Design (UCD)
The main of objective of the app was to be user friendly to a large extent. Therefore, it will not be too difficult for me to develop and maintain, keeping the complexity to the minimum. The

app must be easy to use because it is meant for frequent use on a daily basis. It therefore requires a user interface that is simple to use, clear with a precise set of navigation.

The note-taking app I am proposing will be called RizNotes, which will be used throughout the report to describe my software.

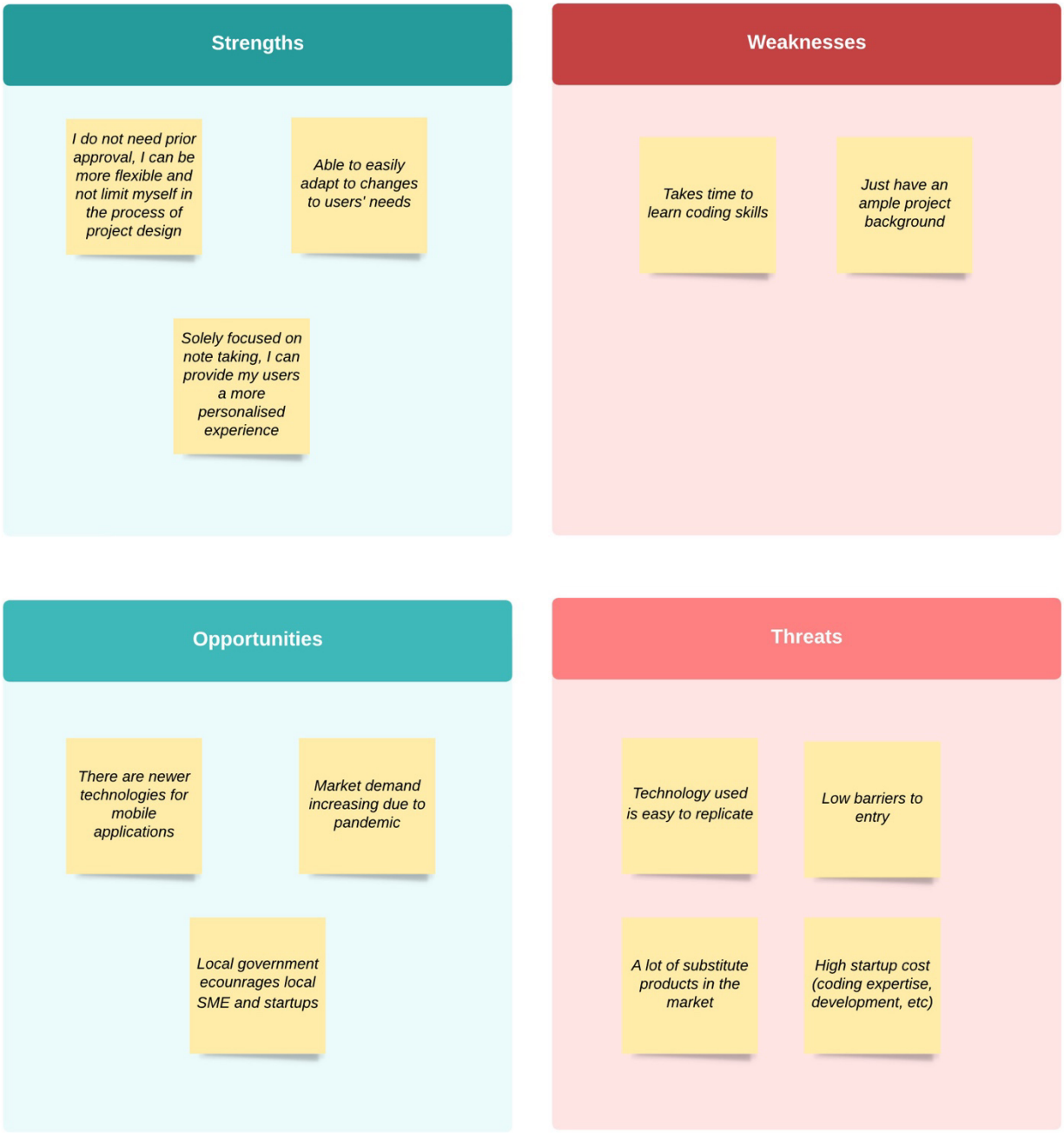
Deliverables Contents

Work Breakdown Structure



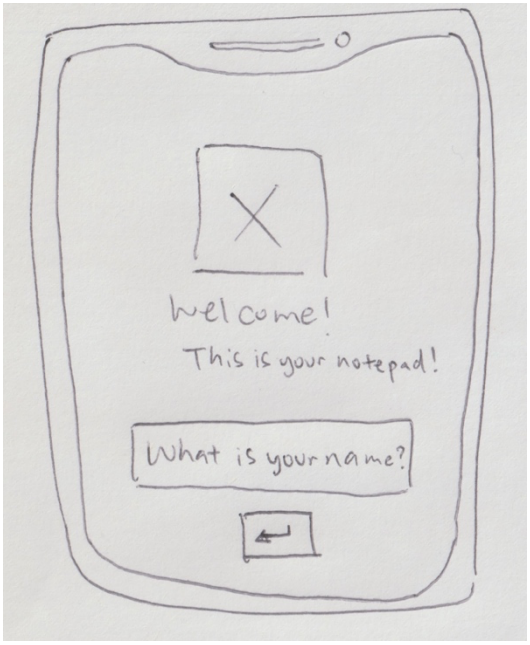
The WBS that is depicted below was created by me to successfully integrate the project's scope and guarantee that the project plans are congruent. Planning, obtaining requirements, software designing, tracking, and testing were the five stages I divided my work into.

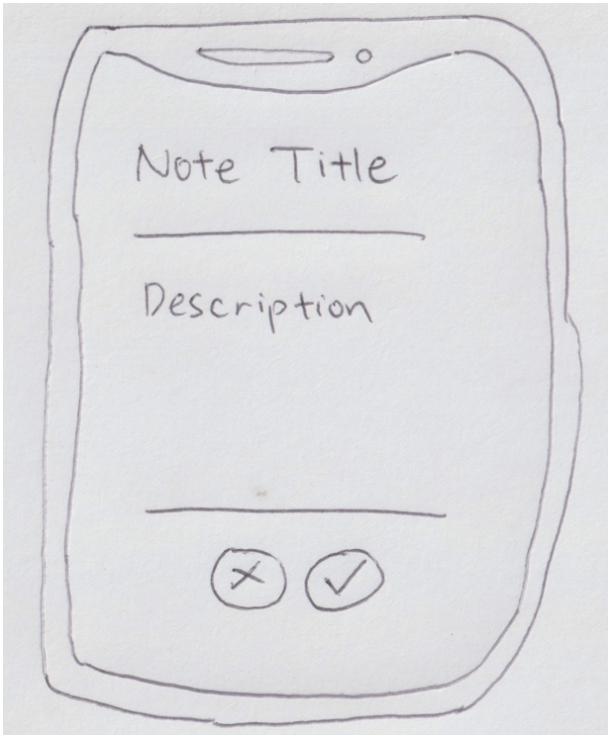
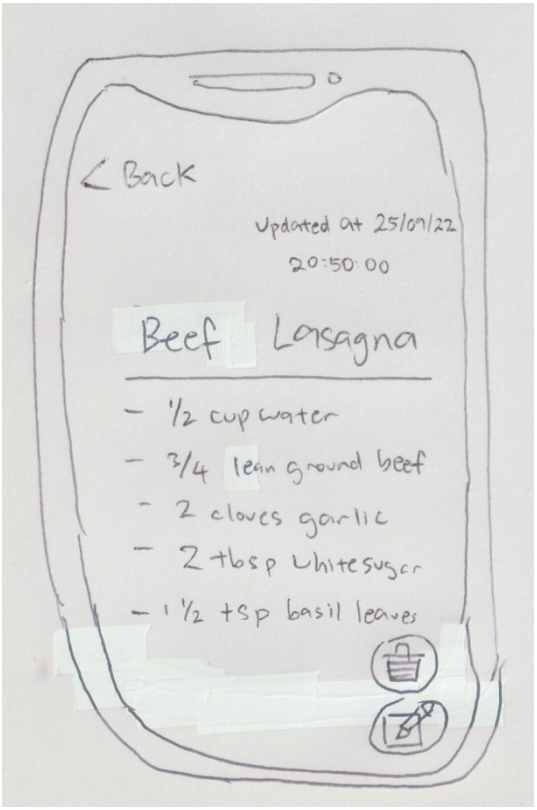
SWOT Analysis

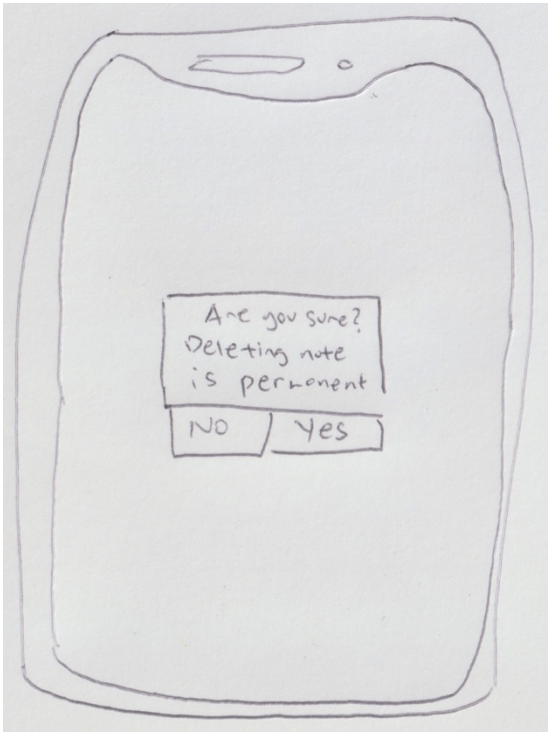


A SWOT analysis' main objective is to help us comprehend all the factors that influence business decisions. Before making any further commitments, a SWOT analysis must be performed. I am better able to comprehend the benefits and drawbacks of our suggested system thanks to this SWOT analysis, as well as my own strengths and weaknesses. Additionally, this will enable me to determine the future marketing and promotion opportunities for my product.

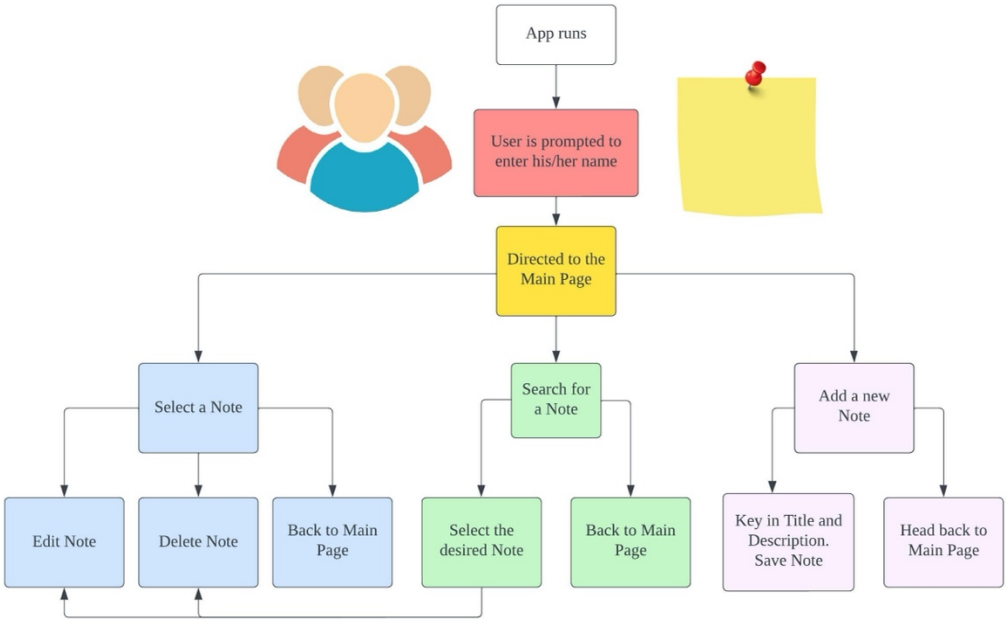
Wireframing

Welcome Page	**Main Page**
	
This will be how the welcome page looks like.	This will be how the main page of the note will look like.

Adding a New Note	**Selecting a Note**
	
This is what the user will see when they want to add a new note.	This is what the user will see when he or she selects a particular note. User can either delete or edit the app respectively.

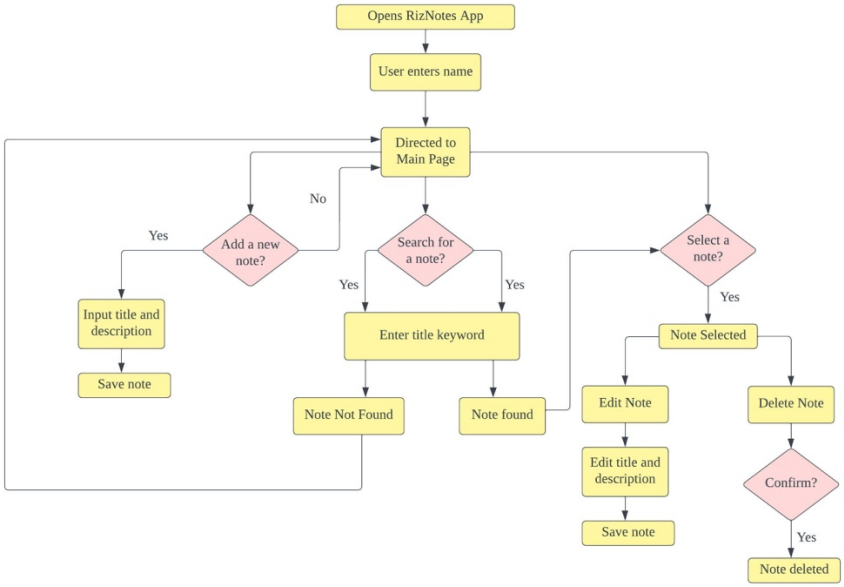
Deleting a Note	**Note is not found**
	
This is what the user will be prompted with a message confirming if he/she wants to delete the note.	This is what the user will see if the note they are looking for is not found.

UML Diagram



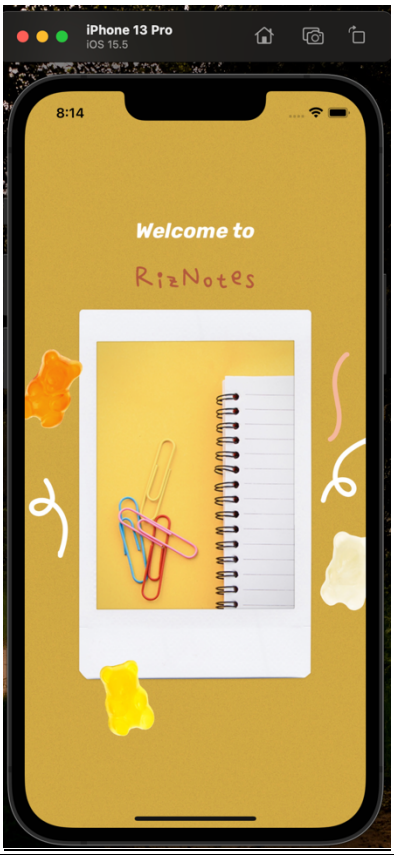
This diagram also aids in visualising every feature/function that I intend to include in my application.

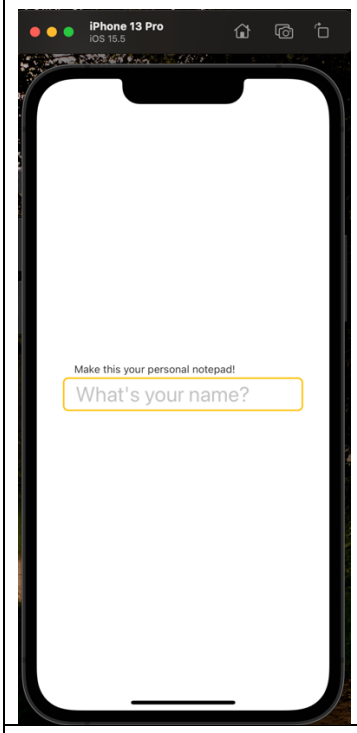
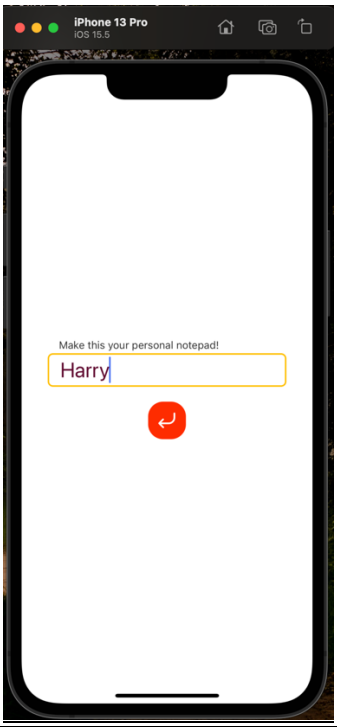
User Flow Diagram

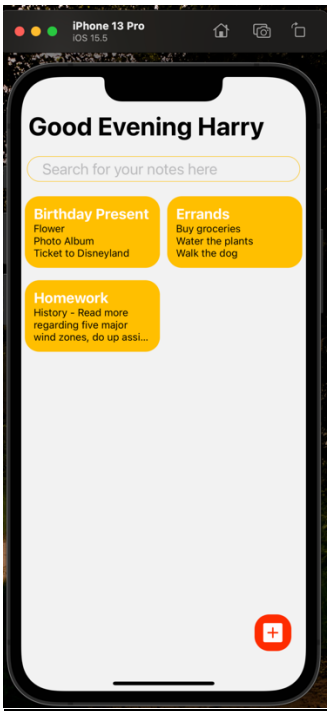
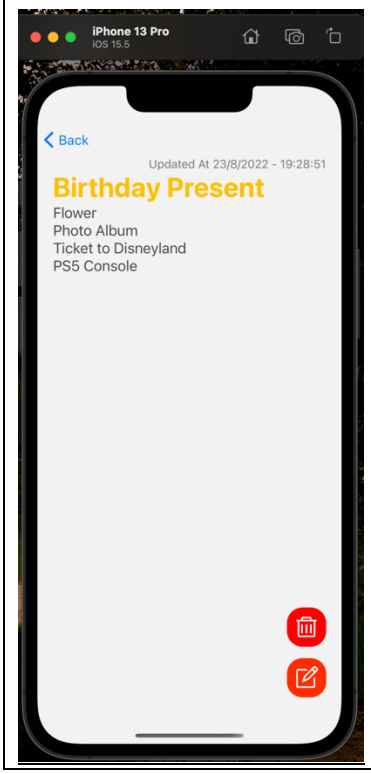


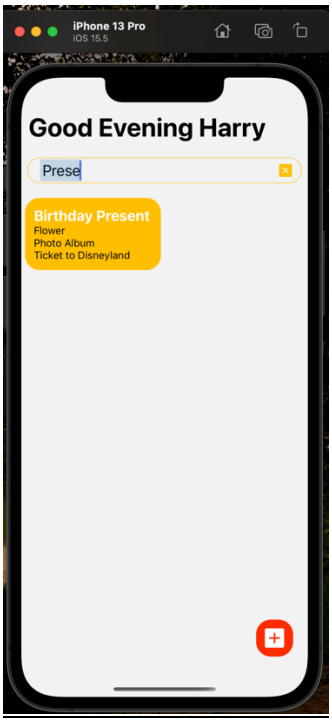
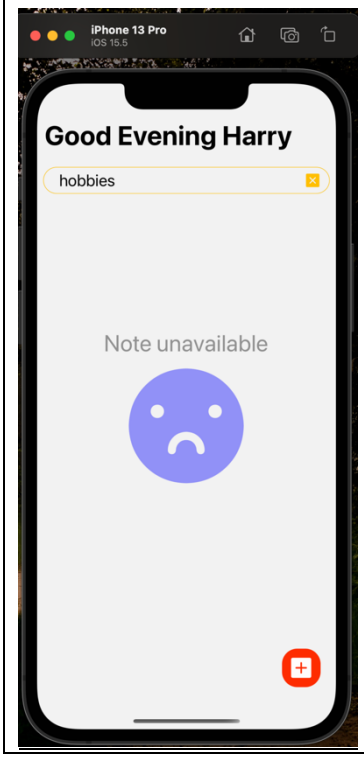
This user flow diagram will be primarily used to determine the flow of the RizNotes application after considering the customer experience and user needs.

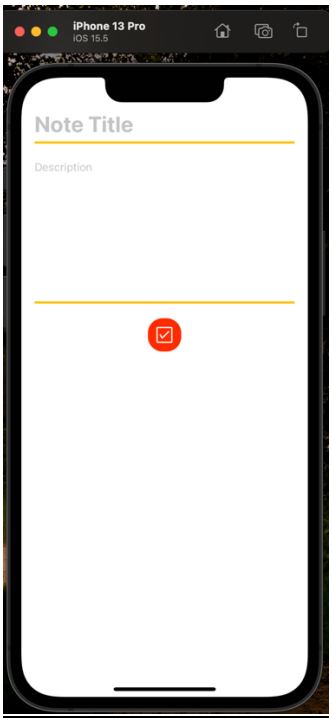
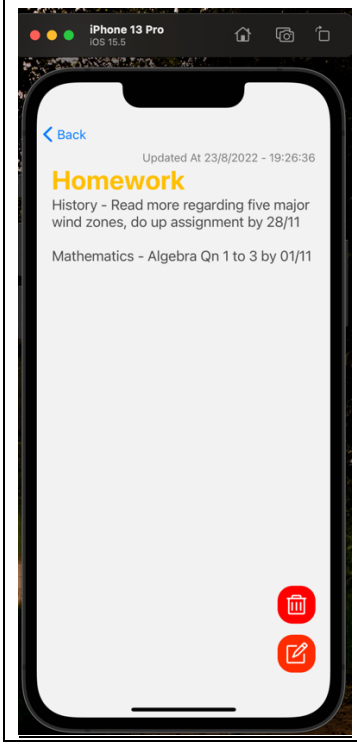
Final Prototype

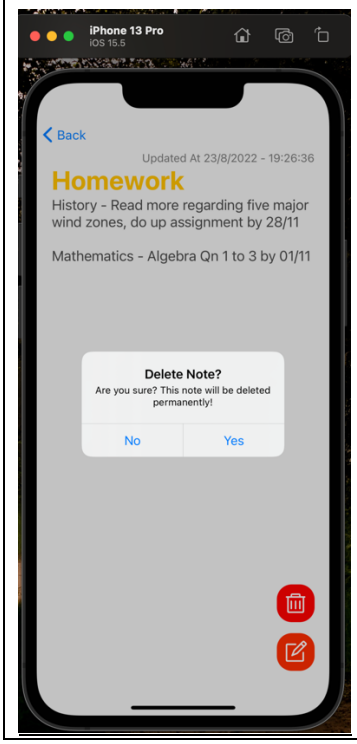
Splash Screen 	When the app is fully loaded and the user opens it, this splash screen will appear, welcoming the user. “Welcome to RizNotes”

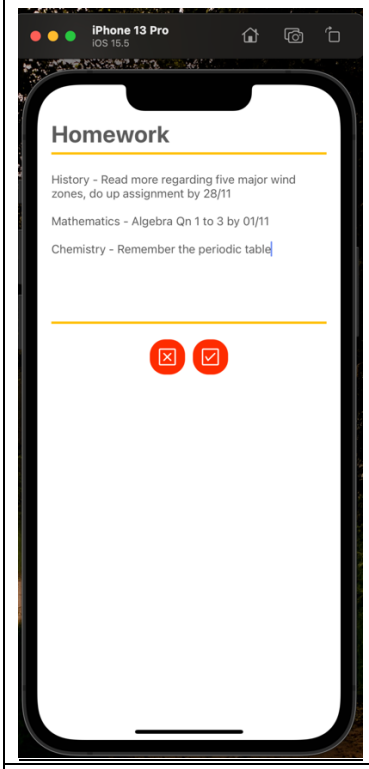
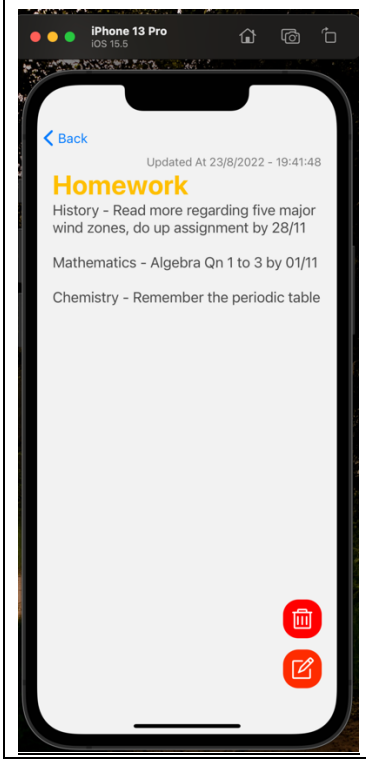
Welcome Screen 	The splash screen will automatically transition to this Welcome Page. Here, the user will be required to enter their name.
Entering your Name (Welcome Screen) 	Once the user input their name, an enter button in red will appear.

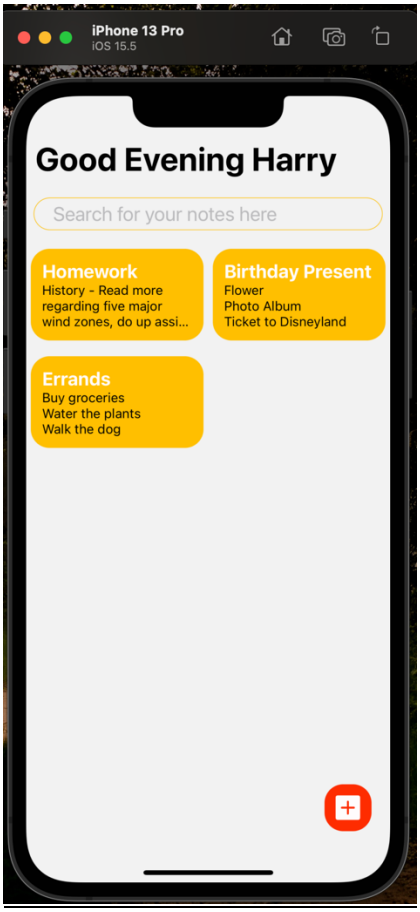
Main Note Screen 	Once the red enter button is pressed, the user will be led into this main page of the notepad app.
Selecting a Note Screen 	This page will appear once a note is selected.

Searching for Note Screen 	The user can search for their desired notes. To do so, they can navigate to the search box, type the note they would like to find, and it will filter through all the notes automatically.
Note Not Found Screen 	However, if the note is nowhere to be found, this page on the left will appear, stating that “Note unavailable” with a sad emoticon.

Adding New Note Screen 	Clicking the Red Add Button on the bottom right on the Main Page will direct you to this page, which will allow the user to add a new note. The user has the option to add a title and description to their notes.
Selecting Note Screen 	This page will appear once the user selects a particular note. On the bottom right of the page, there is a thrash icon, which will allow the user to delete the note. Just directly below the trash icon, there is a paper & pen icon, pressing it will allow the user to edit their note.

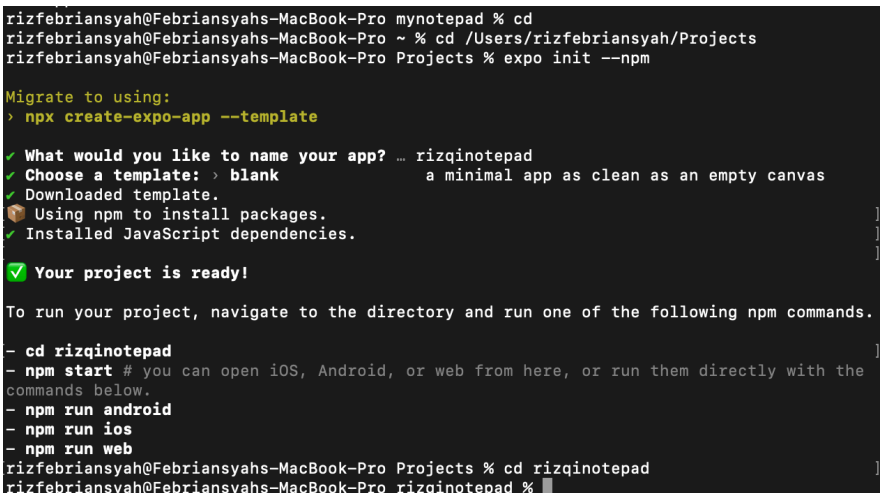
Editing Note Screen 	Clicking on the pen & paper icon, which will allow the user to edit their note, and afterwards will be directed to the page seen on the left. User is then able to edit accordingly. Clicking the cross button will cancel any edits being done, whereas the tick button will save any edits which have been done.
Deleting Note Screen 	If the user presses the trash icon, the user will be prompted with a message whether they really want to delete the note, as this action will be permanent and irreversible.

Editing Note Screen 	The image on the left shows that the user added some annotations to his note. “Chemistry – Remember the periodic table” and proceeds to save it.
Edited Note Screen 	The image on the left shows the edited note. The date and time of when the note was last updated will always be reflected on the top right-hand corner.

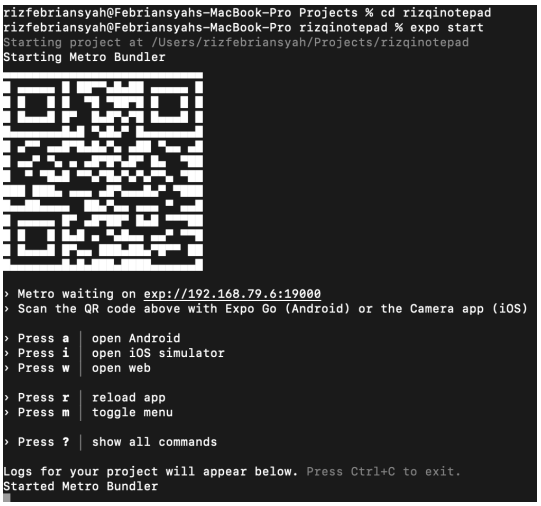
Main Page Screen (After updating note)  	The latest edited note will always appear at the top left, displacing the rest of the notes.

Development

Firstly, we need to initialize a new expo project folder which I called rizqinotepad.
This can be done by going to a desired directory, and afterwards input:
`expo init --npm`



To run the project, proceed to the directory, and afterwards input:
`cd rizqinotepad`
`expo start`



Opening the project via Visual Studio Code, I created a new ‘components’ and ‘screens’ folder.
Under components folder, I created these 8 files:

- Notepad.js
- NotepadMissing.js
- NotepadModal.js
- NotepadProvider.js
- NotepadStructure.js

- SearchField.js
- Colours.js
- EnterButton.js

Under screens folder, I created these 2 files:

- WelcomePage.js
- MainPage.js

AysncStorage is a straightforward, asynchronous, and by default unencrypted module that enables offline data persistence in React Native apps. A key-value storage system is used for data persistence. I will be using AsyncStorage to store the notes locally.

To install, input this into the terminal:

```
npm install @react-native-async-storage/async-storage
```

```
rizfebriansyah@Febriansyahs-MacBook-Pro rizqinotepad % npm install @react-native-async-storage/
async-storage


added 3 packages, and audited 1139 packages in 2s

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

found 0 vulnerabilities
```

To access the AsyncStorage component, need to import it by inputting:

```
import AsyncStorage from '@react-native-async-storage/async-storage';
```

My app can switch between screens using Stack Navigator, where each new screen is stacked on top of the previous one. The stack navigator is set up by default to have the recognisable iOS and Android look and feel: new screens slide in from the right on iOS, while Android uses the OS default animation. However, you can modify the animations to suit your needs.

To install, input this into the terminal:

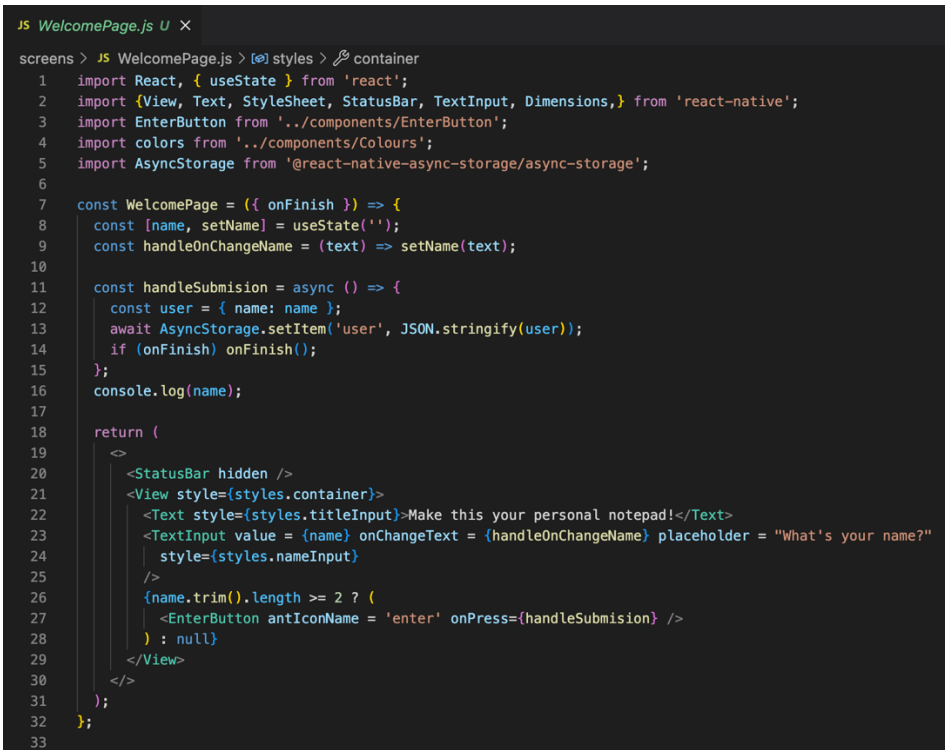
```
npm install @react-navigation/stack
```

```
rizfebriansyah@Febriansyahs-MacBook-Pro rizqinotepad % npm install @react-navigation/stack

added 25 packages, changed 1 package, and audited 1178 packages in 7s

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

found 0 vulnerabilities
```

To access the stack navigator component, need to import it by inputting:

```
import { createStackNavigator } from '@react-navigation/stack';
```

WelcomePage.js



```
JS WelcomePage.js U ×
screens > JS WelcomePage.js > styles > container
1   import React, { useState } from 'react';
2   import {View, Text, StyleSheet, StatusBar, TextInput, Dimensions,} from 'react-native';
3   import EnterButton from '../components/EnterButton';
4   import colors from '../components/Colours';
5   import AsyncStorage from '@react-native-async-storage/async-storage';
6
7   const WelcomePage = ({ onFinish }) => {
8     const [name, setName] = useState('');
9     const handleOnChangeName = (text) => setName(text);
10
11    const handleSubmision = async () => {
12      const user = { name: name };
13      await AsyncStorage.setItem('user', JSON.stringify(user));
14      if (onFinish) onFinish();
15    };
16    console.log(name);
17
18    return (
19      <>
20        <StatusBar hidden />
21        <View style={styles.container}>
22          <Text style={styles.titleInput}>Make this your personal notepad!</Text>
23          <TextInput value = {name} onChangeText = {handleOnChangeName} placeholder = "What's your name?"
24            style={styles.nameInput}
25          />
26          {name.trim().length >= 2 ? (
27            <EnterButton antIconName = 'enter' onPress={handleSubmision} />
28          ) : null}
29        </View>
30      </>
31    );
32  };
33
```

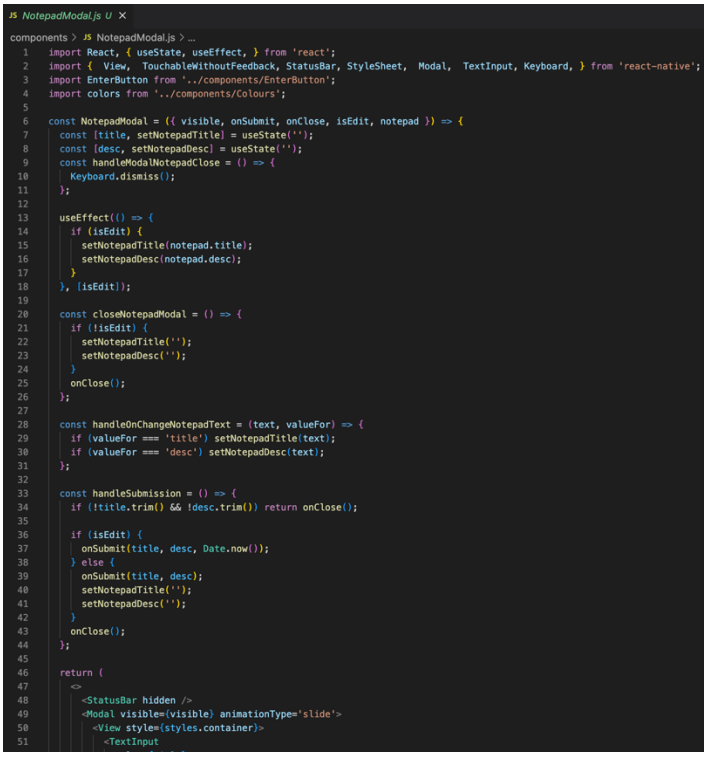
This will be the code responsible when the user first opens the app. They will be greeted by a page, asking them to enter their name. Once the user enters their name, their name will be stored using AsyncStorage, and this notepad will be personalised to that user himself/herself.

NotepadStructure.js

```
import React, { useState } from 'react';
import { StyleSheet, ScrollView, Alert, Text, View } from 'react-native';
import { useHeaderHeight } from '@react-navigation/elements';
import AsyncStorage from '@react-native-async-storage/async-storage';
import EnterButton from '../components/EnterButton';
import colors from '../components/Colours';
import NotepadModal from '../components/NotepadModal';
import { useNotepads } from '../components/NotepadProvider';


const NotepadStructure = props => {
  const [notepad, setNotepad] = useState(props.route.params.notepad);
  const { setNotepads } = useNotepads();
  const [showNotepadModal, setShowNotepadModal] = useState(false);
  const headerHeight = useHeaderHeight();
  const [isEdit, setIsEdit] = useState(false);

  const formatTimeDate = ms => {
    const date = new Date(ms);
    const day = date.getDate();
    const month = date.getMonth() + 1;
    const year = date.getFullYear();
    const hrs = date.getHours();
    const min = date.getMinutes();
    const sec = date.getSeconds();

    return `${day}/${month}/${year} - ${hrs}:${min}:${sec}`;
  };

  const deleteNotepad = async () => {
    const result = await AsyncStorage.getItem('notepads');
    let notepads = [];
    if (result !== null) notepads = JSON.parse(result);

    const newNotepads = notepads.filter(n => n.id !== notepad.id);
    setNotepads(newNotepads);
    await AsyncStorage.setItem('notepads', JSON.stringify(newNotepads));
    props.navigation.goBack();
  };

  const handleNotepadUpdate = async (title, desc, time) => {
    const result = await AsyncStorage.getItem('notepads');
    let notepads = [];
    if (result !== null) notepads = JSON.parse(result);

    const newNotepads = notepads.filter(n => {
      if (n.id === notepad.id) {
        n.title = title;
        n.desc = desc;
        n.isUpdated = true;
        n.time = time;
```

This will be the code responsible for the overall structure and look when a user selects a note. It is also responsible when a user decides to update or delete a note, and this will be updated real time in AsyncStorage.

NotepadModal.js



```
components > JS NotepadModal.js > ...
import React, { useState, useEffect, } from 'react';
import {  View,  TouchableWithoutFeedback, StatusBar, StyleSheet,  Modal,  TextInput, Keyboard, } from 'react-native';
import EnterButton from '../components/EnterButton';
import colors from '../components/Colours';

const NotepadModal = ({ visible, onSubmit, onClose, isEdit, notepad }) => {
  const [title, setNotepadTitle] = useState('');
  const [desc, setNotepadDesc] = useState('');
  const handleModalNotepadClose = () => {
    Keyboard.dismiss();
  };

  useEffect(() => {
    if (isEdit) {
      setNotepadTitle(notepad.title);
      setNotepadDesc(notepad.desc);
    }
  }, [isEdit]);

  const closeNotepadModal = () => {
    if (!isEdit) {
      setNotepadTitle('');
      setNotepadDesc('');
    }
    onClose();
  };

  const handleOnChangeNotepadText = (text, valueFor) => {
    if (valueFor === 'title') setNotepadTitle(text);
    if (valueFor === 'desc') setNotepadDesc(text);
  };

  const handleSubmission = () => {
    if (!title.trim() && !desc.trim()) return onClose();

    if (isEdit) {
      onSubmit(title, desc, Date.now());
    } else {
      onSubmit(title, desc);
      setNotepadTitle('');
      setNotepadDesc('');
    }
    onClose();
  };

  return (
    <>
      <StatusBar hidden />
      <Modal visible={visible} animationType='slide'>
        <View style={styles.container}>
          <TextInput
```

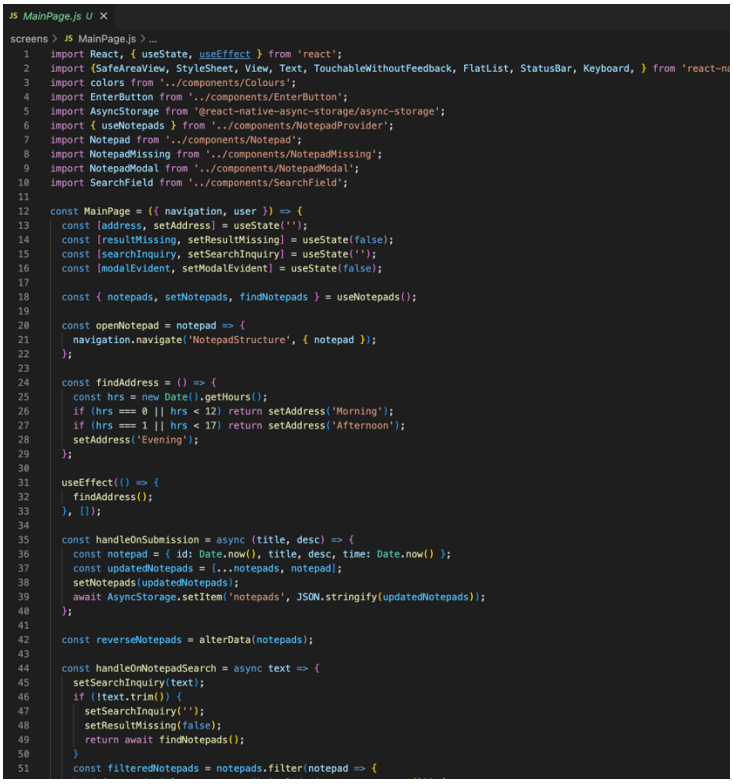
This will be the code responsible of the note title and description.

App.js

```
import AsyncStorage from '@react-native-async-storage/async-storage';
import React, {useState, useEffect} from 'react';
import { StyleSheet } from 'react-native';
import { createStackNavigator } from '@react-navigation/stack';
import { NavigationContainer } from '@react-navigation/native';
import { SafeAreaProvider } from 'react-native-safe-area-context';


import WelcomePage from './screens/WelcomePage';
import NotepadStructure from './components/NotepadStructure';
import NotepadProvider from './components/NotepadProvider';
import MainPage from './screens/MainPage';

const Stack = createStackNavigator();

export default function App() {
  const [user, setUser] = useState ({});
  const [isAppFirstTimeOpen, setIsAppFirstTimeOpen] = useState(false);
  const findUser = async () => {
    const result = await AsyncStorage.getItem('user');

    if (result === null) return setIsAppFirstTimeOpen(true);

    console.log(result)
    setUser(JSON.parse(result));
    setIsAppFirstTimeOpen(false);

  };

  useEffect(() => {
    findUser();

  }, []);

  const RenderMainPage = props => <MainPage {...props} user={user} />;

  if (isAppFirstTimeOpen) return <WelcomePage onFinish={findUser} />;
  return (
    <NavigationContainer>
      <NotepadProvider>
        <Stack.Navigator
          screenOptions={{ headerTitle: '', headerTransparent: true }}
        >
          <Stack.Screen component={RenderMainPage} name='MainPage' />
          <Stack.Screen component={NotepadStructure} name='NotepadStructure' />
        </Stack.Navigator>
      </NotepadProvider>
    </NavigationContainer>
  );
}
```

This will be the code responsible for running the whole app. Stack Navigator is used to navigate between the MainPage and the NotepadStructure page.

MainPage.js



```
JS MainPage.js U ×

screens > JS MainPage.js > ...
1   import React, { useState, useEffect } from 'react';
2   import {SafeAreaView, StyleSheet, View, Text, TouchableWithoutFeedback, FlatList, StatusBar, Keyboard, } from 'react-na
3   import colors from '../components/Colours';
4   import EnterButton from '../components/EnterButton';
5   import AsyncStorage from '@react-native-async-storage/async-storage';
6   import { useNotepads } from '../components/NotepadProvider';
7   import Notepad from '../components/Notepad';
8   import NotepadMissing from '../components/NotepadMissing';
9   import NotepadModal from '../components/NotepadModal';
10  import SearchField from '../components/SearchField';
11
12  const MainPage = ({ navigation, user }) => {
13    const [address, setAddress] = useState('');
14    const [resultMissing, setResultMissing] = useState(false);
15    const [searchInquiry, setSearchInquiry] = useState('');
16    const [modalEvident, setModalEvident] = useState(false);
17
18    const { notepads, setNotepads, findNotepads } = useNotepads();
19
20    const openNotepad = notepad => {
21      navigation.navigate('NotepadStructure', { notepad });
22    };
23
24    const findAddress = () => {
25      const hrs = new Date().getHours();
26      if (hrs === 0 || hrs < 12) return setAddress('Morning');
27      if (hrs === 1 || hrs < 17) return setAddress('Afternoon');
28      setAddress('Evening');
29    };
30
31    useEffect(() => {
32      findAddress();
33    }, []);
34
35    const handleOnSubmission = async (title, desc) => {
36      const notepad = { id: Date.now(), title, desc, time: Date.now() };
37      const updatedNotepads = [...notepads, notepad];
38      setNotepads(updatedNotepads);
39      await AsyncStorage.setItem('notepads', JSON.stringify(updatedNotepads));
40    };
41
42    const reverseNotepads = alterData(notepads);
43
44    const handleOnNotepadSearch = async text => {
45      setSearchInquiry(text);
46      if (!text.trim()) {
47        setSearchInquiry('');
48        setResultMissing(false);
49        return await findNotepads();
50      }
51      const filteredNotepads = notepads.filter(notepad => {
```

This will be the code responsible in displaying all the components. Each user will be greeted by Good Morning/Afternoon/Evening based on the time they are opening the app.

SearchField.js

```
components > SearchField.js > ...
1   import React from 'react';
2   import colors from '../components/Colours';
3   import { AntDesign } from '@expo/vector-icons';
4   import { StyleSheet, View, TextInput } from 'react-native';
5
6
7   const SearchField = ({ containerStyle, onClear, onChangeText, value }) => {
8     return (
9       <View style={[styles.container, { ...containerStyle }]}>
10        <TextInput
11          value={value}
12          onChangeText={onChangeText}
13          style={styles.searchBox}
14          placeholder='Search for your notes here'
15        />
16      {value ? (
17          <AntDesign
18            name='closesquare'
19            size={20}
20            color={colors.MAIN}
21            onPress={onClear}
22            style={styles.eraseButton}
23          />
24        ) : null}
25      </View>
26    );
27  };
28
29  const styles = StyleSheet.create({
30    searchBox: {
31      fontSize: 21,
32      borderColor: colors.MAIN,
33      borderWidth: 0.7,
34      borderRadius: 35,
35      paddingLeft: 20,
36      height: 35,
37    },
38    container: {
39      paddingHorizontal: 8,
40      justifyContent: 'center',
41    },
42    eraseButton: {
43      position: 'absolute',
44      right: 20,
45    },
46  });
47
48  export default SearchField;
```

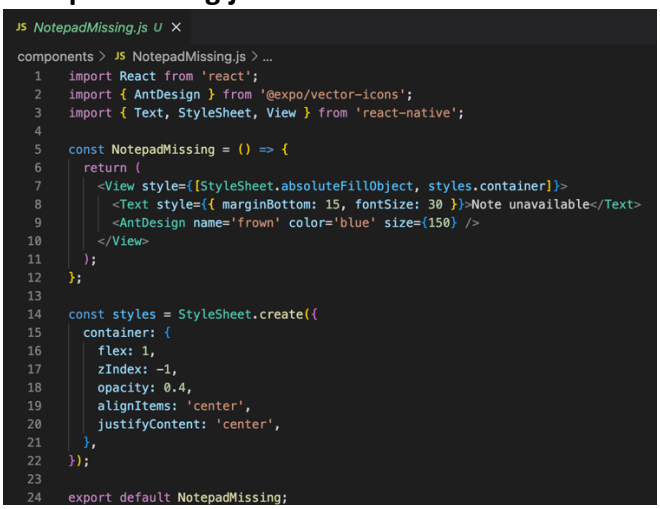
This will be the code responsible when the user searches a note (based on the note's title) on the main page, and the results will be shown accordingly. There is also a closesqure icon using AntDesign on the right-hand corner of the search bar, pressing it will erase the characters on the search bar.

Notepad.js

```
import React from 'react';
import colors from '../components/Colours';
import { Dimensions, StyleSheet, TouchableOpacity, Text, } from 'react-native';

const Notepad = ({ item, onPress }) => {
  const { title, desc } = item;
  return (
    <TouchableOpacity onPress={onPress} style={styles.container}>
      <Text style={styles.notepadTitle} numberOfLines={2}>
        {title}
      </Text>
      <Text numberOfLines={5}>{desc}</Text>
    </TouchableOpacity>
  );
};

const width = Dimensions.get('window').width - 20;

const styles = StyleSheet.create({
  container: {
    width: width / 2,
    backgroundColor: colors.MAIN,
    borderRadius: 20,
    padding: 12,
    marginLeft: 5,
    marginRight: 5,

  },
  notepadTitle: {
    fontWeight: 'bold',
    color: colors.LIGHT,
    fontSize: 20,
  },
});

export default Notepad;
```

This will be the code responsible for the overall display of the notes on the main page. The title will only last for 2 lines, whereas the description of the note will only be displayed for as long as 5 lines.

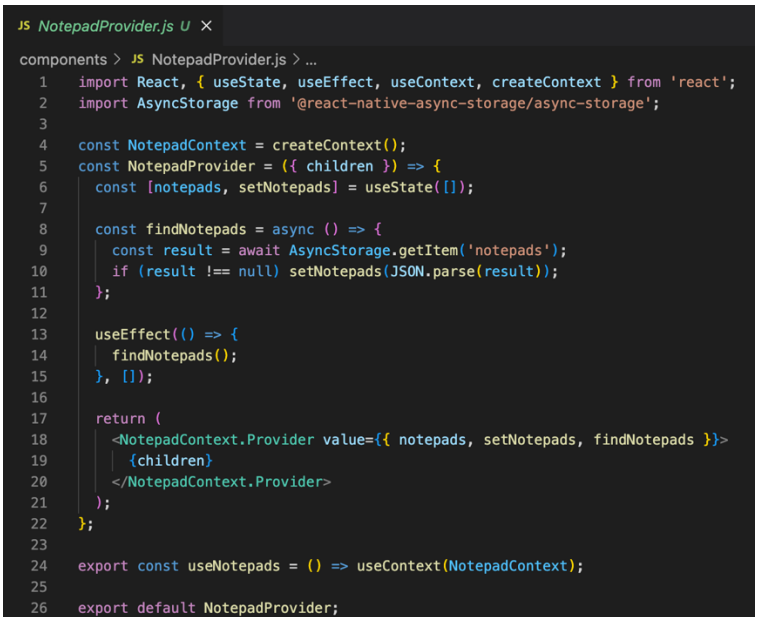
NotepadMissing.js



```
import React from 'react';
import { AntDesign } from '@expo/vector-icons';
import { Text, StyleSheet, View } from 'react-native';

const NotepadMissing = () => {
  return (
    <View style={[StyleSheet.absoluteFillObject, styles.container]}>
      <Text style={{ marginBottom: 15, fontSize: 30 }}>Note unavailable</Text>
      <AntDesign name='frown' color='blue' size={150} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    zIndex: -1,
    opacity: 0.4,
    alignItems: 'center',
    justifyContent: 'center',
  },
});

export default NotepadMissing;
```

This will be the code responsible displaying a page when the user is not able to find a particular note. A text of “Note Unavailable” along with a frown icon from AntDesign will appear.

NotepadProvider.js



This will be the code responsible for retrieving the notepads.

Colours.js

This will be the code responsible for the styling of the RizNotes app, mainly the colours aspect.

EnterButton.js

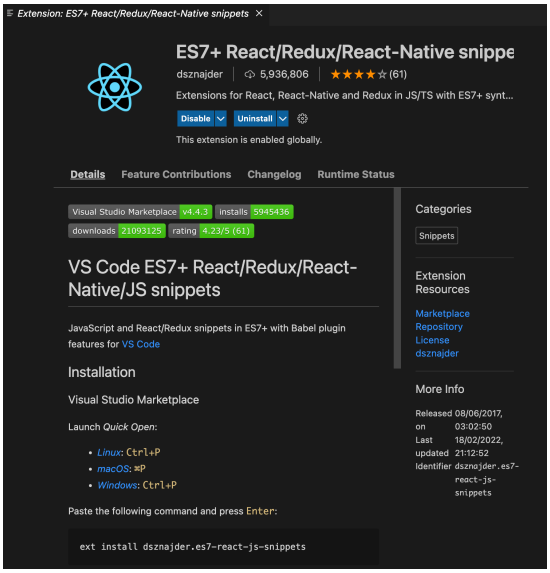
```
components > JS EnterButton.js > default
1   import React from 'react';
2   import { AntDesign } from '@expo/vector-icons';
3   import { View, StyleSheet } from 'react-native';
4   import colors from '../components/Colours';
5
6   const EnterButton = ({ antIconName, onPress, color, size, style  }) => {
7     return (
8       <AntDesign
9         name={antIconName}
10        onPress={onPress}
11        color={color || colors.LIGHT}
12        size={size || 30}
13        style={[styles.icon, { ...style }]}
14      />
15    );
16  };
17
18  const styles = StyleSheet.create({
19    icon: {
20      backgroundColor: '#FE2D00',
21      borderRadius: 20,
22      overflow: 'hidden',
23      padding: 10,
24      elevation: 5,
25      shadowColor: '#1F1F1F',
26    },
27  });
28
29  export default EnterButton;
```

This will be the code responsible for all the icons being used on RizNotes app. I am currently using Expo Vector Icons, mainly the AntDesign. All the icons can be found at (https://icons.expo.fyi/)

To import the the AntDesign icons, do input:

```
import { AntDesign } from '@expo/vector-icons';
```

Make sure to install the ES7 React/Redux/React-Native snippets by dszajder on Visual Studio Code as well.



app.json

This will be the code responsible to display the splash screen at the start. This can be seen here “image”: “./assets/rizqi_notedpad.png”. ResizeMode will be replaced to “cover” so the splash screen will take up the entire screen estate.

Unit Testing

Jest Snapshot Testing

Snapshot is the point in time representation or structure of a React Component. It is analogous to a screenshot testing that is done for a lot of mobile apps.

Firstly, we need to install Jest using npm:

```
npm install --save-dev jest
```

```
rizfebriansyah@Febriansyahs-MacBook-Pro rizqinotepad % npm install --save-dev jest

added 443 packages, and audited 1621 packages in 10s

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

found 0 vulnerabilities
rizfebriansyah@Febriansyahs-MacBook-Pro rizqinotepad %
```

Heuristics Evaluation

We will be discussing the usability heuristics for user interface design. These are actually the ten fundamental rules of interaction design by Jakob Nielsen.

Visibility of System Status

If the user wants to delete any note, the delete function can be easily identified by pressing the trash icon. Likewise for the edit function as well. The latest edited notes will always appear on the top of the main page, and the time and date in which the note was last being updated on can be seen on the right-hand corner of the note.

Freedom of User Control

While using a certain function, the user is permitted to press back or close it. For instance, if the user is searching for a note, there is a small cross on the right of the search bar, meaning that if the user were to press that button, the user would want to cancel his action of finding for notes. Users can also navigate back to the main page by clicking the Back button on the top left-hand corner when they are viewing a note.

Match Between System and the Real World

The application has UI components that resemble examples from the real world. There are quite a few examples in the application, such as adding a new note is represented by a plus button as a plus usually symbolise that a user wants to add something new and deleting a note is represented by a trash button as usually trash in the real world is something you would want to dispose and get rid of, meaning you would like to delete a note.

Error Prevention

In order to give the user a less error-prone experience, the application has error prevention. For instance, when the user is about to delete a message, a pop-up alert will be shown, reminding the user whether they would really want to delete the note as the action is irreversible. Clicking yes will confirm that the user would want to permanently delete the note.

Recognition Rather Than Recall

The RizNotes application is simple enough that it should automatically recognise how to be used, eliminating the need for the user to recall how to do so. The information on the main page of the screen shows how many notes are there, how to add a new note, how to search for a particular note etc. It does not have a lot of information that overwhelms the user. For instance, clicking on a note will reveal a more detailed description of it, where the user is given the option to just view, edit or delete the note. The necessary information is immediately available to the user, so they do not even need to remember it or worry about being overloaded or overwhelmed, everything is just a press of a button away.

Aesthetic and Minimalist Design

The main aim of a minimalist design is to reduce extraneous noise in the interface so that essential elements can be highlighted. The function icons up until now have been plain and

easy to use. Also, it is to remove any components that do not aid user tasks. As a result, the user experiences less information overload. Most functions do not have that much “noise” in the app. Even then, there are descriptions for each function to help users with it, such as the search bar where it says, “Search for your notes here”.

Standards and Consistency

For the user's convenience, the applications have multiple layers of consistency. For instance, the top search bar is consistent and similar to most mobile applications. The application uses a uniform majority of fonts, with the exception of bold text occasionally used to denote importance or the note title. To maintain consistency throughout application use, fonts will not change and remain the same.

Accelerators to increase Flexibility and Efficiency

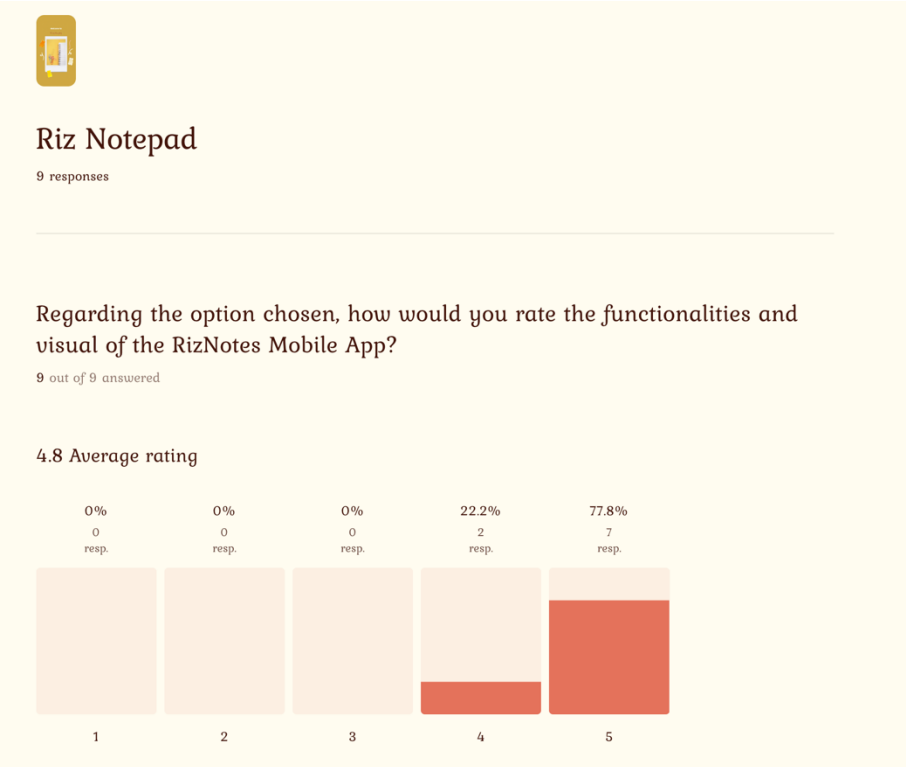
There are some shortcuts available to RizNote app users. When there are too many notes being displayed on the main page, it may be a hassle to filter through them one by one. User can just simply key in a key word on the search bar and the desired notes will be displayed accordingly. The copy and paste function as well as it is a mobile application.

Validation of Design

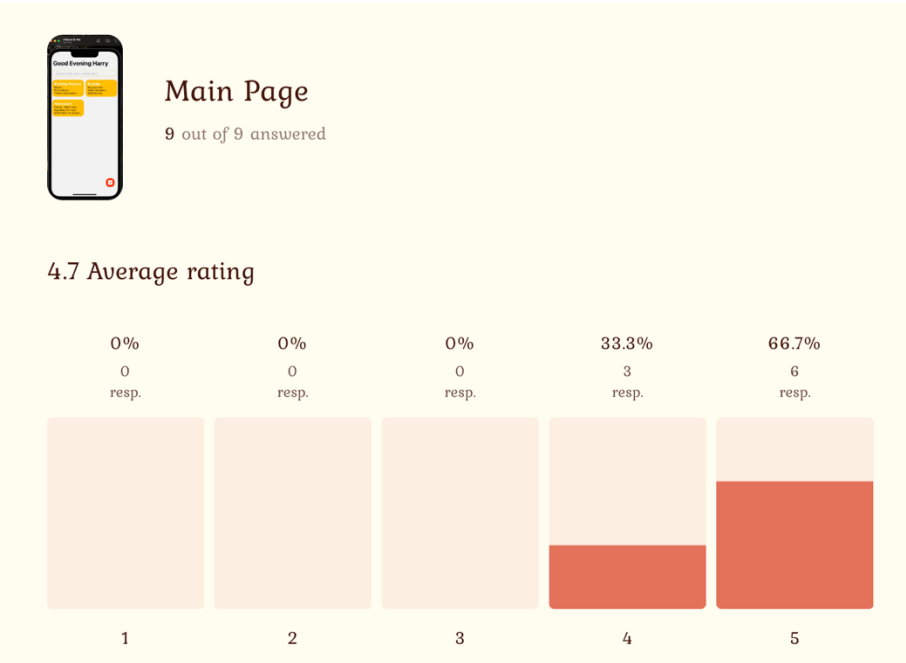
To better serve the requirements of potential users, I have created a feedback form. In the survey itself, I went on to ask how the overall user experience of the app was. Afterwards, I introduce the app's interface and functions to receive some feedback.

The result is as shown:

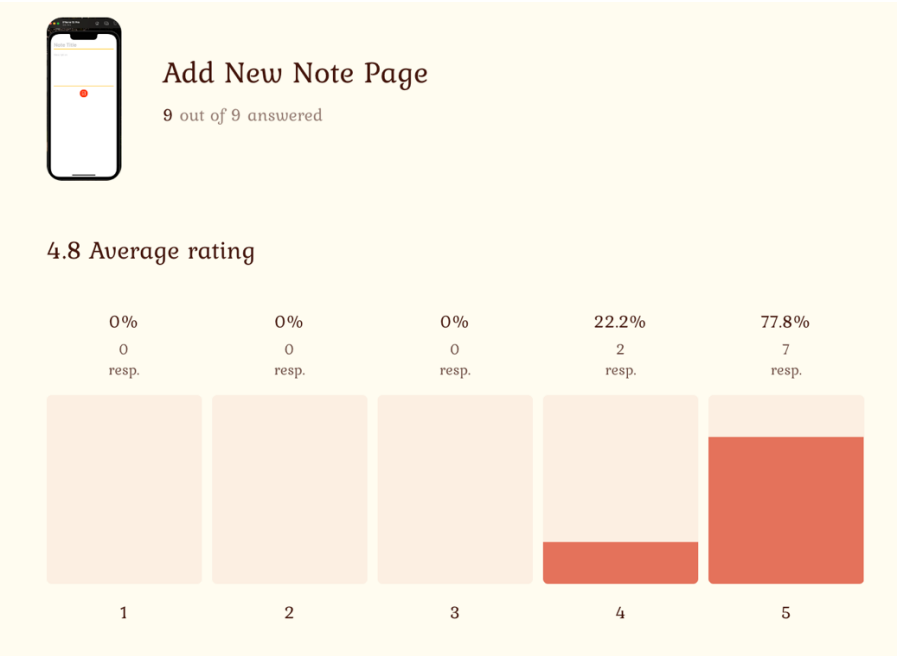
(https://rms41rhgvuu.typeform.com/report/z1x26OZY/2qSicfP4tQMT86X5)



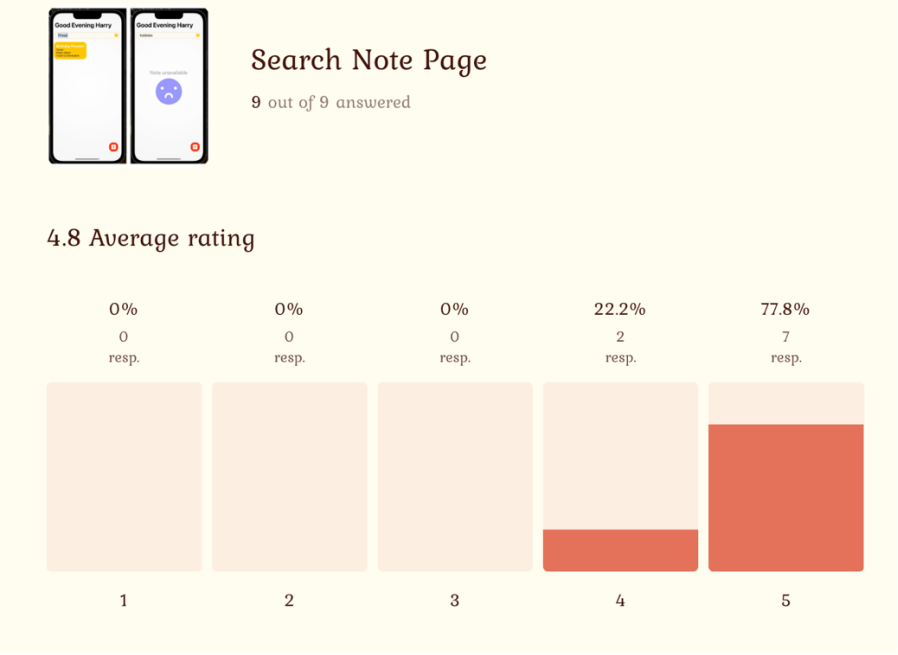
Most people find the functionalities and visuals to be good, with the most rating being 5 stars.



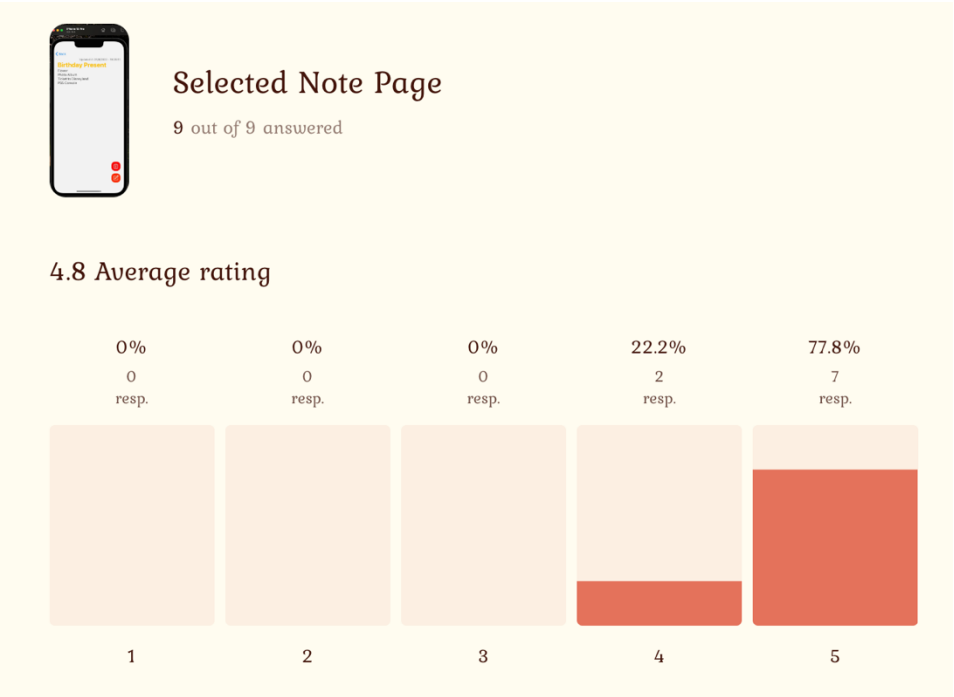
6 responses out of the total 9 gave the Main Page interface 5 stars.



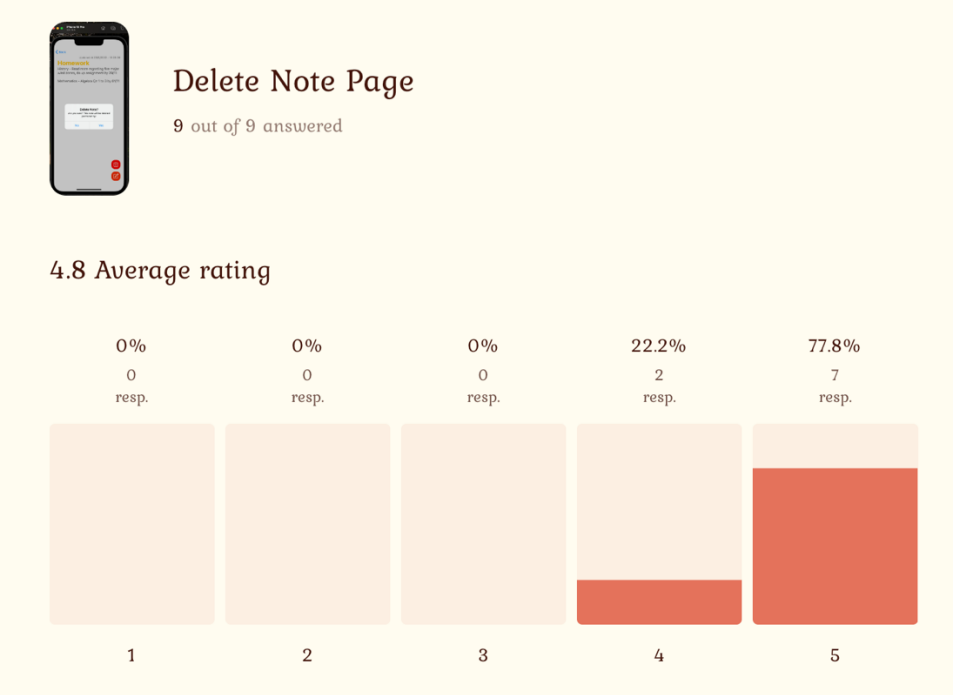
7 responses out of the total 9 gave the Add New Note Page interface 5 stars.



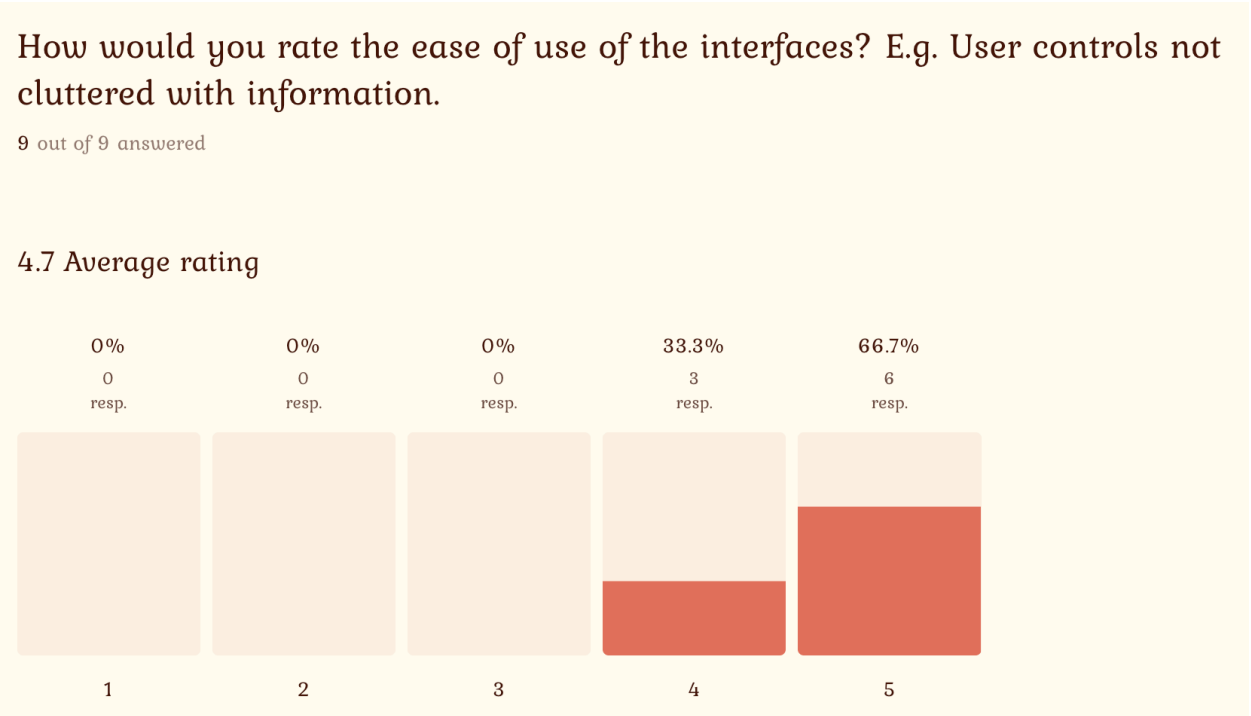
7 responses out of the total 9 gave the Search Note Page interface 5 stars.



7 responses out of the total 9 gave the Selected Note Page interface 5 stars.



7 responses out of the total 9 gave the Delete Note Page interface 5 stars.



Most people find the ease of use to be great, with the most rating being 5 stars.

Based on the 7 table charts above, we received very high ratings for all the interfaces components from our 9 responses.

In the survey form, I also inquired about any additional features or design alterations that users might want to see.
The responses are displayed as follows:

These survey findings allow us to draw the following conclusions:

- Most people are pleased with the RizNotes application. The user-friendly interface is appealing to them, and they believe it will be helpful for their own personal use.
- Regarding some improvements for my application, I feel that I could slowly implement more features such as a separate page for delete notes, including calendar to set deadlines and reminder functionalities.

Critical Evaluation

Mobile phones have become more prevalent as technology has advanced. The ability for users to jot down their thoughts and ideas or simply take notes at their fingertips is very convenient.

Goals or Product and Technical Requirements

1) Viewing notes are now more accessible than ever before
   Users are now able to view their notes wherever and whenever they are on the go. In addition to that, whenever an idea comes up, users are just able to open up the RizNotes App and jot down their thoughts immediately. Everything will be saved locally via AsyncStorage.

2) Simple and Minimalistic UI/UX
   Navigating through the app is easy. From adding a new note, searching a particular note, editing, or even deleting a note is only one press away. Everything is at the right place and the user will definitely feel at home using the app, with no issues at all.

Future Goals

1) Cross Platform and Cloud services
   I would love to improve further by allowing cross platform compatibility between mobile apps and the web. Meaning users can access their notes in whatever platform they want, be it iOS, Android, Windows, MacOS, Linux etc, and everything will be stored in a cloud. Therefore, all their notes will be readily accessible across all platforms.

2) Reminder and Calendar Function
   A calendar will be helpful as it will provide a whole overview of the notes and important deadlines which needs to be met. A reminder function based on date and time, or even location, will also be useful, as this will help users to not forget any important tasks or chores which needs to be done before the cut-off time.

3) User Authentication – Login/Logout
   This will allow users to have their own personalised notes. Having a unique account to themselves will allow users to access their own set of notes.

Snack Expo Dev Link:
https://snack.expo.dev/@febrians001/notepad-app-rizqi

YouTube Link:
https://youtu.be/-M9hnvDUME4