
Objective Evaluation

User

Objective	Evaluation
The productivity system should be able to allow the user to create, store and view journal entries	<p>This objective has been met.</p> <p>The software solution contains a page dedicated to build a list of saved journal entries for the user to view and edit existing entries. The page also features a button that allows them to create new entries. Whether new or existing entries, the program is able to save them to the local database and retrieve them for future usage.</p>
The solution should be a desktop application	<p>This objective has been met.</p> <p>The software solution has been developed to be compatible with Windows throughout the implementation stage. Despite that the programming language used has been changed to Dart, which supports cross-platform software development, across Windows, MacOS, Android and IOS, the final program is still compatible with a desktop computer. Although the current release of the solution is only compatible with Windows, it is possible in future releases to be made compatible with MacOS and Linux to support all desktop users.</p>
The system should have an easily accessible and usable calendar for the user's planning purposes	<p>This objective has not been met.</p> <p>The initial plan for the software solution was to facilitate the user's planning of daily tasks by means of organising them by dates, time, and customised task details. A feature that was planned to be developed to solve this was to combine a calendar with a Kanban/to-do task board.</p> <p>However, after the development of the task board, it has been evaluated that the task board is enough to serve this purpose, and the development of a calendar will only be extra costly and redundant, as the board already contains</p>

	<p>features that allow the user to manipulate date and time, and customise task details.</p> <p>Therefore, the objective of developing a calendar has not been met, though the approach of meeting its original purpose has been changed to lower the costs.</p>
<p>The system should have at least an optional sleep tracker, mood tracker, study/work tracker</p>	<p>This objective has partially been met.</p> <p>Throughout the software development process, the cost of development gradually became a major concern. Building the trackers that were originally planned was evaluated to be rather cost-inefficient, as they each require a specific and different type of method of data collection. Additionally, a customised calendar was intended to be developed for this, but this idea has been altered also with concerns to its costs.</p> <p>Therefore, the habit tracker type was generalised into those that can be measured with quantitative data, and all measured using the same type of data. This is equally effectively in meeting the client's needs, and more cost-effective to develop.</p>
<p>The system should be able to perform an analysis of the user's wellbeing based on the results from the trackers and give appropriate feedback for lifestyle improvements</p>	<p>This objective has not been met.</p> <p>This particular functionality of the program has been removed due to the extra costs that would be needed for its development. The intended design of this analysis function was to provide feedback to the non-quantitative data collected, such as mood, as it would be difficult to appropriately provide feedback according to the level of happiness that has been tracked.</p> <p>Furthermore, the overall aim to "provide feedback" would have required a set of guidelines or rubric that the program follows, when determining what type of feedback to give according to the data collected. However by this means, each tracker (that were planned to be built – sleep, mood, study/work) would require a specific rubric to follow. Thus, an alternate method was undertaken and that is remove this function. Considering that this program was designed to facilitate the client to be more organised, the client would not require regular analysis of their "wellbeing" anyway.</p>

Developer

Objective	Evaluation
The application should be easy to install and use	<p>This objective has been met.</p> <p>An installation guide has been made to ensure that the user is instructed through the process of installation and that it can be easily installed. Nevertheless, the application has been designed to come in a package that can easily be installed and run, which the user can access by means of unzipping a zip-file and clicking to run the program.</p> <p>The application is accompanied by a user manual and user help inside the program such as help screens and tutorials for additional support. Moreover, all navigations within the program are appropriately implemented with labels to guide the user wherever necessary. This means that the application is easy to use, even for first-time users as they are provided with a comprehensive set of user help.</p>
The application should contain a variety of debugging tools	<p>This objective has partially been met.</p> <p>For the purpose of debugging, a troubleshooting guide has been made to facilitate the client in their debugging process. The guide has been designed to inform the user about a general method of troubleshooting, which includes the process of identifying and defining problems, and the process of debugging. It also describes a variety of potential issues that the user could encounter during the execution of the program and provides solutions to these issues. This includes both program-related and system-related issues. The guide is detailed and comprehensive, which will effectively help the user to debug, though it is the only debugging tool provided. Despite this, the software solution consists of a comprehensive set of technical and user documentation including this troubleshooting guide, and an installation guide and a user manual which will all ensure that the software can be used easily and correctly.</p>
The UI should be aesthetically pleasing with all navigations easy to use	<p>This objective has been met.</p> <p>The UI of the software solution employs a colour palette with the major colour being a gentle blue. It also uses a</p>


	<p>minimalistic background image and smooth, rounded buttons and other navigations that all work together to appeal to contemporary aesthetics concerns.</p> <p>Furthermore, a range of appropriate screen elements has been used to suit the user's needs, which allows the user to easily navigate the program.</p>
A variety of data structures should be used to allow for storage of a variety types of data.	<p>This objective has been met.</p> <p>Complex data structures have been used in the development of the source code to enable the storing of data. Lists are the major structures that have been implemented to store variables within the source code, which have been effectively used for a variety of purposes. This includes the management of variables, both local and global, from lists of traditional data to non-traditional data, including widgets for UI-building.</p> <p>To save data to another database, the Flutter plugin Hive has been used, which is a key-value encrypted database that uses BoxCollections to store data to the device's local storage. This has been used to save complex object data, that take the form of a custom class, for each of the functionality of the program.</p>
The application should be ergonomically designed	<p>This objective has been met.</p> <p>The software solution presents a comprehensive GUI that abides by current design rules, with clear consideration of ergonomics issues. Specifically, the size of all screen elements and text are set to an appropriate size, ensuring that they are easy to use and read, for those with and without visual impairment. Additionally, the program is free of flashing colours or screens, and uses a gentle colour scheme and visual qualities that prevent eyestrain.</p>
The design of the application should protect user privacy by ensuring that the saved data are secured	<p>This objective has been met.</p> <p>The use of Hive in the software solution allows user data to be saved securely onto the local device, as their files are highly encrypted. Furthermore, the user will not have access to these saved data outside the program, which prevents alterations that could be made externally to the saved files. Therefore, the design of the application effectively protects the user's privacy by securely saving their data.</p>

Development Diary

MAJOR MILESTONES

8 December 2021

Progress

 Dec 8

Tasks achieved

- Completed base layout for UI
- Learnt data binding

Stumbling blocks

- Struggled with visual inheritance in xaml
- Learning how to implement a user control

Resources

- <https://docs.microsoft.com/en-us/dotnet/desktop/wpf/data/how-to-implement-property-change-notification?view=netframeworkdesktop-4.8>
- <https://stackoverflow.com/questions/9015778/no-template-property-on-grid-style-in-wpf>

Evaluation

As I first started learning WPF, I came to understand that unlike WinForms, there is no visual inheritance in WPF as the visuals and logic are very separate and have little dependence on each other in comparison to in WinForms. Upon experimenting with code in WPF, I discovered that only components built with C# code can be inherited, but xaml will be used to create all the visual components.

Instead, I learnt how to use user controls.

Learning to implement a user control allows me to create and reuse templates for a variety of purposes, including UI layout, and customised controls.

Learning data binding was essential for the implementation of user controls for my project.

My first attempt at using a user control was making a reusable template for my UI layout (see Figure 1.0), which I aimed to use for every separate page within my program.



```
20 <RowDefinition Height="60"/>
21 <RowDefinition Height="70"/>
22 </Grid.RowDefinitions>
23 <Grid.ColumnDefinitions>
24 <ColumnDefinition Width="80"/>
25 <ColumnDefinition Width="70"/>
26 <ColumnDefinition Width="80"/>
27 <ColumnDefinition Width="80"/>
28 </Grid.ColumnDefinitions>
29 <ContentControl Grid.RowSpan="100"
30 <Grid.ColumnSpan="100"
31 Content="{Binding BackgroundImage}"/>
32
33 <Border Grid.Column="1"
34 <Grid.Row="1"
35 CornerRadius="8, 0, 25, 0"
36 <ContentControl Content="{Binding Logo}"
37 HorizontalAlignment="Center"
38 VerticalAlignment="Center"/>
39
40 <Border
41 <Grid.Column="3"
42 <Grid.Row="1"
43 CornerRadius="8, 0, 25, 25"
44 <ContentControl Content="{Binding Title}"
45 HorizontalAlignment="Center"
46 VerticalAlignment="Center"/>
47
48 <Border
49 <Grid.Row="2"
50 <Grid.Column="1"
51 CornerRadius="8, 25, 25, 0"
52 Padding="8, 25, 0, 25"
53
54 <ListView.ItemsSource="{Binding MenuButtons}"
55 <ListView.ItemTemplate
56 <DataTemplate
57 <Button Content="{Binding}"
58 Click="{Binding Click}"
59 </DataTemplate>
60 </ListView.ItemTemplate>
61 </ListView>
62 <Border
63 <Grid.Column="3"
64 <Grid.Column="3"
65 CornerRadius="25"
66 <ContentControl Content="{Binding MainContent}"
67 HorizontalAlignment="Center"
68 VerticalAlignment="Center"/>
69
70 </Border>
71 </Grid>
72 </UserControl>
```

Figure 1.0

20 December 2021

Progress

Evaluation

 Dec 20 

Tasks achieved

- Created template for an appointment card

Stumbling blocks

- Displaying a list of appointment cards by using a listview containing user controls
- Stopped working on the calendar
- Could abandon calendar feature or create it by customising the existing WPF calendar control; making my own calendar does not seem feasible

Resources

- <https://help.syncfusion.com/wpf/kanban-board/getting-started?cs-save-lang=1&cs-lang=xaml>

I used a user control to create a reusable template for my “appointment card”, which is a preview of a task on my Kanban/to-do board.

In order for it to be clickable, I customised the WPF button properties to allow this “appointment card” to be a button that is able to display a lot more information, with a special design, and is able to open up the appointment details when clicked. (See Figure 1.1)

```


<UserControl.Resources>
    <Style TargetType="Button">
        <Setter Property="FocusVisualStyle" Value="{x:Null}" />
        <Setter Property="OverridesDefaultStyle" Value="true" />
        <Setter Property="Template">
            <ControlTemplate TargetType="Button">
                <Border x:Name="Button_Back" Background="White" Quality="1" Margin="5, 5, 5, 5" CornerRadius="10">
                    <Grid Margin="10, 5, 10, 10">
                        <Grid.Resources>
                            <Style TargetType="TextBlock">
                                <Setter Property="FontSize" Value="12"/>
                                <Setter Property="FontStyle" Value="Normal"/>
                            </Style>
                        </Grid.Resources>
                        <Grid.RowDefinitions>
                            <RowDefinition Height="1*" />
                            <RowDefinition Height="1*" />
                            <RowDefinition Height="1*" />
                            <RowDefinition Height="1*" />
                        </Grid.RowDefinitions>
                        <TextBlock HorizontalAlignment="Center"
                                VerticalAlignment="Bottom"
                                Text="{Binding AppointmentTitle}"
                                FontSize="12"/>
                        <Border Grid.Row="1" Background="Black"
                                <TextBlock Grid.Row="2" Text="{Binding AppointmentData}" HorizontalAlignment="Center"/>
                    </Grid>
                </Border>
            </ControlTemplate>
        </Setter>
    </Style>
</UserControl.Resources>
<UserControl.Template>
    <ControlTemplate TargetType="{x:Type UserControl}">

```

To display these, I experimented with ListViews, but eventually found ItemsControl to be the most effective.

Figure 1.1

Progress

 Dec 28

Tasks achieved

- Semi-finished UI design for the "my entries" page
- Initially attempted to display entries using a listview but now switched to an itemscontrol

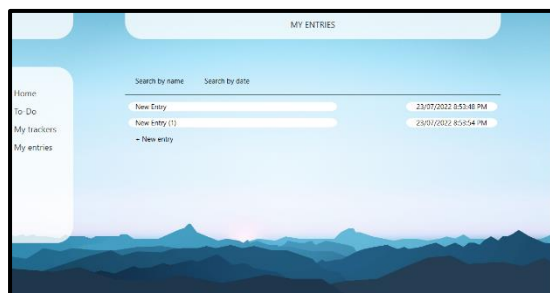
Stumbling blocks

- Making rounded buttons; need to create a control template for that

At this stage, coding the frontend has not been troubling. However, as I moved on from coding the UI to the backend development of my program, lots of learning and debugging with itemscontrol were required.

[illegible]

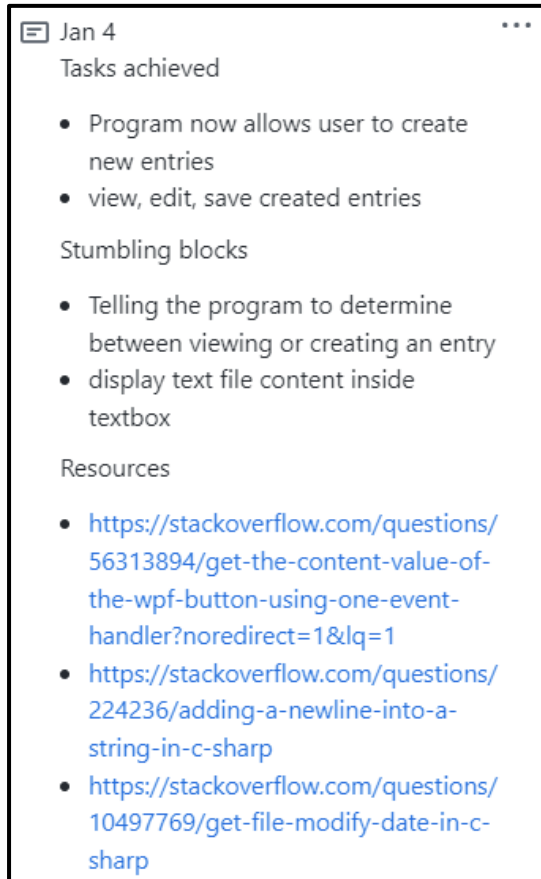
Nevertheless, the skills and knowledge obtained from this stage of the development, particularly in the areas of data binding and using `itemscontrols`, will help me in the development of the to-do board.



Furthermore, learning to create rounded buttons via a control template although have been a challenge, it will be extremely useful for the UI development of my program, as it is a major UI characteristic of my program.

Progress

Evaluation



The development of the journal-entry page has been rather successful so far, particularly with the UI development. (See Figure 1.2)

However, there have been stumbling blocks during the process of learning about data binding and the use of itemscontrol. The backend of this functionality is currently a major concern, as I need to develop more modules that prevent bugs from occurring i.e. distinguishing between an old entry page and a new one, as I've created a reusable layout for this function (See Figure 1.3), stopping the user from creating/saving entries of the same name – searching through the existing list of entries to prevent duplicate entry names before saving.

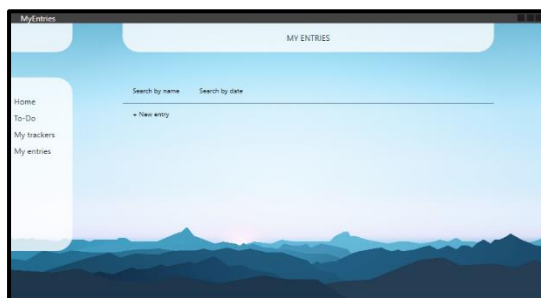


Figure 1.2

At this stage, the program has been developed to allow the user to create a new entry via the “+ New entry” button, which will open up a new page where they will be able to input more entry details in the textbox provided.

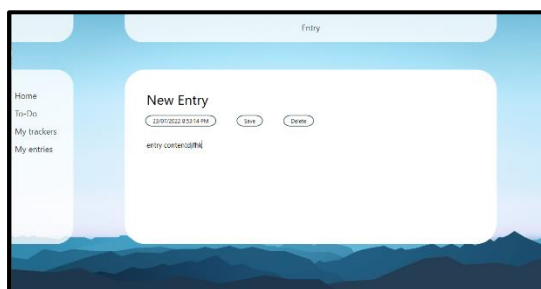



Figure 1.3

A stumbling block was loading the saved entry details inside this textbox when the user chooses to view a saved entry. However, this issue has been resolved by formatting the loading of data using a string array via StreamReader.

20-24 January 2022

Progress

Evaluation

 Jan 20

Tasks achieved

- Debugged saving-appointment
- User can now view appointment details by clicking into a preview card

Stumbling blocks

- Binding a button click event onto the appointment-preview user control
- Hence took a different approach - attaching a new, separate click event handler for each loaded preview card

Next


- Implement delete-appointment function
- Marking important appointments as "important" -> colour-coding them maybe

During this stage of the project, the main focus was development of the To-Do page/Kanban board, and the “appointment” function – the individual items that the Kanban board holds.

I initially thought to create a user control for the format of the “appointment” preview card, which allows the user to click into to view the appointment details.

(See Figure 1.4)

However, this implementation method caused major issues with data binding in an ItemsControl when displaying the created-appointments on the Kanban board.

 Jan 24

Tasks achieved

- Finalised appointment's important-filter feature
- Updated some controls; added some icons to buttons
- Implemented delete-appointment function

Stumbling blocks

- Filtering "important" appointments by changing bg of user control

Next

- Take a look at customising the calendar control (again)
- Maybe abandon the habit trackers, or change the idea of using a calendar

The primary struggle during this stage was the implementation of the user control that I created for the appointment card/preview, particularly with data binding to implement the appointment-filter (depending on whether it's marked as important or not) feature.

(See Figure 1.5)

20-24 January 2022

Progress

Evaluation

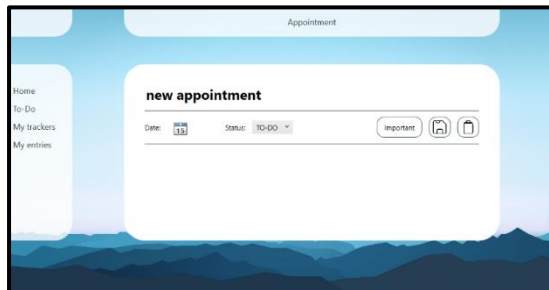


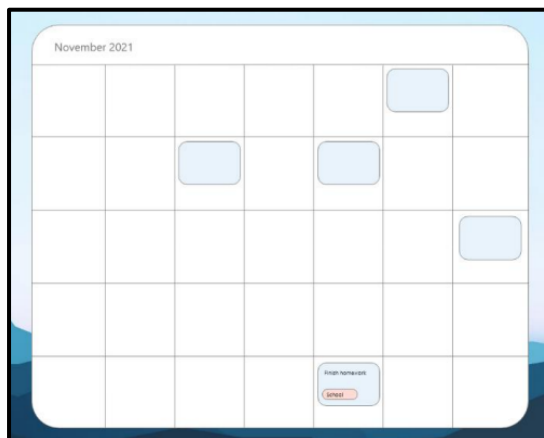
Figure 1.4

Having developed the majority parts of the appointment feature, the user will be able to view the appointment details by clicking on an appointment preview.



Figure 1.5

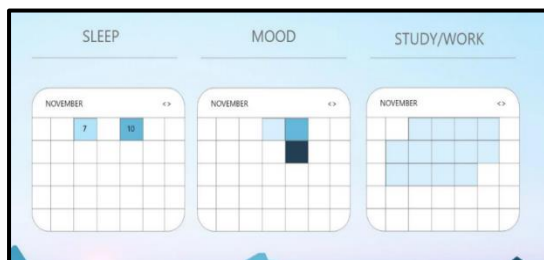
The user can now prioritise appointments. The “important” appointments have a coloured border around their preview card.



On the flip side, the calendar function has been a major concern of the project.

The initial design of the calendar (from the screen design from earlier stages of the project development) is as shown on the left.

However, this calendar design requires me to customise the WPF Calendar Control and change the majority of its attributes, which has been difficult to learn.



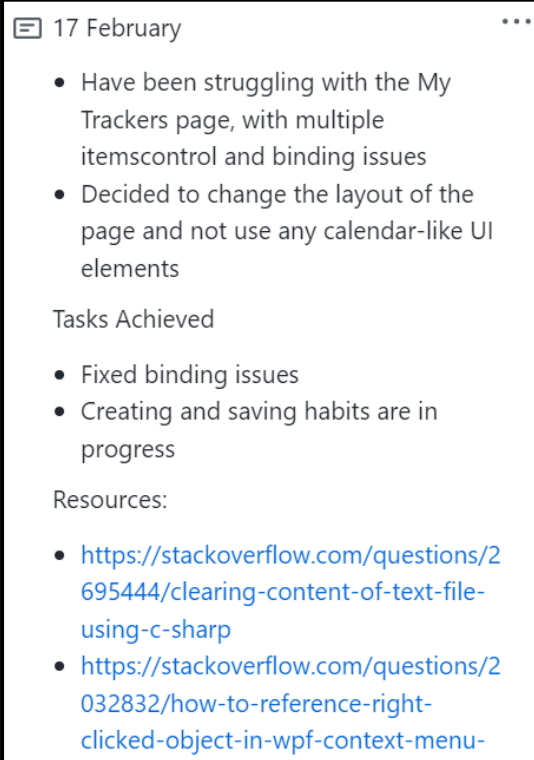
The calendar is also intended to be a major component of the habit-tracker functionality as demonstrated on the left.

Though at this stage of the development, it does not seem feasible.

[illegible]

□

Evaluation



For the habit trackers page, I have been attempting to implement an `ItemsControl` of `ItemsControl` to display the habits.

The binding errors that were encountered during the process were fixed. The source code as of this stage (See Figure 1.6) works without binding failures.

However, the client stated that the layout accompanying the current habit tracker functionality (See Figure 1.7) could be improved much more, and therefore should be changed for easier usage/navigation.

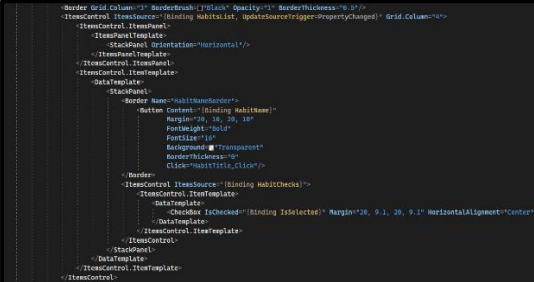


Figure 1.6

The data binding behind this functionality of the program was very costly, and in fact, very inefficient considering how cost-ineffective it was, as it involved working with a nested `ItemsControl`.

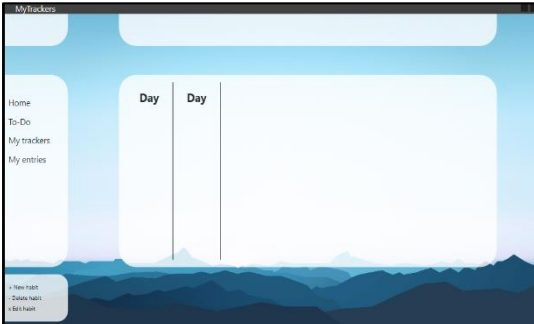


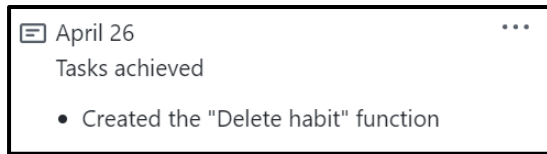
Figure 1.7

The client in particular demanded the addition of UI features such as labels to indicate the purpose of the columns and the intended functionality of this page. The current design of the page is not easily understandable and is therefore hard to use.

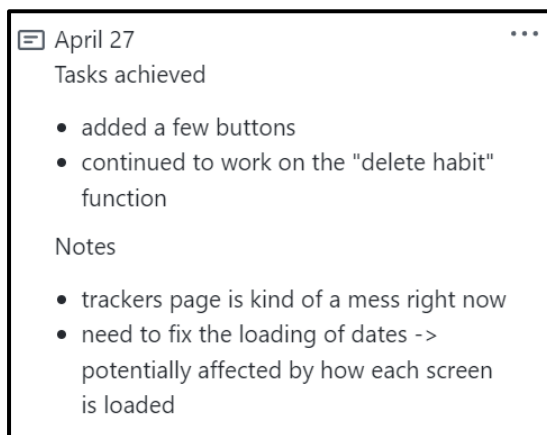
26-27 April 2022

Progress

Evaluation

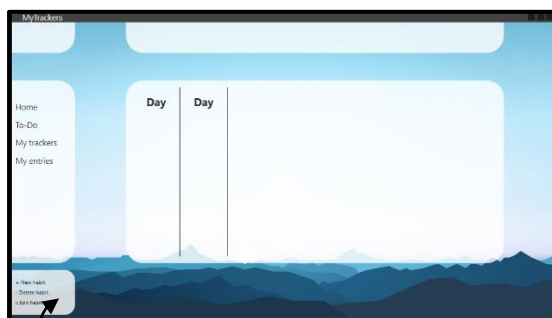


This stage of the development process involved developing the sub-features of the habit tracker functionality.



This includes the functionality of creating, editing, and deleting a habit.

This has been implemented through a panel of buttons at the bottom of the screen, which direct to a smaller pop-up screen to collect further user inputs. (See Figure 1.8)



However, the client has required to change the layout accompanying this functionality to make it easier to understand and use.

Figure 1.8

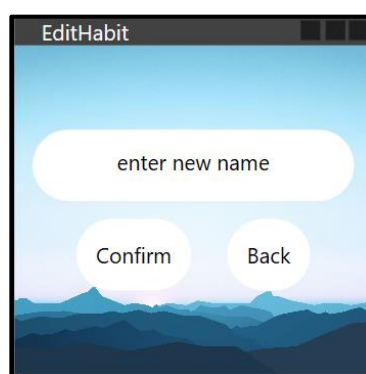


Figure 1.8

1-4 June 2022

Progress

Evaluation

☰ June 1

- Start tidying up program, make sure every feature that has been implemented so far works perfectly i.e. perfect already established-features

Notes

- binding failures on To-Do page
- deleting the top entry deletes all (fixed)
- don't use user control of "appointmentpreview", create a new customised button for this in the datatemplate for the itemscontrol?

The program has encountered more binding failures, which in the end, were fixed. Nevertheless, the entire process has been rather cost-ineffective.

Specifically, having changed the template of the appointment card from a user control to a customised button has facilitated the data binding behind the Kanban board.

```
void _command()
{
    CurrentAppointment.IsNew = false;
    CurrentAppointment.Title = name;
    CurrentAppointment.Date = date;
    CurrentAppointment.Status = GetStatusIndex(appointmenttype);
    CurrentAppointment.IsImportant = isimportant;
    CurrentAppointment.Content = File.ReadAllLines(file);

    DataService.ChangeWindow(this, new Appointment());
}
```

However, it involved learning ICommand in MVVM, and meant that the method of highlighting the "important" appointments needed to be changed. (See Figure 1.5 and Figure 1.9)

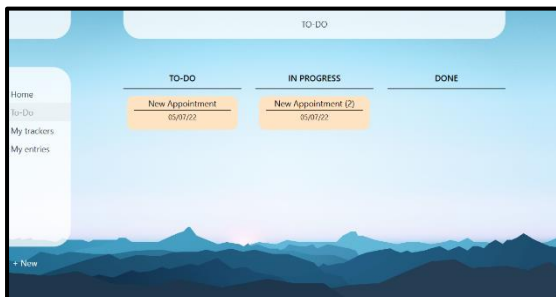


Figure 1.9

1-5 July 2022

Progress

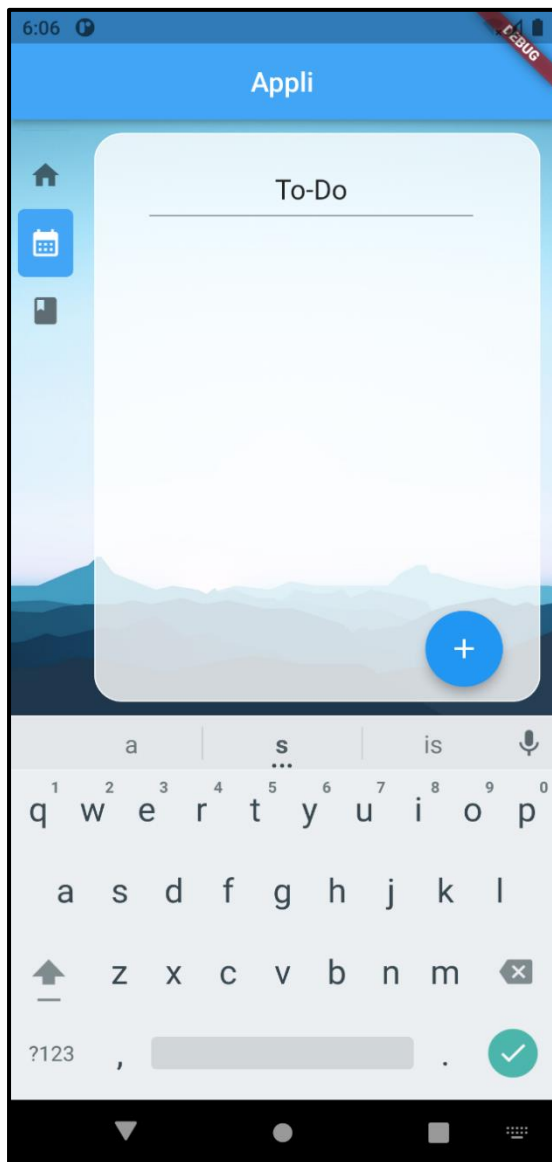
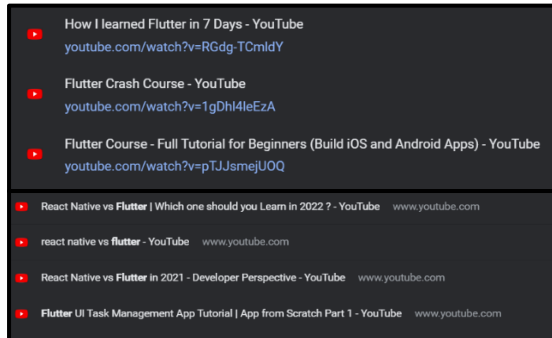


Figure 2.0

Evaluation

At this stage, developing the program in WPF seemed very cost-ineffective, with respect to the struggles with data binding and the multitude of binding errors that were needed to be fixed. Although they have been fixed, this still inspired the start of more research and a renovation of the program carried out in Flutter, which codes in Dart.

Flutter allows cross platform development, which better suits the client's needs as it means that the program would be usable on various devices, including a mobile device, which is extremely suitable for the development of a personal-organisation application.

My learning of Flutter has been predominately done through research via the Flutter docs and YouTube tutorials. The backend of the previous version of the program are planned to be transferred into the new version. However, changes to the methods are expected to be made, particularly in terms of the storing of user data.

Nevertheless, the start of this new program-development has been much more cost-effective resulting in smoother UI development. (See Figure 2.0)

1-5 July 2022

Progress

Evaluation

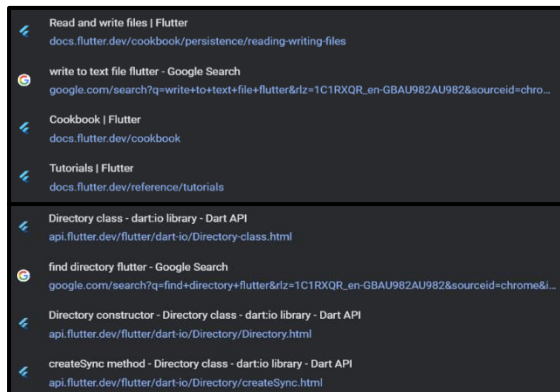
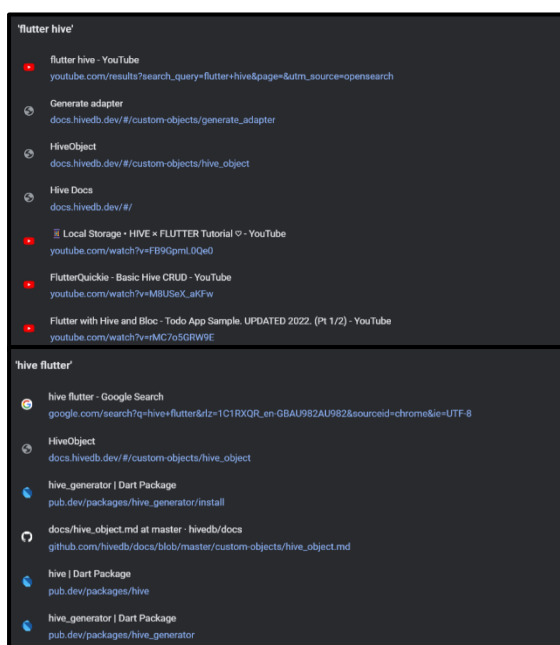


Figure 2.2



Figure 2.1



After developing the basic layout of the new UI (see Figure 2.0), I started working on the backend of the program. I began by researching and developing the method of storing and retrieving data from the local storage in Flutter (see Figure 2.1 Figure 2.2).

Similar to my approach in the previous version of my program, I built my own method of storing data to the local storage – creating directories and .txt files from the source code and writing the user data to them (see Figure 2.1). However, this has been a struggle in a new language as it was completely different to using StreamWriter and StreamReader in C#, which I used to code for the previous version of the program.

The development of this method has taken quite some time and does function correctly. However, it is a lot less efficient and the saved user data could easily be altered, in comparison to a new method that I've decided to implement instead – the Flutter plugin Hive, which uses Boxes to store and encrypt data to the local database. I've done lots of research and attempts to implement this plugin, which has initially broken my program by preventing it to be built. Nevertheless, I have ultimately resolved this issue by means of debugging and transferring my code into a brand-new solution, step-by-step installing and carefully implementing this new data structure.

1-5 July 2022

Progress

Evaluation

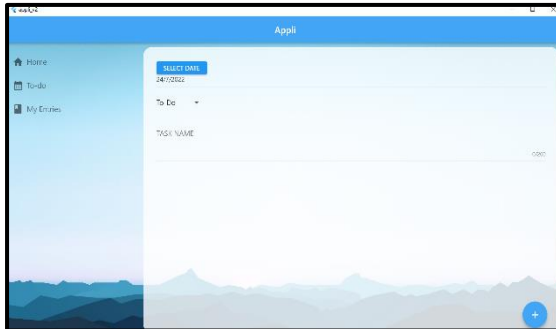


Figure 2.2

The program must allow the user to create or edit a “task” on a new page. The development of this page has been successful (see Figure 2.2). The next concern was to transfer the data from this page and ultimately load it as an item onto the task board.

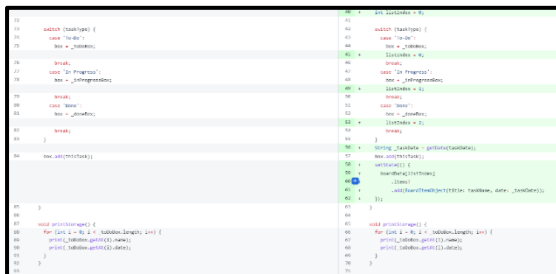


Figure 2.3

Before the development of loading items onto the task board, a greater concern was implementing a way to save the task data. Although Hive is very efficient and effective, it was still a very foreign method to me, therefore lots of testing and debugging had to be done.

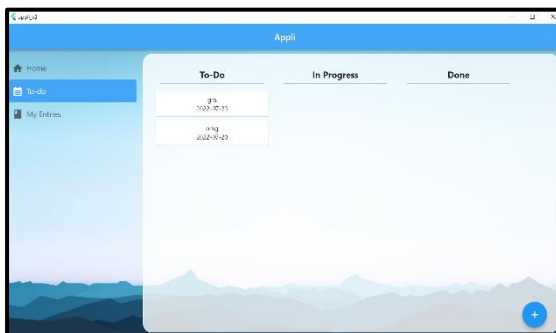


Figure 2.4

The printStorage method (see Figure 2.3) provides debugging output statements to check if the data has been saved correctly via Hive boxes. Ultimately Hive works very effectively and significantly benefited the development of the task board.

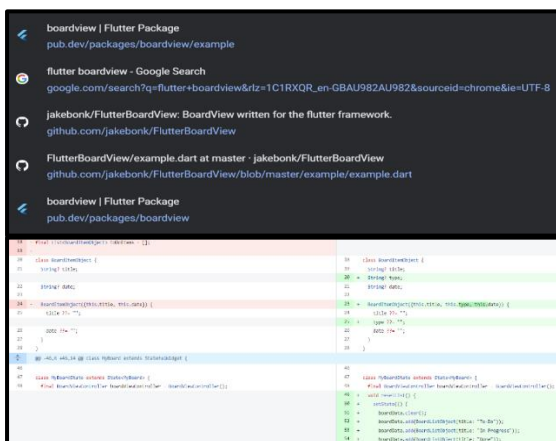


Figure 2.5

I Initially used the Flutter plugin BoardView to create the layout of a Kanban board (see Figure 2.4). However, major alterations have been made, for example removing the drag and drop feature and adding a feature where it allows the user to click on the items on the board to view and change details. Due to the method of using Hive to load the board item data, I attempted to create my own board using ListViews and other Flutter widgets (see Figure 2.5).

1-5 July 2022

Progress

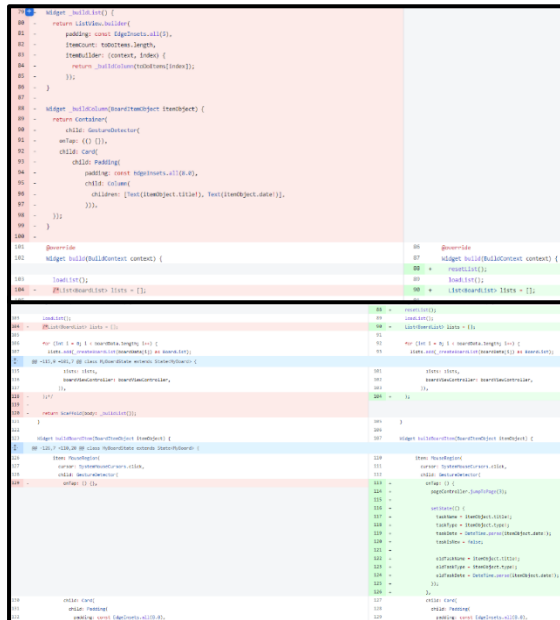
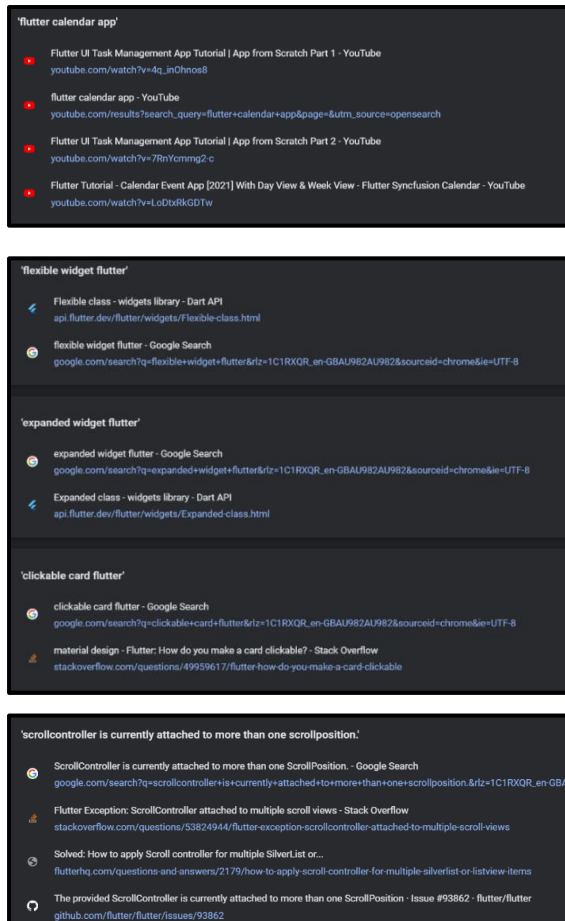


Figure 2.6

Evaluation

However, this method has been ineffective and has therefore been changed, as seen in the alterations made in source code (see Figure 2.6).

Ultimately, the BoardView plugin is still the most efficient method with some changes made to its features, such as disabling drag and drop, and changing the style of its board design.



Research has also been done to prepare for the development of the calendar functionality of the software solution. However, the development team evaluated that the task board already contain the features to meet the purpose of an organisation calendar and the needs of the client, thus the development of this new functionality would not be cost-effective.

As a result, after finalising the task board, next to be implemented was the journal entry page. However there were major issue encountered during the development of both the to-do board and the journal entry list page, as they both required the use of listviews or gridviews, which are both scrollable widgets, that are to be nested inside the pageview. A pageview is also scrollable widget and a scrollcontrol cannot be used on different widgets.

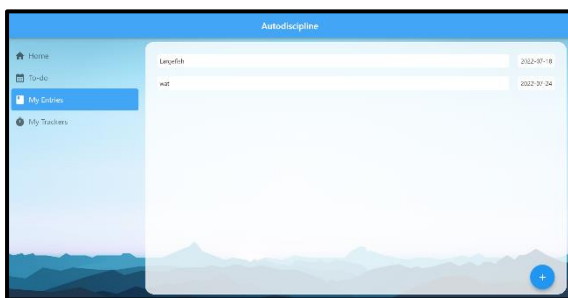
7-10 July 2022

Progress

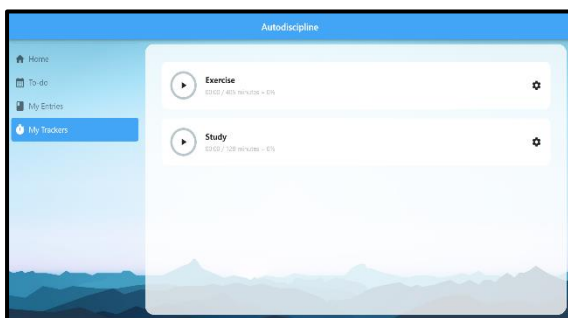
Evaluation



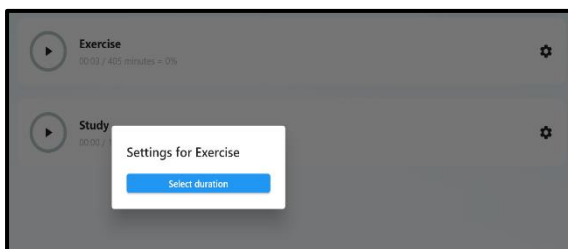
Unlike the build method of the task board which uses the BoardView plugin, a listview of rows are used to build the list of journal entries instead.



The previous issue with the scrollcontroller has been fixed by instantiating and attaching the listview controller to a separate scrollcontroller to the pageview's, which now enables the scrolling of this page.



The development of the habit tracker page (fundamentally inspired by Mitch Koko's) has utilised a similar build layout to that of the journal entry page, which has been successful. A listview has been used in a similar approach with its item type being a custom HabitTile that I created, with each habit being able to be tracked by a timer in minutes.



The main struggle during development has been with the function of setting the goal duration for each habit (see Figure 2.7), though it has been resolved (see Figure 2.8).

Figure 2.7

13-15 July 2022

Progress

Evaluation

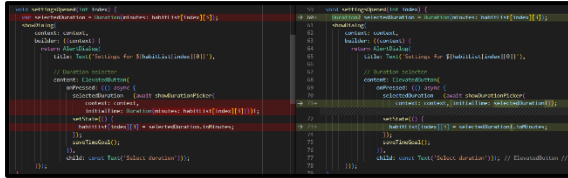


Figure 2.8

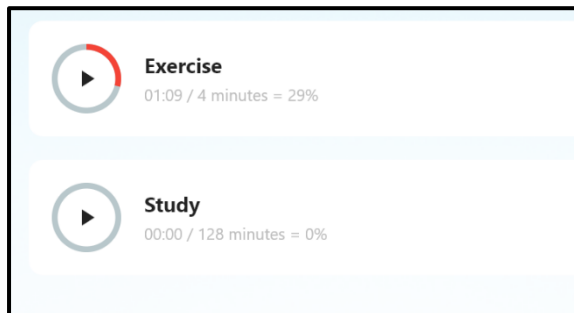


Figure 2.9

This approach is very cost effective as it tracks each habit in the same manner i.e. using the same measurement of quantitative data (see Figure 2.9), as opposed to the ones that were initially planned which would've been very difficult to analyse.

Therefore the initial objective for this habit-tracker functionality has been changed due to concerns with costs. The trackers that were planned to be developed as the default trackers have also been slightly changed. The user will be provided with two default trackers, one for exercise and one for study, as opposed to the three that were initially planned due to the difficulties in tracking non-quantitative data. Nevertheless, this method of habit meets the purpose of the software solution and the needs of the client.

Testing

MODULE TESTING

Test Report

Item being tested: Autodiscipline program to Save 'To-Do' Tasks

Date: 3 July 2022

Tester: Sarah Cheng

Type of test: End-user, alpha module test

Appendix 1: saveTask() module tested

Appendix 2: saveTask() test data used and expected vs actual results

Interpretation of Test

The return type `nameIsValid()` has been developed (see Appendix 1) to be a stub that temporarily only returns (Boolean) true to test the functionality of the `saveTask()` module. The name validating Boolean return type will be developed later to prevent the user from saving multiple entries of the same name.

The following errors were countered during this test:

1. Retrieving from the box storage requires the parameter input to be the key that has been assigned to the saved item when it saved the first time, therefore the `box.get()` method (see Appendix 2) returns null when attempting to retrieve it using an instance of `MyTask`.
2. Although the Hive documentation have stated that `.save` could be used to update the saved data of a custom object, this method did not work and threw an exception.

MODULE TESTING

Test Report

Item being tested: Autodiscipline program to Customise Habit Tracker Timer

Date: 4 July 2022

Tester: Sarah Cheng

Type of test: End-user, alpha module test

Appendix 3: saveTimeGoal() and loadTimeGoal() modules tested

Appendix 4: debugging outputs produced from loadTimeGoal()

Appendix 5: saveTimeGoal() and loadTimeGoal() test data used and expected vs actual results

Interpretation of Test

Two debugging output statements were used in this test (see Appendix 3 and Appendix 4) to output each of the DurationPicker's data into their own storage positions – box[0] and box[1], one for each habit tracker. Due to the design of the DurationPicker, it had been coded to force the user to input data in the correct format (positive int), thus only 0 and positive integer values could be tested. The module was expected to save each habit's duration set from its DurationPicker to its own box position, and upon loading these values into the habitList list on build, print them to check if the data had been stored correctly.

The test did not detect any error messages and from the expected outputs vs the actual outputs in this test (see Appendix 5), it is evident that the modules had no errors. Test data were processed correctly to the intended purpose and achieved the expected results.

PROGRAM TESTING

Test Report

Item being tested: Autodiscipline program

Date: 14 July 2022

Tester: Autodiscipline users

Type of test: Beta black box test

Appendix 6: Beta test questionnaire – installation feedback

Appendix 7: Beta test questionnaire – UI and navigation feedback

Appendix 8: Beta test questionnaire – task board feedback

Appendix 9: Beta test questionnaire – journal entry system feedback

Interpretation of Test

From the questionnaire results gathered from 12 testers of the Autodiscipline beta application, the program functions correctly and performs well. However, there was a compatibility issue encountered by a few of the testers.

This desktop application could only be run on Windows, out of the 12 program testers, the 2 users who used Mac to install the beta program package found that the application was not “easy to install” (see Appendix 6), and in fact, could not be installed nor used. Future releases of the program could be made compatible with MacOS and mobile OS as a solution to this issue, which is feasible as the program has been developed in Flutter which supports cross-platform development.

Most users responded that the GUI design was aesthetically pleasing (see Appendix 7), and all navigations were easy to use. However, those who could not install and run the application on their Mac devices neither agreed nor disagreed with this statement – responded with “maybe”, which is precise as they could not access the program.

As a result of this compatibility issue, a minority – who could not run the application reported that the to-do board and the journal entry system were not easy to use (see Appendix 8 and Appendix 9). Despite this, the majority of the users evaluated that these two features of the program were well implemented and provided a score of 4 or 5 out of 5. In the concluding comment section of the questionnaire, the users were asked to outline any bugs that they may have encountered during the process of using the application, it was reported that no bugs have been found amongst the 12 tests that were undertaken.

SYSTEM TESTING

Test Report

Item being tested: Autodiscipline program

Date: 20 July 2022

Tester: Sarah Cheng

Type of test: Acceptance test

Appendix 10: Acceptance test results

Interpretation of Test

Program was tested in its intended environment – on a Windows desktop device, testing all functionality, features, and navigation of the application.

The software solution has passed its entry criteria as it has met the objectives and intended purpose to suit the client's needs for organisation. Most of the features that were planned to be developed in the objectives have been implemented, with some alterations made yet still suits the original requirements. Its usability and performance level are good, with high functional correctness and completeness, although the habit tracker system could be improved by allowing the user to create more trackers and customise them in more ways.

Compatibility (relating to installation) and upgradability could be enhanced in future releases of new updates and maintenance, as the current release is only compatible with Windows.

The program has high data integrity and accurate data conversion, as it did not result in defects with all the data used. Stored data could not be easily accessed, with ensures confidentiality and privacy of user data. The availability of these data was also ensured as they could be readily accessed via the program for their intended purposes i.e. viewing, editing and updating saved details.

Appendix

Appendix 1: saveTask() module tested

```
void updateTask() {  
    var box = getBox();  
    var thisTask = MyTask(name: taskName, type: taskType, date: taskDate);  
    var oldTask = box  
        .get(MyTask(name: oldTaskName, type: oldTaskType, date: oldTaskDate));  
    bool canSave = nameIsValid();  
  
    if (canSave) {  
        var task = box.get(oldTask);  
        task = thisTask;  
        task.save;  
        print(box.get(thisTask));  
  
        const notif =  
            Snackbar(content: Text('Saved - restart the app to view changes'));  
        ScaffoldMessenger.of(context).showSnackBar(notif);  
        pageController.jumpToPage(1);  
    }  
}
```

Appendix 2: saveTask() test data used and expected vs actual results

thisTask	oldTask	Expected task	Actual task	Expected output	Actual output
MyTask("task1", "To-Do", "20-07-2022")	MyTask("task1", "To-Do", "15-07-2022")	MyTask("task1", "To-Do", "20-07-2022")	null	MyTask("task1", "To-Do", "20-07-2022")	null
MyTask("hellotask", "To-Do", "20-07-2022")	MyTask("task1", "To-Do", "15-07-2022")	MyTask("hellotask", "To-Do", "20-07-2022")	null	MyTask("hellotask", "To-Do", "20-07-2022")	null
MyTask("task1", "To-Do", "15-07-2022")	MyTask("task1", "To-Do", "15-07-2022")	MyTask("task1", "To-Do", "15-07-2022")	null	MyTask("task1", "To-Do", "15-07-2022")	null

Appendix 3: saveTimeGoal() and loadTimeGoal() modules tested

```
void saveTimeGoal() async {
  var box = getBox();
  //box.clear();

  // exercise tracker
  await box.putAt(0, habitList[0][3]);
  // study tracker
  await box.putAt(1, habitList[1][3]);
}
```

```
void loadTimeGoal() {
  var box = getBox();

  try {
    print(box.getAt(0));
    print(box.getAt(1));
    habitList[0][3] = box.getAt(0);
    habitList[1][3] = box.getAt(1);
  } catch (e) {}
}
```

Appendix 4: debugging outputs produced from loadTimeGoal()

```
flutter: 96
flutter: 49
Reloaded 1 of 964 libraries in 327ms.
flutter: 96
flutter: 107
flutter: 82
flutter: 107
```

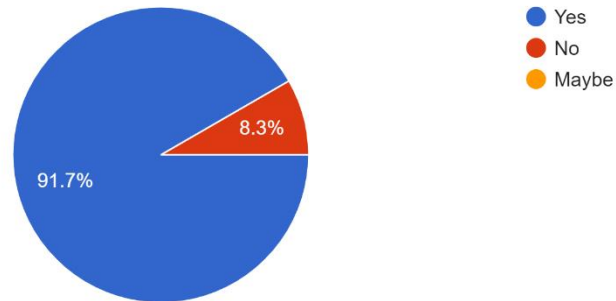
Appendix 5: saveTimeGoal() and loadTimeGoal() test data used and expected vs actual results

DurationPicker (min) for Exercise	DurationPicker (min) for Study	habitList[0][3]	habitList[1][3]	box[0]	box[1]	Expected output	Actual output
96		96	49	96	49	96	96
						49	49
	107	96	107	96	107	96	96
						107	107
82		82	107	82	107	82	82
						107	107
	169	82	169	82	169	82	82
						169	169
230		230	169	230	169	230	230
						169	169
	89	230	89	230	89	230	230
						89	89
	56	230	56	230	56	230	230
						56	56
317		317	56	317	56	317	317
						56	56
	0	317	0	317	0	317	317
						0	0
282		282	0	282	0	282	282
						0	0

Appendix 6: Beta test questionnaire – installation feedback

The application was easy to install

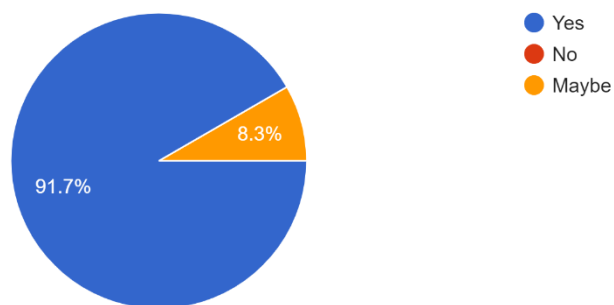
12 responses



Appendix 7: Beta test questionnaire – UI and navigation feedback

The GUI design is aesthetically pleasing and all navigations are easy to use

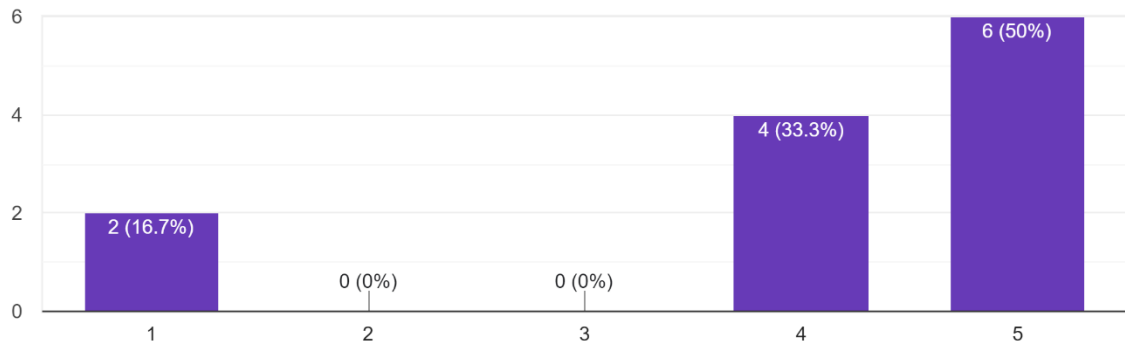
12 responses



Appendix 8: Beta test questionnaire – task board feedback

The to-do board is easy to use and works well

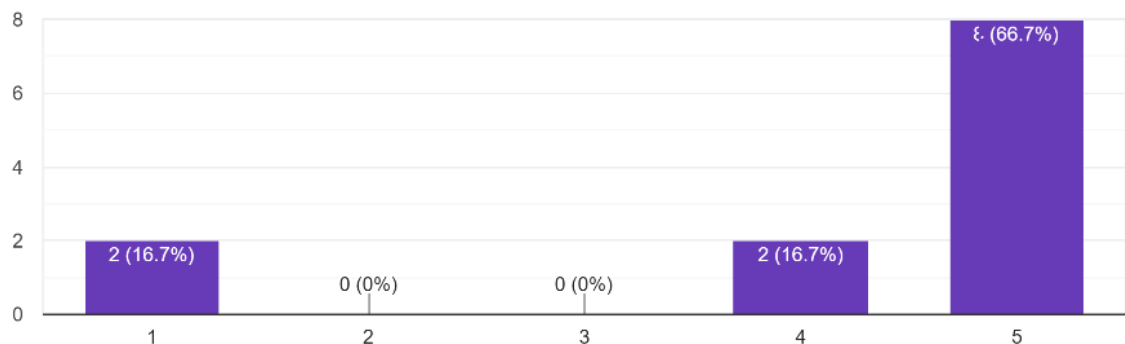
12 responses



Appendix 9: Beta test questionnaire – journal entry system feedback

The journal-entry system allows me to easily create, store and view entries

12 responses



Appendix 10: Acceptance test results

Total # of tests	# of tests passed	# of tests failed	# of tests not executed	pass %	# of defects logged	# of valid defects
9	8	0	1	88.89%	3	2