Childcare Management Application

Student Name: Owen Atkinson
STUDENT NUMBER: B00744277
COURSE TITLE: COMPUTING SYSTEMS (PT)
PROJECT SUPERVISOR: RICHARD DAVIES

Table of Contents

1. Introduction

1.1 Background

The role of a childcare provider requires administrative tasks in the form of daily paperwork to cover areas such as attendance, medicine administration, accident reports, etc. This needs to be repeated daily for each child under care (max of 6 children per childcare provider), so the paperwork quickly piles up. Within this profession, it's typical for childcare providers to hold onto records until children turn 22 years old for insurance purposes. This results in an accumulation of huge binders filled with paperwork spanning across decades, which can be problematic in several areas. When seeking specific information, it can be difficult to sift through thousands of pages worth of logs and paper can easily go missing if not stored securely, this can lead to large chunks of historical data vanishing over time. Owen is building an application that would support the digitalisation of administrative tasks required of a childcare provider through the means of a mobile application. This would allow childcare providers to add key information and resources which they could access through their mobile device on-demand.

A digital transformation for childminding presents an accurate, paperless and simplified solution compared to the traditional maintenance of existing processes. According to the family support NI registration, there are currently over 2200 registered private childcare providers within Northern Ireland but the occupation has been left behind in terms of digitalisation (familysupportNI, 2022). The demand for a solution to address these issues continues to rise.

1.2 Project Aim

Owen's main aim of this project is to build an application to digitalise the traditional administrative and maintenance tasks required of a childcare provider. He will achieve this by promoting a paperless approach and providing services in a digital format to assist in the daily routines of childcare providers.

1.3 Project Objectives

Owen outlined nine project objectives at the beginning of his project:

1. To understand and identify the key requirements of the stakeholder.
2. To refine a list of functional and non-functional requirements.
3. To evaluate and identify the most suitable software methodology for the project.

4. To conduct a risk analysis on potential problems which may emerge during development, establishing a back-up plan or mitigating the risks.
5. To research and identify the most suitable technology options to achieve the stakeholder's requirements.
6. To construct a Gantt chart which will include details of sprint durations, key milestones and deadlines.
7. To build an application which serves as a central resource and reduces the complexities surrounding the administrative and maintenance tasks involved in the role of a childcare provider.
8. To carry out user testing on a range of individuals and record their feedback for future improvements
9. To produce a user guide that will enable the user to harness the application with as little reading as possible.

1.4 Report Structure

Chapter 2 – Literature Review & Technical Background

This includes details regarding the project's foundations and research into the existing systems within the same market.

Chapter 3 – Requirements Gathering & Analysis

This includes details regarding requirement gathering, prioritisation, validation, sprint planning and the final user requirement list.

Chapter 4 – Project Preparation & Planning

This includes details regarding project management tools, version control, risk assessment, chosen software development lifecycle methodology, Gantt chart as well as an overview of the hardware and software utilised.

Chapter 5 – Design

This includes details regarding use case diagrams, screen design phases, colour palettes, accessibility and a discussion regarding the ER Diagram.

Chapter 6 – Implementation & Testing

This includes details regarding the implementation of user stories into features within the application. This highlights the code used to achieve each user story, using unit tests to confirm that user stories are fully met and providing screenshots to provide evidence of this.

Chapter 7 – Evaluation

This includes details regarding Owen's self-evaluation as he reflects upon his overall performance during the project and validates how many user stories were completed within the duration of his project. Owen also discusses future developments of the project, innovative or creative aspects and the commercial opportunities that the application possesses.

Chapter 8 – Conclusion

This includes details regarding Owen's personal reflection on what he has learnt through completing this project and the outcomes that he is most proud of.

2. Literature Review & Technical Background

2.1 Project Foundations

Owen has connections to stakeholders within the childcare provider profession and has previously worked towards digitalising the recording of data. Though these existing means were not as advanced as the features being made available through the creation of his application.

2.2 Existing Systems

When conducting his research, Owen discovered there were several existing systems that have created solutions aimed towards childcare. Examples of this include himama, cheqdin, cradle and foundations. However, these systems, like most others, were aimed towards nursery schools or other larger business types involved in childcaring. Owen's system is unique in this matter as he is creating a system that is aimed towards supporting private childcare providers that are operating as individuals, typically from their own homes.

3. Requirements Gathering & Analysis

3.1 Requirements Gathering

To get a better understanding of some of the struggles which childcare providers encounter in day-to-day tasks, Owen began this process by meeting with his stakeholder to discuss the features and objectives that the solution should aim to achieve. We gave a particular focus to the key areas within the occupation of a childcare provider that require specific attention for improvement. Through understanding the stakeholder's needs & the nuances within the occupation, this helped us to formulate foundations for the initial requirements of the project. All the user's information from the meeting was noted and this was used to develop an initial list of user requirements. The requirements were then organised into sub-sections, depending on the area

of the solution which they impacted. They were also separated into functional & non-functional requirements to provide clear structure to the document. The requirements were then converted into user stories to align with the processes of Scrum. User stories are informal and general explanations of a software feature which is written from the perspective of the end user. It helps to articulate how a software feature will provide value for a customer (Rehkopf, n.d.).

3.1.1 User story example

"As a childcare provider, I want to ***insert requirement***. This will ***insert benefit of requirement being achieved***"

3.2 Prioritisation

Owen considered several prioritisation methods for the user stories:

1. Ranking: In this method, a unique number is assigned to each user story to reflect its importance. The result of this is a list of ascending numbers, with the most valuable user story at the top of the list and the least valuable at the bottom of the list.

2. MoSCoW: In this method, user stories are categorised into four groups with different levels of prioritisation. These groups are: Must Have, Should Have, Could Have and Will Not Have.

3. Value vs Complexity: In this method, a numerical value between 1-5 is assigned to each user story to represent the value that it represents to the system and the complexity involved in implementing it. With 1 being very simple, 5 being very complex, 1 being not valuable and 5 being very valuable. These two values are then multiplied together to give an overall total for the user story. This gives a list where the user story with the highest total should be very useful for the stakeholder and should be very simple to develop. As you get further down the list, there will be user stories deemed less useful and more complex to develop.

Owen decided to use a combination of the MoSCoW and Value vs Complexity methods. This allowed him to not only have a numerical value to allocate to each user story, but also to use these figures to sort the user stories into the four MoSCoW categories. Owen spoke with the stakeholder to understand the importance of each user story and the value that it brought to the stakeholder. This number was multiplied by the complexity of implementing the user story, which Owen could provide insights on after conducting research into the relevant development tasks. After calculating the overall figure for the user story, it was grouped into a MoSCoW category using the following rule:

- 20+ points - Must Have
- 15-20 points - Should Have
- 10-15 points - Could Have
- 1-10 points - Will Not Have

Please see **Appendix D** for the full list of requirements, this includes the individual value and complexity scores.

3.3 Requirement Validation

After completing his prioritisation methods, Owen confirmed the priorities with his stakeholder to give them an opportunity to correct any of these. The stakeholder was satisfied with the arranged priority and happy for features to be delivered in the specified sequence.

3.4 Sprint Planning

After receiving approval from his stakeholder, Owen began to plan the content that he would be developing for each sprint. Owen allocated user stories to sprints based on the complexity of the task that he had previously scored. This scale was previously based on 1 (very complex) to 5 (very simple), but for gaining an insight into the difficulty of each task, Owen flipped the scales. He created a table and plotted the complexity score of each task. Then, he created an additional column for difficulty from 1 (very easy) to 5 (very difficult). This allowed Owen to understand how many tasks that he could allocate to each sprint, ensuring that he wasn't allocating too little or too much.

If there were a lot of easier tasks, Owen could allocate a higher number of them to a sprint. But if there were more difficult tasks involved, there would be a lower quantity of user stories included in a sprint. Owen aimed to allocate no more than 15 difficulty points worth of tasks per sprint. By maintaining this balance it would ensure that he could make consistent progress throughout his sprints without overloading himself with tasks. After he knew the contents of each sprint, this allowed Owen to begin planning the start and end dates for each sprint. He also scheduled dates for routine show and tell sessions with his stakeholder. This gave him an opportunity to demonstrate the current state of the application and allow the stakeholder to provide feedback on existing features or other features which they felt would be valuable to include.

User Stories	Complexity	Difficulty
Sprint 1		
1.1 As a childcare provider, I want to log visitor information.	**5**	**1**
1.2 As a childcare provider, I want to log medicine administration about each child.	**5**	**1**
1.3 As a childcare provider, I want to log daily attendance information about each child.	**5**	**1**
1.4 As a childcare provider, I want to log accident reports if an incident occurs.	**5**	**1**
3.1 As a childcare provider, I want to record dietary requirements about each child.	**5**	**1**
3.2 As a childcare provider, I want to record medical information about each child.	**5**	**1**
3.3 As a childcare provider, I want to record emergency contact information about each child.	**5**	**1**
3.5 As a childcare provider, I want to have the ability to add additional children to the application.	**5**	**1**
Total		**8**
Sprint 2		
1.6 As a childcare provider, I want to have the ability to edit the information that I have logged.	**4**	**2**
1.7 As a childcare provider, I want to have the ability to view any of the information that I have submitted via forms.	**3**	**3**
1.9 As a childcare provider, I want to have the ability to delete the information that I have submitted via forms.	**5**	**1**
3.4 As a childcare provider, I want to have the ability to update this information about each child should their circumstances change throughout their time in my care.	**4**	**2**
3.6 As a childcare provider, I want to have the ability to remove children from my application, should they no longer be in my care.	**5**	**1**
3.7 As a childcare provider, I want to have the ability to view child details should I need to retrieve this information.	**3**	**3**
Total		**12**
Sprint 3		
1.8 As a childcare provider, I want to have the ability to view my logged data through filters.	**3**	**3**
8.1 As a childcare provider, I want to be provided with a form to guide me through my daily risk assessments.	**4**	**2**
8.2 As a childcare provider, I want to be provided with a form to guide me through my daily COVID checks.	**4**	**2**
8.3 As a childcare provider, I want to be provided with a form to guide me through my monthly fire drills.	**4**	**2**
8.4 As a childcare provider, I want to be provided with a form to guide me through my monthly fire safety equipment check.	**4**	**2**
8.5 As a childcare provider, I want to have the ability to edit information that I have logged in my health and safety checks.	**4**	**2**
8.6 As a childcare provider, I want to view the status of my health and safety checks.	**4**	**2**
Total		**15**
Sprint 4		
2.1 As a childcare provider, I want to have the ability to upload policies from my device onto the application.	**3**	**3**
2.2 As a childcare provider, I want to have the ability to delete my policies.	**3**	**3**
2.3 As a childcare provider, I want to have the ability to view my policies.	**3**	**3**
2.4 As a childcare provider, I want to have the ability to share my policies.	**4**	**2**
Total		**11**
Sprint 5		
11.1 As a childcare provider, I want to have the ability to track mileage expenses.	**4**	**2**
11.2 As a childcare provider, I want to have the ability to track invoices.	**4**	**2**
11.3 As a childcare provider, I want to have the ability to track other relevant expenses (equipment, gifts, fuel etc).	**4**	**2**
11.4 As a childcare provider, I want to have access to an overview of my expenses.	**3**	**3**
Total		**9**
Sprint 6		
11.5 As a childcare provider, I want to have the ability to upload an image of a receipt.	**3**	**3**
11.6 As a childcare provider, I want to have the ability to view the receipts that I have previously uploaded.	**4**	**2**
11.7 As a childcare provider, I want to have the ability to calculate my mileage expenses within the application.	**3**	**3**
Total		**8**
Sprint 7		
3.8 As a childcare provider, I want to have the ability to phone emergency contacts directly through the application.	**4**	**2**
4.1 As a childcare provider, I want to have the ability to scan a food item and understand if it is healthy for a child to consume.	**3**	**3**
4.2 As a childcare provider, I want to have the ability to scan a food item and understand if it is unsuitable for a child to consume due to their allergies.	**3**	**3**
Total		**8**

Figure 3.1: Sprint content preparation

3.5 User Requirements Template

The list of requirements was finalised using the VOLERE model template in the following format:

ID	Requirement Type	Value x Complexity score	MoSCoW category
Title			
Rationale			

Acceptance Criteria

Table 3.1: User requirements template

3.6 Final User Requirements List

1.1	Functional Requirement	25	Must Have
As a childcare provider, I want to log visitor information.			
This will ensure that there is a digital record created for all visitor logs, should it need to be referenced in the future.			
The childcare provider can log visitor information which is stored in a database.			

Table 3.2: Requirement 1.1 definition

1.2	Functional Requirement	25	Must Have
As a childcare provider, I want to log medicine administration about each child.			
This will ensure that there is a digital record created for all medicine administration logs, should it need to be referenced in the future.			
The childcare provider can log medicine administration information which is stored in a database.			

Table 3.3: Requirement 1.2 definition

1.3	Functional Requirement	25	Must Have
As a childcare provider, I want to log daily attendance information about each child.			
This will ensure that there is a digital record created for all daily attendance logs, should it need to be referenced in the future.			
The childcare provider can log daily attendance information which is stored in a database.			

Table 3.4: Requirement 1.3 definition

1.4	Functional Requirement	25	Must Have
As a childcare provider, I want to log accident reports if an incident occurs.			
This will ensure that there is a digital record created for all accident report logs, should it need to be referenced in the future.			
The childcare provider can log accident report information which is stored in a database.			

Table 3.5: Requirement 1.4 definition

1.5	Functional Requirement	8	Will Not Have
As a childcare provider, I want to have the ability to allow parents to sign forms electronically.			
This will allow the digitalisation of form signing entirely, avoiding the need to print and acquire a hand-written signature from the parent which must then be stored physically.			
The childcare provider can obtain electronic signatures from parents to fulfil form-filling processes.			

Table 3.6: Requirement 1.5 definition

1.6	Functional Requirement	20	Must Have

As a childcare provider, I want to have the ability to edit the information that I have logged.
This will allow me to correct any errors that may have been entered to ensure that information is maintained as accurately as possible.
The childcare provider can edit the information which they have previously logged. These changes are implemented on a database.

Table 3.7: Requirement 1.6 definition

1.7	Functional Requirement	15	Should Have
As a childcare provider, I want to have the ability to view any of the information that I have submitted via forms.			
This will allow me to review details of recorded logs from the past.			
The childcare provider can view the information which they have previously logged.			

Table 3.8: Requirement 1.7 definition

1.8	Functional Requirement	15	Should Have
As a childcare provider, I want to have the ability to view my logged data through filters.			
This will allow me to easily identify data, when navigating through large quantities.			
The childcare provider can use filters to reduce the results which are returned by the application.			

Table 3.9: Requirement 1.8 definition

1.9	Functional Requirement	25	Must Have
As a childcare provider, I want to have the ability to delete the information that I have submitted via forms.			
This will allow me to remove any unwanted details from the application.			
The childcare provider can delete information which has been submitted previously by them, this removes this information from a database.			

Table 3.10: Requirement 1.9 definition

2.1	Functional Requirement	15	Should Have
As a childcare provider, I want to have the ability to upload policies from my device onto the application.			
This will allow me to upload and store documents for future access.			
The childcare provider can upload documents to a private and secure database through the application.			

Table 3.11: Requirement 2.1 definition

2.2	Functional Requirement	15	Should Have
As a childcare provider, I want to have the ability to delete my policies.			
This will allow me to remove any documents which no longer need to be stored for reference.			
The childcare provider can delete policies within the application which removes them from a database.			

Table 3.12: Requirement 2.2 definition

2.3	Functional Requirement	15	Should Have
As a childcare provider, I want to have the ability to view my policies.			
This will allow me to digitally view documents at any time with easy access.			
The childcare provider can view their policies through a PDF viewer, allowing them to have features such as navigate and zoom when viewing the document.			

Table 3.13: Requirement 2.3 definition

2.4	Functional Requirement	16	Should Have
As a childcare provider, I want to have the ability to share my policies.			
This will allow me to have a simple method of sharing relevant documents externally with others.			
The childcare provider has an option for sharing the document externally.			

Table 3.14: Requirement 2.4 definition

3.1	Functional Requirement	25	Must Have
As a childcare provider, I want to record dietary requirements about each child.			
This ensures that an accurate record has been made for dietary requirement information, should it need to be referenced in the future.			
The childcare provider can record dietary requirement information for children in their care, this data is stored in a database.			

Table 3.15: Requirement 3.1 definition

3.2	Functional Requirement	25	Must Have
As a childcare provider, I want to record medical information about each child.			
This ensures that an accurate record has been made for medical information, should it need to be referenced in the future.			
The childcare provider can record medical information for children in their care, this data is stored in a database.			

Table 3.16: Requirement 3.2 definition

3.3	Functional Requirement	25	Must Have
As a childcare provider, I want to record emergency contact information about each child.			
This ensures that an accurate record has been made for emergency contact information, should it need to be referenced in the future.			
The childcare provider can record emergency contact information for children in their care, this data is stored in a database.			

Table 3.17: Requirement 3.3 definition

3.4	Functional Requirement	20	Must Have
As a childcare provider, I want to have the ability to update this information about each child should their circumstances change throughout their time in my care.			
This allows me to correct any errors that may have been entered to ensure that information is maintained as accurately as possible.			

The childcare provider can update information of children in their care, these changes are implemented on a database.

Table 3.18: Requirement 3.4 definition

3.5	Functional Requirement	25	Must Have
As a childcare provider, I want to have the ability to add additional children to the application.			
This ensures that I can keep the application updated with all children currently in my care.			
The childcare provider can add children into the application through submitting details which are stored in a database.			

Table 3.19: Requirement 3.5 definition

3.6	Functional Requirement	25	Must Have
As a childcare provider, I want to have the ability to remove children from my application, should they no longer be in my care.			
This ensures that I can keep the application updated with all children currently in my care.			
The childcare provider can remove children and their relevant information from the database.			

Table 3.20: Requirement 3.6 definition

3.7	Functional Requirement	15	Should Have
As a childcare provider, I want to have the ability to view child details should I need to retrieve this information.			
This allows me to review details of recorded logs from the past.			
The childcare provider can view the child details information which they have previously logged.			

Table 3.21: Requirement 3.7 definition

3.8	Functional Requirement	12	Could Have
As a childcare provider, I want to have the ability to phone emergency contacts directly through the application.			
This allows me to have quick access to emergency contact information and avoids having to mix these details with my personal contacts.			
The childcare provider has quick access to make calls to emergency contacts that have been stored on the application.			

Table 3.22: Requirement 3.8 definition

4.1	Functional Requirement	12	Could Have
As a childcare provider, I want to have the ability to scan a food item and understand if it is healthy for a child to consume.			
This will improve the speed that I can carry out these checks to ensure the children in my care are eating food which is healthy.			
The childcare provider can scan the barcode of a product and understand the nutritional values of that product.			

Table 3.23: Requirement 4.1 definition

4.2	Functional Requirement	12	Could Have

As a childcare provider, I want to have the ability to scan a food item and understand if it is unsuitable for a child to consume due to their allergies.
This will improve the speed that I can carry out these checks for the safety of the children in my care.
The childcare provider can scan the barcode of a product and understand the allergens within the product and which children this product is unsuitable for.

Table 3.24: Requirement 4.2 definition

5.1	Functional Requirement	4	Will Not Have
As a childcare provider, I want to have the ability to upload images from day trips.			
This simplifies the process of manually messaging each family a copy of these images, instead only one upload is required.			
The childcare provider can upload images from day trips within the application, these images are stored within a database.			

Table 3.25: Requirement 5.1 definition

5.2	Functional Requirement	4	Will Not Have
As a childcare provider, I want to have an automated tool that shares images from day trips with parents.			
This simplifies the existing process of having to manually select which photos to share with the relevant parents.			
The childcare provider can have their uploaded images from day trips automatically forwarded for parents to access.			

Table 3.26: Requirement 5.2 definition

6.1	Functional Requirement	6	Will Not Have
As a childcare inspector, I want to have read-only access of the childcare provider's health & safety information.			
This allows me to complete components of the annual assessment before arriving at the premises.			
The childcare inspectors have read-only access granted which allows them to view the childcare provider's health & safety information which has been previously logged.			

Table 3.27: Requirement 6.1 definition

6.2	Functional Requirement	6	Will Not Have
As a childcare inspector, I want to have read-only access of the childcare provider's attendance information.			
This allows me to complete components of the annual assessment before arriving at the premises.			
The childcare inspectors have read-only access granted which allows them to view the childcare provider's attendance information which has been previously logged.			

Table 3.28: Requirement 6.2 definition

7.1	Functional Requirement	4	Will Not Have

As a parent, I want to have read-only access of my child's personal information.
This allows me to have a record of the childcare provider's details of my child, should I need to raise any areas that need updated.
Parents have read-only access granted which allows them to view the childcare provider's information which has been stored regarding their child's details.

Table 3.29: Requirement 7.1 definition

7.2	Functional Requirement	4	Will Not Have
As a parent, I want to have read-only access of my child's daily attendance reports.			
This allows me to have a record of relevant detail for my child throughout a given day.			
Parents have read-only access granted which allows them to view the childcare provider's daily attendance reports for their child.			

Table 3.30: Requirement 7.2 definition

8.1	Functional Requirement	20	Must Have
As a childcare provider, I want to be provided with a form to guide me through my daily risk assessments.			
This will improve the accuracy and speed at which I can carry out daily risk assessments, ensuring that the safety of the children in my care is maximised.			
The childcare provider is provided with a form to guide them through the expected checks involved in a daily risk assessment. The details of the completed daily risk assessments will be stored in a database.			

Table 3.31: Requirement 8.1 definition

8.2	Functional Requirement	20	Must Have
As a childcare provider, I want to be provided with a form to guide me through my daily COVID checks.			
This will improve the accuracy and speed at which I can carry out daily COVID checks, ensuring that the safety of the children in my care is maximised			
The childcare provider is provided with a form to guide them through daily COVID checks. The details of the completed daily COVID checks will be stored in a database.			

Table 3.32: Requirement 8.2 definition

8.3	Functional Requirement	20	Must Have
As a childcare provider, I want to be provided with a form to guide me through my monthly fire drills.			
This will improve the accuracy and speed at which I can carry out monthly fire drills, ensuring that the safety of the children in my care is maximised.			
The childcare provider is provided with a form to guide them through monthly fire drills. The details of the completed monthly fire drills will be stored in a database.			

Table 3.33: Requirement 8.3 definition

8.4	Functional Requirement	20	Must Have
As a childcare provider, I want to be provided with a form to guide me through my monthly fire safety equipment check.			
This will improve the accuracy and speed at which I can carry out monthly fire safety equipment checks, ensuring that the safety of the children in my care is maximised.			
The childcare provider is provided with a form to guide them through monthly fire safety equipment checks. The details of the completed monthly fire safety equipment checks will be stored in a database.			

Table 3.34: Requirement 8.4 definition

8.5	Functional Requirement	20	Must Have
As a childcare provider, I want to have the ability to edit information that I have logged in my health and safety checks.			
This allows the childcare provider to correct any errors that may have been entered during health and safety checks to ensure that information is maintained as accurately as possible.			
The childcare provider has access to previously submitted health and safety checks and has permission to update the data stored within the database.			

Table 3.35: Requirement 8.5 definition

8.6	Functional Requirement	20	Must Have
As a childcare provider, I want to view the status of my health and safety checks.			
This will help me to understand and identify any checks that I may have missed.			
The childcare provider can clearly view the status of each health and safety check per day, this status should display if the check has or hasn't been completed.			

Table 3.36: Requirement 8.6 definition

9.1	Functional Requirement	8	Will Not Have
As a childcare provider, I want to be provided with reminders for my daily and monthly & safety checks.			
This will ensure that I have carried out the correct safety measures that I am required to complete.			
The childcare provider receives notifications to remind them to complete their daily and monthly safety checks.			

Table 3.37: Requirement 9.1 definition

10.1	Functional Requirement	3	Will Not Have
As a childcare provider, I want to increase the speed that repetitive logs can be entered by using previous information to predict my input.			
This will reduce the time that it takes for logs to be submitted.			
When the childcare provider is submitting logs, they should have suggestions appear within inputs that means they can avoid manually adding such data.			

Table 3.38: Requirement 10.1 definition

10.2	Functional Requirement	3	Will Not Have
As a childcare provider, I want to reduce the error rate within my logged information. The application should detect potential errors from the information that I have logged.			
This will prevent invalid data from being submitted within the application and stored within a database.			
The application must validate user input to ensure that only valid inputs will be accepted.			

Table 3.39: Requirement 10.2 definition

11.1	Functional Requirement	20	Must Have
As a childcare provider, I want to have the ability to track mileage expenses.			
Having a digital record of mileage expenses will make it much easier to calculate my total expenses at the end of the financial period.			
The childcare provider can add mileage expenses within the application, this information is stored within a database.			

Table 3.40: Requirement 11.1 definition

11.2	Functional Requirement	20	Must Have
As a childcare provider, I want to have the ability to track invoices.			
Having a digital record of invoices will make it much easier to calculate my total income at the end of the financial period.			
The childcare provider can add invoices within the application, this information is stored within a database.			

Table 3.41: Requirement 11.2 definition

11.3	Functional Requirement	20	Must Have
As a childcare provider, I want to have the ability to track other relevant expenses (equipment, gifts, fuel etc).			
Having a digital record of all other expenses will make it much easier to calculate my total expenses at the end of the financial period.			
The childcare provider can add all other relevant expenses within the application, this information is stored within a database.			

Table 3.42: Requirement 11.3 definition

11.4	Functional Requirement	12	Could Have
As a childcare provider, I want to have access to an overview of my expenses.			
This will provide me with an overall understanding and breakdown of my expenses.			
The childcare provider should be presented with an overview of their total expenses across a range of time.			

Table 3.43: Requirement 11.4 definition

11.5	Functional Requirement	12	Could Have
As a childcare provider, I want to have the ability to upload an image of a receipt.			

This will allow me to have a digital reference to the expense that I have made and means that I won't need to hold onto large amounts of physical receipts.
The childcare provider has the option to upload an image of a receipt when submitting an expense.

Table 3.44: Requirement 11.5 definition

11.6	Functional Requirement	16	Should Have
As a childcare provider, I want to have the ability to view the receipts that I have previously uploaded.			
This will allow me to view the details of expenses that I have made.			
The childcare provider can view the receipts which they have previously uploaded within the application.			

Table 3.45: Requirement 11.6 definition

11.7	Functional Requirement	12	Could Have
As a childcare provider, I want to have the ability to calculate my mileage expenses within the application.			
This will improve the speed that I can complete expenses, allowing me to focus more attention on other areas of my role.			
The childcare provider has a feature within the mileage expense page to calculate the total expense for mileage based on miles driven and the variable rate.			

Table 3.46: Requirement 11.7 definition

12.1	Functional Requirement	6	Will Not Have
As a childcare provider, I want to have a planning feature that allows me to add notable events such as children's birthdays, inspection dates, renewal dates, etc.			
This will help me to prepare for these events with the required preparation and help me to maintain a separation between my private calendar used for non-work-related events.			
The childcare provider has access to a planning feature within the application which allows them to plan events. This data will be added into a database.			

Table 3.47: Requirement 12.1 definition

12.2	Functional Requirement	6	Will Not Have
As a childcare provider, I want to have the ability to edit events within the planning feature.			
This will allow me to amend the details of the event should the date, time or details need to be changed.			
The childcare provider can update the information stored within the events of the planner. The changes will be updated in a database.			

Table 3.48: Requirement 12.2 definition

12.3	Functional Requirement	6	Will Not Have
As a childcare provider, I want to have the ability to delete events within the planning feature.			

This will allow me to remove any events planner should I require to do so.
The childcare provider can remove events from the planner. The information relating to that event will be removed from a database.

Table 3.49: Requirement 12.3 definition

12.4	Functional Requirement	6	Will Not Have
As a childcare provider, I want to receive notifications for these planned events.			
This will notify me when these events are upcoming, ensuring that I have considered the required preparation.			
The childcare provider receives notifications to remind them of upcoming events, this event notice will be a custom value specified by the childcare provider.			

Table 3.50: Requirement 12.4 definition

13.1	Non-functional Requirement	Must Have
The application should be coloured appropriately with colour-blind users in consideration.		
This will make the design of the application appropriate for a wider range of users as colour blindness has been considered.		
Blue and orange colours should be used predominantly throughout the application as these are the most colour-blind-friendly colours.		

Table 3.51: Requirement 13.1 definition

14.1	Non-functional Requirement	Must Have
The application should be available on mobile devices.		
This will ensure the app has flexible availability whenever I should need access to it.		
The application is fully available on mobile devices.		

Table 3.52: Requirement 14.1 definition

15.1	Non-functional Requirement	Must Have
A database should be used to store, edit and delete data logged by the user		
This will ensure that I can add, edit or remove information using the application when required.		
The application is connected to a database which allows the user to insert, update or delete data.		

Table 3.53: Requirement 15.1 definition

16.1	Non-functional Requirement	Must Have
A user guide should be provided to help explain the core functionality of the application.		
This will ensure that I fully understand how each feature works.		

A user guide will be created to provide explanations of how functionality within the applications operates.

Table 3.54: Requirement 16.1 definition

17.1	Non-functional Requirement	Must Have
The application should contain validation mechanisms.		
This will ensure that the user's input will always be valid, preventing invalid data from throwing errors in the application.		
Validation will be carried out within forms where user input is logged.		

Table 3.55: Requirement 17.1 definition

18.1	Non-functional Requirement	Must Have
In scenarios where the device's internal storage must be accessed, permissions must first be accepted by the user.		
This will ensure that the user's has provided permission for their internal storage to be accessed by the application, this is good common practice to maintain within mobile applications.		
When the device's internal storage must be accessed, permission will be requested for the user to accept before actions can proceed.		

Table 3.56: Requirement 18.1 definition

4. Project Planning

4.1 Project Management Tools

When it came to selecting the project management tool that he wanted to use throughout his project, Owen assessed a few services. These options were Trello, Kanbanize and Jira. After evaluating each option, Owen decided to utilise Trello over other alternatives as it is free to use and he had prior experience of using Trello in his occupation. The other two options require payments to use their service and Owen would need some upskilling to fully utilise the services.

Owen utilised Trello for managing the workload of his sprints. Using it's Kanban-like style to manage columns representing the status of tasks. This included 'Sprint Backlog', 'In Progress' and 'Completed'. Trello allows each individual card to be labelled, which enabled Owen to establish a colour-coded system for each sprint iteration. This provided a good visual tool for analysing the progress of a sprint at a quick glance. Trello also provides the option of adding a description to each card on the board. Owen utilised this feature to include development tasks and include relevant links to guides or examples of similar work that he had found within documentation online. The Trello board also provided a good foundation for Owen when it came to writing his report as he had references to completed work with the relevant sources that he used to help him, such as APIs and external modules.

Owen utilised Office Timeline Online to generate his Gantt chart for this project. This is a cloud-based timeline maker which provides templates for the likes of Gantt charts. This tool allowed Owen to plan the dates of each of his sprints and key dates such as development start and end dates, coursework deadlines and show and tell sessions with the stakeholder. Owen decided to use Office Timeline Online over other alternatives as it was free to use and provided some extensive templates which allowed him to complete the planning process with fewer complications.

4.2 Version Control

Owen managed his code throughout his project through utilising GitHub. This is a version control provider that allowed Owen to divide his sprints into separate branches within his GitHub project. This helped to improve the traceability of the code and provided a back-up should Owen's laptop break or malfunction. Due to the code being hosted on a managed platform such as GitHub, the project could be easily setup on a new device if that was required.

4.3 Risk Assessment

After analysing the development tasks within the project sprints, Owen conducted a risk assessment to identify potential pitfalls along the development journey. The aim of this was to take measures to minimise delays or establish back-up plans should they be required.

Owen used a severity and likelihood scale when measuring each risk. This was a similar process that he used for calculating a priority score for user stories. Severity was measured from 1-5 with 5 being the most severe and 1 being the least severe. Likelihood was similarly measured from 1-5 with 5 being likely to occur and 1 being unlikely to occur. Through multiplying these two figures together, this provides an overall risk value. For each risk that was identified, a mitigation strategy was considered. Should the risk occur, this strategy would be implemented as a back-up option. For each mitigation, this helped to reduce the likelihood or severity of the risk, which helped to reduce the overall risk value.

Item	Description	Severity	Likelihood	Risk
Loss of code progression	Should Owen's laptop crash or break during development, he will lose progress of his code.	5	2	10
Outdated node packages	Node packages that Owen has shortlisted may be outdated or contain bugs within them. This may impact certain features that he had hoped to use the packages for.	3	3	9
Issues establishing	Owen doesn't have much experience in establishing a backend within webapps – especially for mobile web apps. If he struggled	4	2	8

the database backend	to construct the database within the backend, it could delay a lot of his development progress.			
Not all key features will be developed	There are lots of requirements to work through and due to this, Owen may not get to develop all of them.	4	2	8
Food API – Open Food Facts	An open source API is used to retrieve information regarding allergens within food products. There is risk that the API may not function properly or is no longer supported.	2	2	4
Not enough time left for testing	There are plenty of tasks within the requirements documentation and it may limit the amount of testing that Owen can complete.	2	2	4
Struggling to work with React Native	Owen has never used React Native to develop an app and so there will be a period of familiarising himself with how the development & debugging processes work.	4	1	4
Not enough time for dissertation	Owen may not have enough time to finish his dissertation.	4	1	4

Table 4.1 Risk Assessment

4.3.1 Mitigation Strategies

Item	Mitigation	Severity	Likelihood	Risk
Loss of code progression	Owen will be using version control (GitHub) to safely maintain the progress of his development.	5	1	5
Outdated node packages	Owen has shortlisted several similar packages for each purpose that he requires them for. Through doing so, this will provide him with a backup option should he need one.	3	2	6
Issues establishing the database backend	There is plenty of documentation and guidance for establishing databases for React Native as it's a common requirement. Owen has allocated time before sprint 1 to setup his development environment and prepare the database.	4	1	4
Not all key features will be developed	Owen has used requirements prioritisation to sort his list of user requirements in order of importance. Through doing this, he knows the exact order that he will be completing development tasks and has planned this work into realistic sprint plans.	4	1	4

Food API – Open Food Facts	Owen has identified a backup API, should his chosen API not be usable.	2	1	2
Not enough time left for testing	Owen has used requirements prioritisation to sort his list of requirements. Using MoSCoW, it will ensure that he completes the high priority requirements. This will allow Owen to utilise the remaining time to complete testing of his application	2	1	2
Struggling to work with React Native	There is an established developer community for React Native which was one of the main reasons that Owen selected it over alternative options. There is also an extensive number of guides & tutorials that can be found online for React Native. Owen has allocated time before his first sprint to familiarise himself with React Native.	3	1	3
Not enough time for dissertation	Throughout his coursework to date, Owen has been constantly adding detail to his dissertation. This means that after development has been completed, there will be less time pressure on Owen to finalise his dissertation.	3	1	3

Table 4.2: Risk Assessment after mitigation strategies applied

4.4 Licenses

Firebase was the cloud service that Owen availed to host the database for his application. Firebase is a toolset used when building an application. These tools help the developer to build large portions of the services that they would otherwise need to develop themselves. This allows the developers to focus more intensely on the application experience itself and focus more on their stakeholder. Firebase includes services such as analytics, authentication, databases, configuration, file storage, push messaging, and much more. The services are hosted in the cloud, and scale with little to no effort on the part of the developer (Stevenson, 2018). There were a few services that Owen immediately realised he could utilise within his application. Such services included Cloud Firestore, Cloud Storage and Realtime Database to help him achieve his stakeholder's requirements.

4.4.1 Cloud Firestore

Cloud Firestore is a NoSQL database built for automatic scaling, high performance, and ease of application development. While the Firestore interface has many of the same features as traditional databases, as a NoSQL database it differs from them in the way it describes relationships between data objects (Firebase, 2022).

4.4.2 Cloud Storage

Cloud Storage is used to store and serve user-generated content, such as photos or videos. It is a powerful, simple, and cost-effective object storage service that adds Google security to file uploads and downloads for an application, regardless of network quality. Cloud Storage stores files in a Google Cloud Storage bucket, making them accessible through both Firebase and Google Cloud. This allows the flexibility to both upload and download files from mobile clients via the Firebase SDKs for Cloud Storage (Firebase, 2022).

4.4.3 Realtime Database

The Realtime Database is like Cloud Firestore in that it is also a cloud-hosted NoSQL database. Although data within the Realtime Database is stored as JSON and remains fully available when an app goes offline (Firebase, 2022). Owen utilised the free tier that Firebase offered as it provided sufficient resources for his project concept. If Owen was to scale up this solution and publicly release it, it would require him to upgrade his Firebase plan to a premium-paid version. This premium version operates on a pay-as-you-go basis which is based on the service being used and the amount of resources consumed.

4.5 Software Development Lifecycle Methodology

Upon considering the Software Development Lifecycle Methodology to utilise within his project, there were a number of these which Owen initially investigated. These were XP (eXtreme Programming), Waterfall and Scrum. Through researching methodologies that specifically adapted towards a project completed by a solo developer, Owen discovered ‘Scrum for One’.

4.5.1 Scrum for One

Scrum for One is an agile methodology aimed towards solo developers wanting to build a project individually. It maintains the same features of Scrum such as planning, sprints, reviews and retrospectives. It also focuses on utilising the different roles and responsibilities that must be occupied by one individual throughout the project. The individual must behave as a developer, scrum master, product owner and software tester to ensure that the project runs smoothly and is delivered in an acceptable format once it is completed (Lucid Content Team, n.d.).

4.5.2 Software Development Lifecycle Methodology Analysis

It didn’t take long for Owen to eliminate Waterfall from this selection as it stood out quite clearly that it wouldn’t suit the structure of this project. This is due to it being far too rigid and not suiting the flexibility needed when

handling the requirements. Because Owen was planning to be in frequent contact with his stakeholder, he knew that changing requirements could be a possibility and Waterfall didn't offer the flexibility required to fulfil this. Owen wanted to ensure that he had made the correct choice and so researched and evaluated XP and Scrum-for-one, balancing the benefits and drawbacks of each of these. He decided to choose Scrum in favour of XP as he has experience using it within his day-to-day role as a Software Developer. He is very familiar with how it operates and the significance of each role within the Scrum team. The processes are familiar to him and present little friction which may have otherwise occurred should he need to familiarise myself with the alternative methodologies. Due to the structure of the sprints, it suits Owen a lot more to operate with Scrum sprints as he won't be developing intensively throughout due to his part-time availability of two days per week. Scrum sprints can last from 1-4 weeks whereas XP sprints typically last 1-2 weeks. XP presents its own individual practices which may be beneficial to the structure of my project, but Scrum has adopted many of these already. Examples of these include user stories, sprint planning, continuous releases and test-driven development (Cohn, 2007).

4.6 Gantt Chart

Within his Gantt chart, Owen set the duration to only include weekdays as this was his only availability to contribute to his project. As a result, a two-week sprint consisted of ten days.

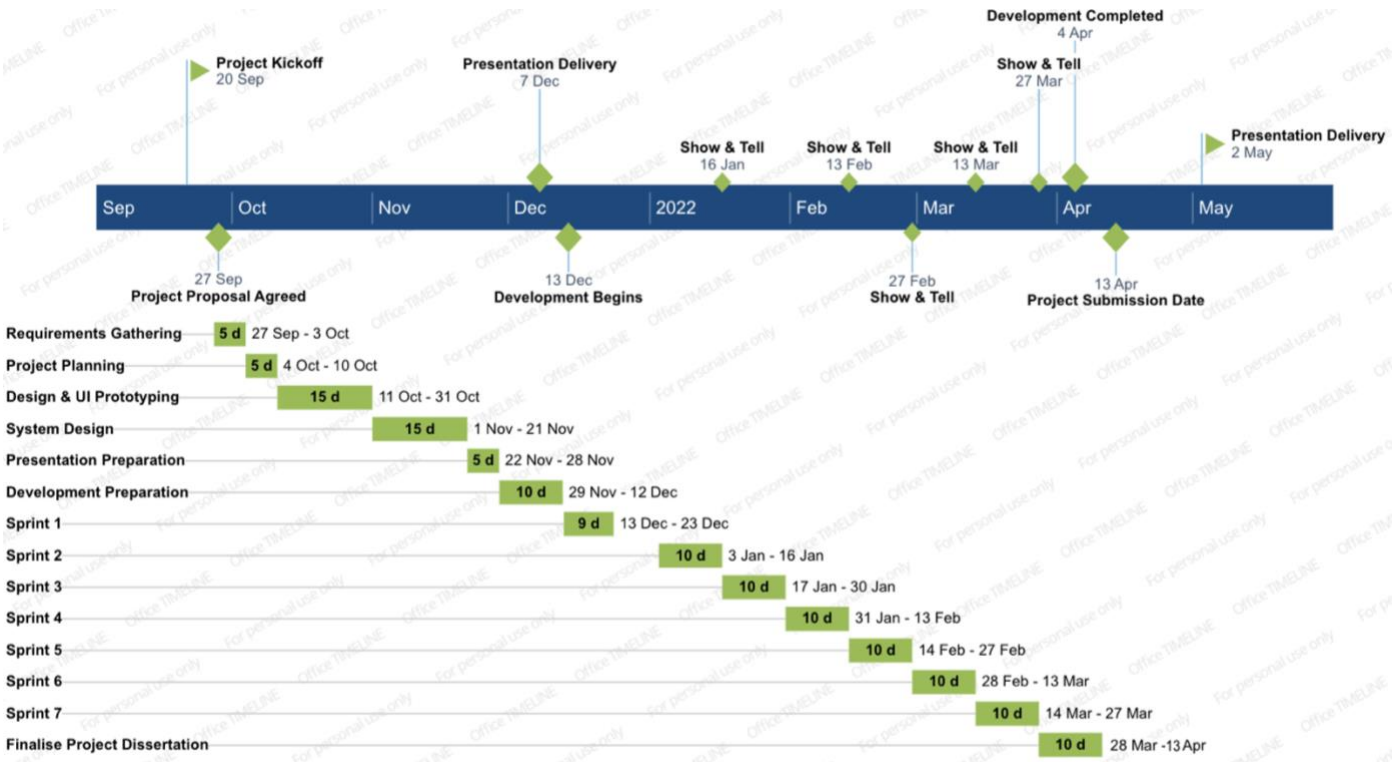


Figure 4.1: Gantt Chart

4.6.1 Requirements Gathering

Owen estimated five days to carry out his requirements gathering. This involved meeting with his stakeholder to draw up a user requirements list, converting requirements into user stories.

4.6.2 Project Planning

Owen estimated five days to implement his project planning tasks. This included creating a use case diagram, Gantt chart and establishing a Trello board.

4.6.3 Design & UI Prototyping

Owen estimated fifteen days to implement his design and UI prototyping tasks. This involved designing drafts of how the system should look, this was eventually converted into high fidelity concepts through using UXPin. Additionally, Owen also identified a colour scheme for his application during this period.

4.6.4 System Design

Owen estimated fifteen days to implement his system design. This involved creating an ER diagram to understand the entities within the database, their attributes and how their relationships would work with one another. This also included understanding how navigation within the application would function.

4.6.5 Presentation Preparation

Owen allocated five days towards preparing his presentation to deliver for the first coursework delivery component – the project stand-up review.

4.6.6 Development Preparation

Owen allocated five days towards preparation for development. This involved setting up his development environment, establishing a firebase database connection and conducting research on suitable APIs and external modules.

4.6.7 Sprints

Owen planned the durations of his seven sprints throughout the project, these would typically last two weeks which worked out as ten-day sprints (other than sprint 1 as it ran into the Christmas holiday period)

4.6.8 Project Dissertation

Owen allocated fifteen days towards completing remaining tasks within his dissertation.

4.6.9 Show & Tell

Owen scheduled show & tell sessions in advance with his stakeholder. This aligned his project to the scrum methodology and allowed Owen to take an opportunity to demonstrate functionality to his stakeholder. This provided the stakeholder with an opportunity to provide feedback on features within the application and identify any gaps within the application which could be addressed. Aligning with scrum, should the stakeholder identify functionality that they want added, Owen can add this requirement into the sprint backlog and judge the value and complexity of the task to identify which user stories will be bumped down the priority list.

4.7 Predicted vs Actual

Task	Predicted start	Predicted end	Actual start	Actual end
Requirements Gathering	27/09/21	03/10/21	27/09/21	03/10/21
Project Planning	04/10/21	10/10/21	04/10/21	10/10/21
Design & UI prototyping	11/10/21	31/10/21	11/10/21	31/10/21
System Design	01/11/21	21/11/21	01/11/21	21/11/21
Presentation preparation	22/11/21	28/11/21	22/22/21	28/11/21
Development preparation	29/11/21	12/12/21	29/11/21	12/12/21
Sprint 1	13/12/21	23/12/21	13/12/21	20/12/21
Sprint 2	03/01/22	16/01/22	21/12/21	08/01/22
Sprint 3	17/01/22	30/01/22	07/01/22	15/01/22
Sprint 4	31/01/22	13/02/22	16/01/22	02/02/22
Sprint 5	14/02/22	27/02/22	03/02/22	01/03/22
Sprint 6	28/02/22	13/03/22	02/03/22	16/03/22
Sprint 7	14/03/22	27/03/22	17/03/22	27/03/22
Finalise project dissertation	28/03/22	13/04/22	28/03/22	13/04/22

Table 4.3: Project task predictions & actual dates

The pre-development tasks which Owen had planned on the Gantt chart al met their predicted start and finish dates. This was largely due to most of the tasks being rather predictable in terms of Owen's required time contribution. Through meeting these predicted estimates, it allowed Owen to commence his development on the intended start date. Owen's development preparation ensured that he had a development environment set-up and operating. He had also taken the opportunity to complete React Native tutorials which helped him to understand the structures and methods that he would need to follow to successfully build his project. Owen completed sprint 1 tasks faster than he had initially estimated, this was due to him finding

success in establishing database features within the application and avoided hitting any major delays in this process. Through gaining a time advantage during this sprint, this allowed Owen to introduce tasks from his sprint backlog and take an early start at those. This trend continued through the next few sprints as Owen continued to meet targets for his predicted end dates of sprint tasks. However, it was during Owen's second show and tell date where the stakeholder had a good opportunity to test the functionality of the application. This allowed Owen to gain some deep insights on the features which he had created to date within the application. This allowed the stakeholder to provide positive feedback on elements that they liked, comment on any changes that they felt would improve the application and suggest any other components which would be useful to include. Some examples of these include adding a drop-down selector which is populated by children's names, including additional information within form screens and ordering data. These extra tasks increased the workload within several sprints however, it was all managed appropriately as the extra sprint allocated for remaining tasks at the end of development allowed Owen to utilise this time to finalise tasks. This allowed Owen to commence his project dissertation of the date which he had estimated which allowed him to have a sufficient duration to submit the project before its deadline has passed.

Please see **Appendix C** to view Owen's Trello, this includes the user stories that he completed during each sprint.

4.8 Hardware Selection

The hardware that Owen used within the development process was a MacBook Pro which was running MacOS Big Sur 11.6. React Native applications can also be built on Windows devices, although Owen primarily operates on iOS devices, so this was the option which best suited him.

4.9 Software Selection

When selecting suitable software, there were several options which Owen had to consider through evaluating each of them.

4.9.1 Mobile vs Desktop

The first of these was if he wanted to have a mobile or desktop application. In terms of suitability for the end-user, it would make much more sense for the application to operate on a mobile platform. A childcare provider is an active and physically demanding role where they are often going to be away from their home with the children in their care for most of the day. For this reason, a desktop application wasn't suitable at all. The end-users will be using this application on the go and on demand. The application is designed in a way that allows the childcare provider to easily add, view or update their

information with ease. Through providing a mobile option for them, this means that they can have access to a large collection of vital information while they are on the go.

4.9.2 PWA vs Native

The next decision that Owen had to make was if he wanted to build a progressive web application (PWA) or a native mobile application. A PWA is a web application that uses service workers, manifests, and other web-platform features in combination with progressive enhancement to give users an experience on par with native application. They are accessible on the web but can also be installed onto mobile devices where they can behave similarly to native applications. The main benefit of PWAs is their offline capability, although for this application it wasn't necessary to achieve the stakeholder's requirements (Mozilla, n.d.). Though this is something that Owen would be keen to add to the application, given further time to do so. Owen opted for a native solution as this would result in a more secure solution than a PWA due to higher levels of integration with the device. Additionally, it would allow him to have easier low-level access to components of the device such as phone contacts, file access and camera access.

4.9.3 Native Framework Selection

When considering his native mobile application options, he discovered frameworks that support cross-platform native support for both iOS and Android devices, the two main frameworks that do so are React Native & Flutter. React Native is a software framework that uses JavaScript to produce native iOS and Android applications. It's estimated that almost 5% of applications on the Android marketplace are built with React Native (AppBrain, 2022). Many businesses across the world are utilising the framework, from established Fortune 500 companies to new start-ups. Big names such as Meta, UberEats and Netflix have utilised the framework within their applications (React Native, 2022). Flutter has similar capabilities as React Native, but it uses the Dart programming language which is much less popular than the likes of JavaScript.

After evaluating his two native mobile application options, Owen decided that React Native would be the most suitable option from this selection as it was the most established framework which had the most documentation and support, as well as a large selection of node modules to choose from. Owen has an established understanding of how to write code in JavaScript and had no experience at all in using the Dart. Owen knew that if he selected Flutter, it would require him to allocate time towards upskilling in the programming language which could decrease his available time to contribute to development within sprints.

4.10 Database Selection

The cloud platform provider that Owen selected for his React Native application was Google's Firebase. Owen made this decision as Firebase offered several advantages which alternative options could not. This included features such as real-time data synchronisation and offline persistence – which opens the potential of turning the application into a PWA should this become a valuable requirement. Overall, Owen selected Firebase as it includes highly detailed support and guidance for integrating with React Native applications. It is also known to have a relatively minimal setup which reduces the likelihood of encountering delays within the project timeline. Firebase does include security implementations, which are sufficient for developing a concept such as this. Should this application be scaled further, data encryption would need to be implemented to ensure that sensitive information is fully protected on the database side. There were other database providers that Owen had found through completing research online. This included options such as SQLite, MySQL and AsyncStorage. Any of these options would achieve the user requirements, but Owen's decision was based on several key factors. It was largely due to the support and documentation that was provided for Firebase as well as a much more straightforward setup during development when compared to the alternatives, but Firebase's capability of being suited to real-time applications helped to set it apart from the alternatives (codemagic, 2021).

4.11 Integrated Development Environment (IDE) Selection

There were several IDE options that Owen evaluated for using within this project's implementation. These IDEs were Visual Studio Code, IntelliJ and Deco.

4.11.1 Visual Studio Code

This is a free to use solution that is lightweight and provides a quick and easy option for getting development underway. It includes several highly useful extensions that provide shortcuts, format code and improves the code's readability by highlighting the text.

4.11.2 IntelliJ

To fully utilise the capabilities of IntelliJ for React Native, the paid version of this IDE (IntelliJ Ultimate) is required. This is not something that Owen was willing to pursue as alternative options were free to use and could support React Native development fully.

4.11.3 Deco

This is an all-in-one solution that is specialised for developing React Native applications. It allows the developer to utilise a built-in simulator which implements live code changes and provides component templates which aims to increase the reusability of code and speed of development (Native, 2016). However, Deco was discontinued in 2016 so as a result, the support and maintenance for this IDE isn't maintained which has reduced its popularity in the developer community.

The IDE that Owen utilised for the development process was Visual Studio Code. This was due to Owen having prior experience with this IDE and because it offered itself as a free to use service which could be very productive for developing this project through utilising extensions. For building and compiling code, Owen utilised Expo. This is a framework that is used by most React Native developers as it provides tools and services to simplify the creation and testing of applications (Expo, 2022).

4.12 Programming Languages Selection

The programming languages the Owen used to create the application included JavaScript and TypeScript - in the form of '.tsx' (a React language extension of JavaScript). These are the programming languages which the React Native framework requires. Owen has an abundance of experience with JavaScript through his occupation and is less familiar with TypeScript, though there are many similarities between the two programming languages which helped to ease the challenge when it came to writing code in TypeScript.

4.13 Application Programming Interface (API) Selection

Owen utilised the open food facts database API to extract product information of foods which he utilised within the barcode scanning and allergy detection component of his application. This service was free to use and is populated with data which is updated by the administrators of the website and its users (Open Food Facts, n.d.). The API retrieved data such as nutritional breakdowns, an image of the product and an ingredients list including allergens and additives. Owen identified a backup API which he identified through his risk assessment. This alternative is named Edamam and offered a free tier service that would meet the demands for Owen to establish a working concept (Edamam, n.d.). He planned to use this alternative should his primary option not work as expected.

4.14 External Modules

The external modules that Owen used within his project:

4.14.1 Database

- firebase: Owen used this to establish a connection from the application to the firebase cloud platform service.

- firebase/firestore: Owen used this to write, view, update and delete data to a Firebase Cloud Firestore database.

- firebase/storage: Owen used this to upload and delete documents to a Firebase Cloud Storage Google Cloud Storage bucket, this was used to allow the user to upload policies and receipts within the application.

4.14.2 Date

- moment: Owen used this to format and parse dates.

- date-fns: Owen used this to manipulate date values, particularly the subMonths and addMonths features that the monthly date filters were built upon.

4.14.3 Navigation

- react-navigation/native-stack: Owen used this to establish a navigation system within the application.

4.14.4 UI

- react-native: Owen used this to have access to the core UI components within React Native

- react-native-elements: Owen used this as an extension to the standard React Native UI components

- react-native-paper: Owen used this as an extension to the standard React Native UI components

- @expo/vector-icons: Owen used this to include icons within the UI of the application.

- react-native-modal-selector: Owen used this to include modal selectors which were used throughout the application. This was used to filter data within the application and provided the user with an input option within form screens. For example, a selector populated with the name of all children currently under care.

- @react-native-community/datetimepicker: Owen used this to include date and time pickers within the application. This was used to filter data and improved the data validation of the application as the users couldn't submit invalid inputs.

- @react-native-community/checkbox: Owen used this to include checkbox components within form screens, these were utilised particularly within the health and safety drills where users must work through large checklists as guidance.

- react-native-progress: Owen used this to display a progress bar within the policies page, this will inform the user of their document's upload progress.

4.14.5 Allergy detection

- expo-barcode-scanner: Owen used this to enable a barcode to be scanned within the application and the product id to be returned which provides further information within detail screens.

4.14.6 Receipt upload

- expo-image-picker: Owen used this to enable the user to select an image from their device's local storage, this assisted the process of uploading a receipt to include when logging an expense.

4.14.7 Policy management

- rn-pdf-reader-js: Owen used this to enable the user to view and navigate through the policy documents that they had uploaded.

- react-native-progress: Owen used this to display a progress bar to provide a visual representation to the user of how long their policy document upload will take.

- expo-document-picker: Owen used this to enable the user to select a policy document from their device's local storage, this facilitated the process of selecting a document to upload.

4.15 Required Knowledge

There were several areas that require Owen to upskill within before and during the development. Owen had allocated a period before sprint 1 commenced to allow him time to develop his knowledge and understanding of several key components of the project's development. Some of the tasks that Owen completed during this period included:

- React Native – completing fundamental tutorials provided in the official documentation
- Expo – building a React Native application with Expo
- Expo – debugging with Expo
- Firebase – set up a Firebase database

- Firebase – establishing a connection from a Firebase endpoint to a React Native application
- Firebase – writing, updating and deleting data to a Firebase database from a React Native application

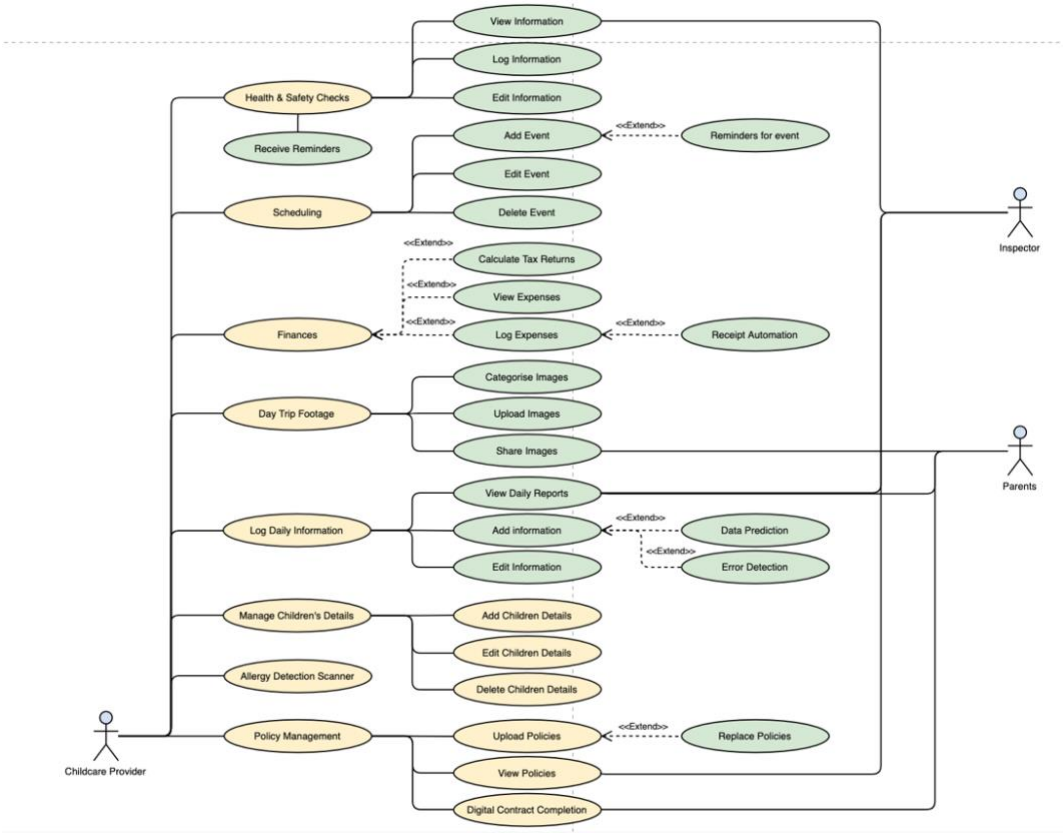
During the development stage there were several occasions which required for Owen to complete additional upskilling to his develop knowledge. This ensured that he had sufficient experience and understanding of the tools that he was using when completing the development stages of this project:

- Firebase – set up a Firebase Realtime database
- Firebase – set up a Firebase Cloud Storage database
- Firebase – uploading and removing to a Firebase Realtime database
- Firebase – uploading and removing to a Firebase Cloud Storage database
- Node – understanding how to utilise selected packages within the application

5. Design

5.1 Use Case Diagram

Owen constructed a use case diagram to capture all possible flows that any users could take when using the system. This allowed Owen to understand how each user would interact with his system, should all user stories be achieved. The use case diagram helped Owen to visualise the pages that



would be required within the application, the functionality required within each page and how the navigation system would be structured. This included the possible flows that any of the users could take within the system and how various components interacted with each other. Through completing this diagram, it allowed Owen to identify gaps within the navigation of the system and realise that additional pages were required to connect these. This diagram also assisted Owen when it came to system design stage as he had to establish the structure of his database. Through understanding what information was required for each screen, he could structure his database around this.

Figure 5.1: Use case diagram

5.2 Use Case Diagram – Technical

Owen also created an additional use case diagram, using his previous one to identify the technical requirements to satisfy each of the functionality components within each page. This provided Owen with a solid overview of the tasks required to complete the functionality of each page in the application and allowed him to start thinking about which APIs and external modules that he wished to utilise within the application.

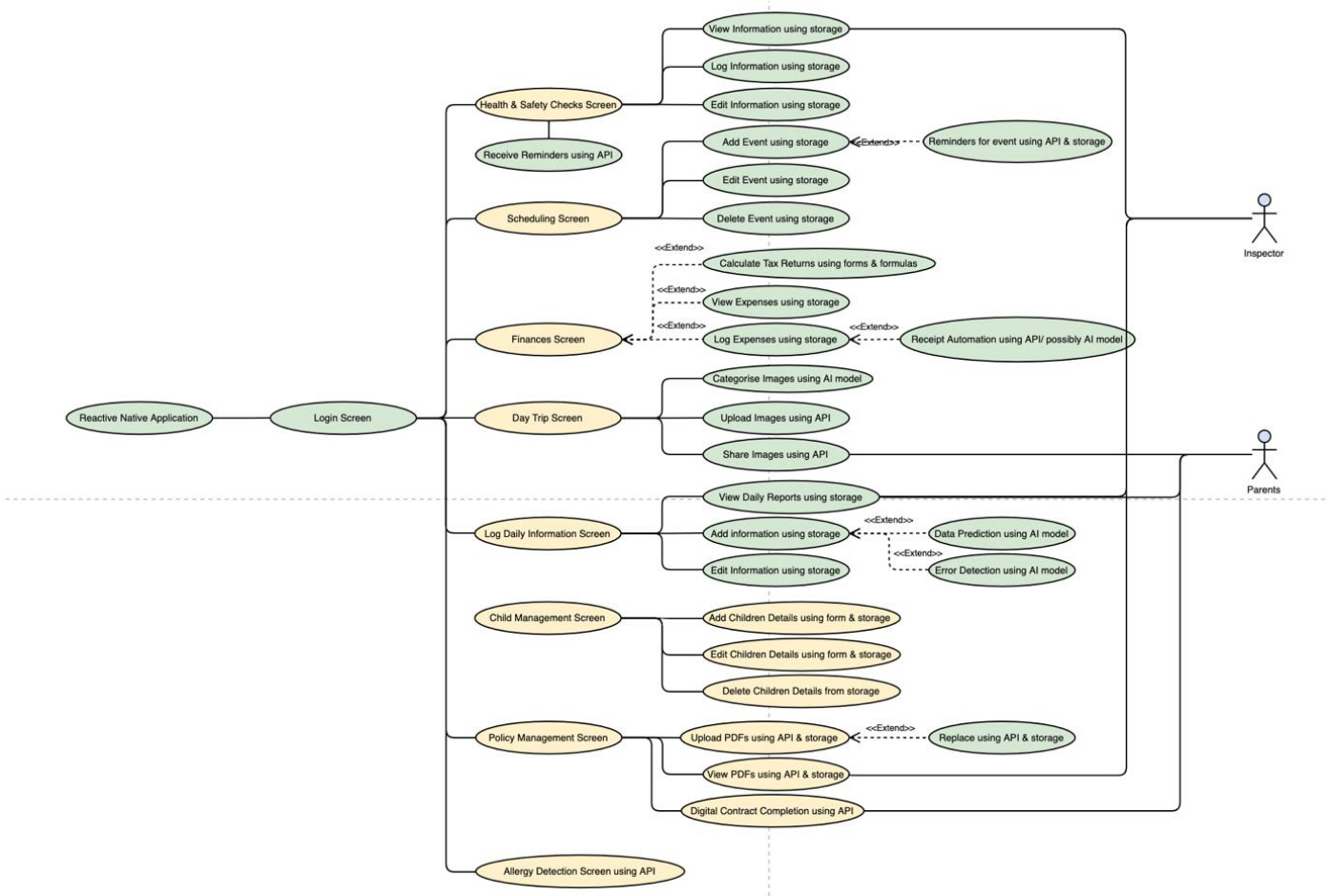


Figure 5.2: Technical use case diagram

5.3 Design wireframes

Owen began his screen designs by using an online sketching tool to create several unique design wireframes for key pages within the application, based on the priority of the user stories. These sketches included minimal detail but allowed Owen to experiment with different types of designs and create a wide range of unique concepts. Owen included notations with the sketches to expand upon the components that were included within the wireframes. This resulted in Owen designing three to four rough concepts for the different types of screens.

Please see **Appendix E** for the additional wireframe designs that Owen created.

5.3.1 Health & Safety Checks

Design 1

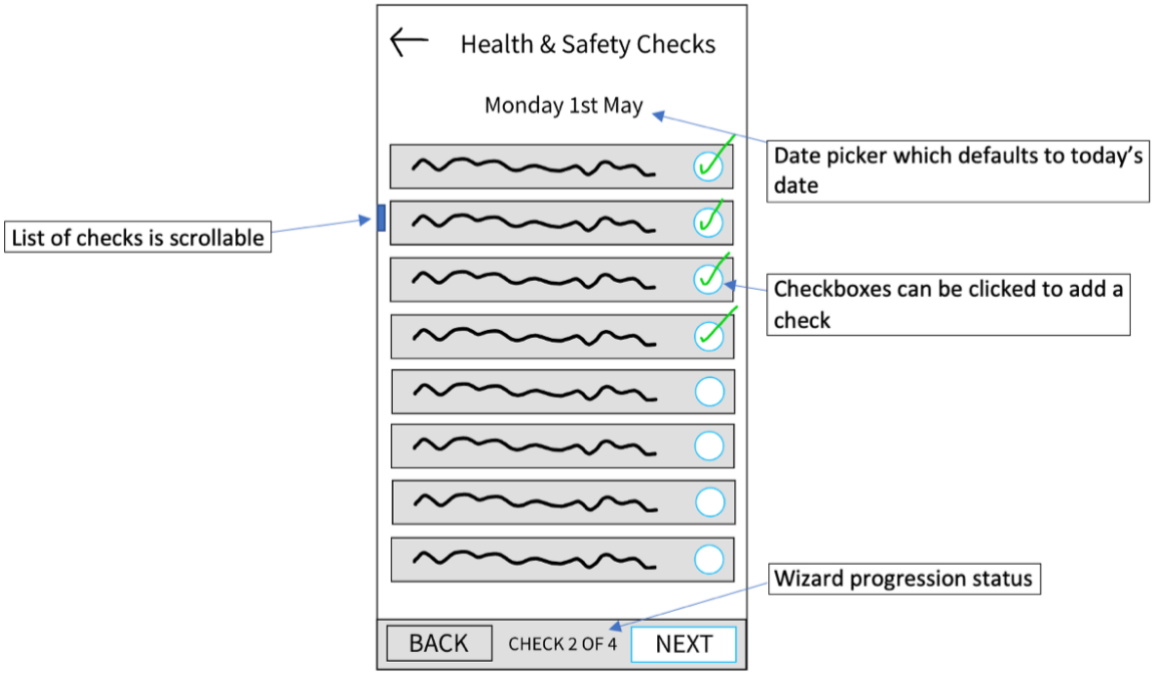


Figure 5.3: Health & Safety checks concept 1

Design 2

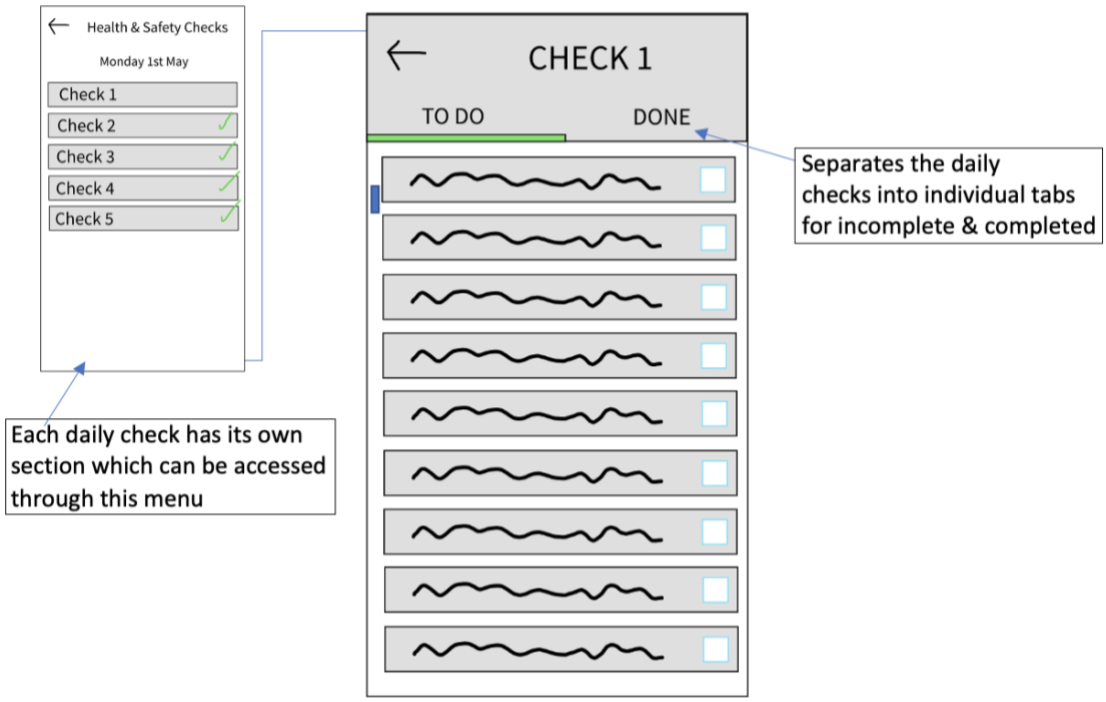


Figure 5.4: Health & Safety checks concept 2

Design 3

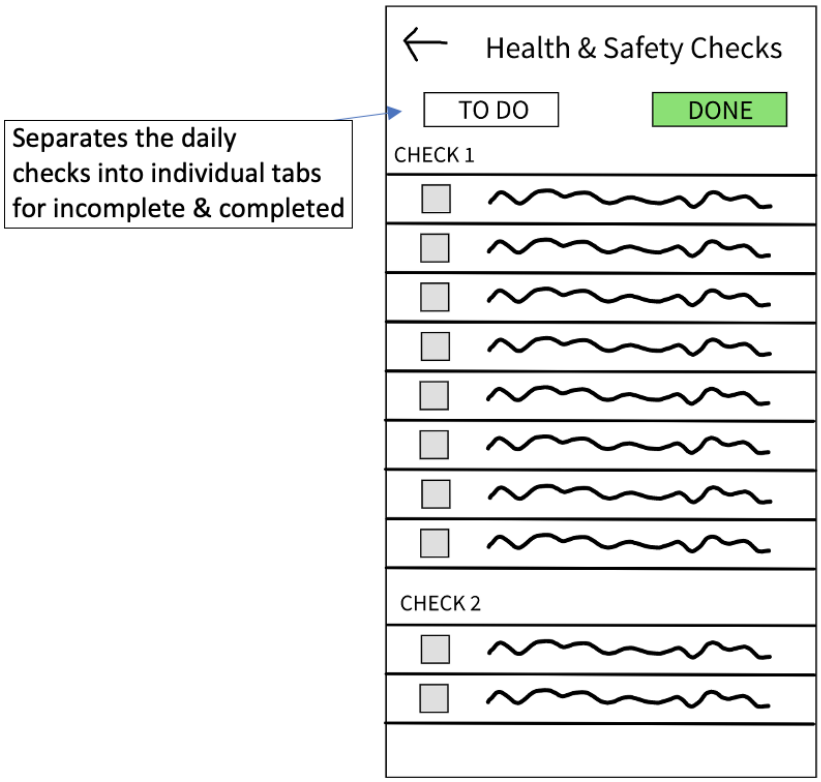


Figure 5.5: Health & Safety checks concept 3

Design 4

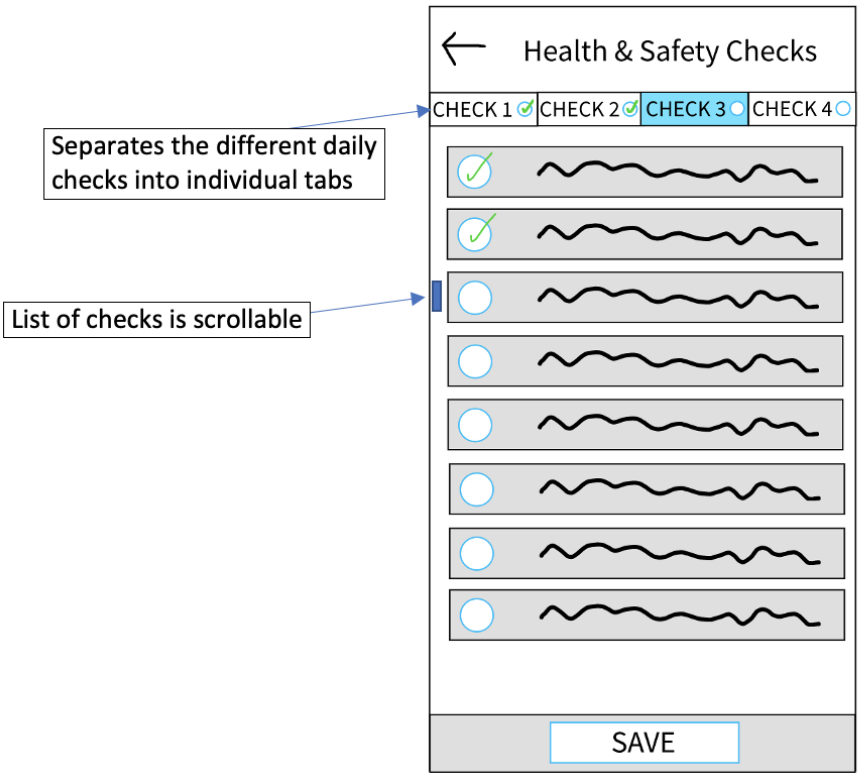


Figure 5.6: Health & Safety checks concept 4

5.4 Advanced Designs

After the wireframes were designed, this provided Owen with several options to consider for his screen designs. Using these, he began to design high-fidelity concepts of these screens using UXPin software to achieve this. Due to Owen sketching a few wireframes for each screen, his high-fidelity designs were often a combination of these multiple designs. These concepts gave Owen a much better idea of how the final product would look and provided him with a strong structure for when it came to implementing the designs within the UI during his development stage.

Figure 5.7: Advanced concept design of fire safety checks page

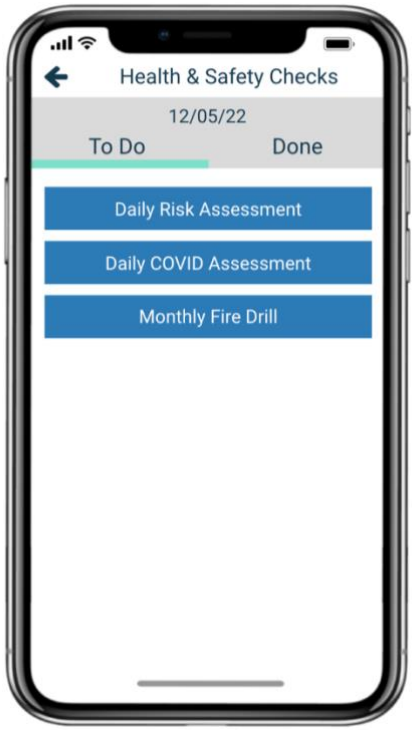


Figure 5.8: Advanced concept design of health and safety checks page

Please see **Appendix F** for the additional advanced designs that Owen created.

5.5 Colour Palettes

Owen was mindful of accessibility and the needs of his users and so considered colour blindness when selecting his colour theme. Due to this, it was rather easy for Owen to decide on his chosen colours as oranges and blues were deemed colour blind friendly. Owen initially looked at two possible options for the colour palettes for the application, these included a vivid option and a pastel option. Owen opted for the more vivid colour palette as it allowed him to highlight key information within screens that should be gaining the user's attention. Additionally, it provided a much more

vibrant appearance which is eye catching and attractive to a user. Whereas the pastel palette looked more washed and faded in comparison.

Figure 5.9: Vivid colour palette

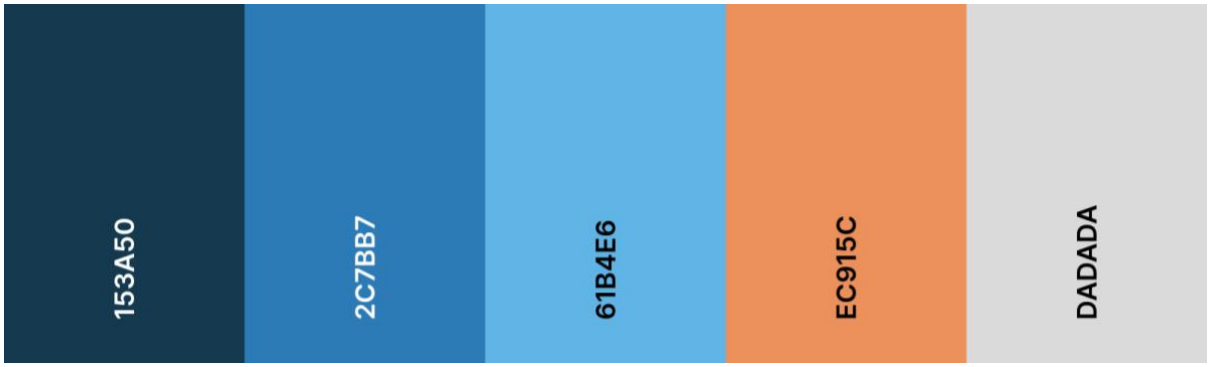


Figure 5.10: Pastel colour palette

Owen also found representations to show his selected colour palette through the eyes of an individual with colour blindness:

Figure 5.11: Protanopia colour blindness

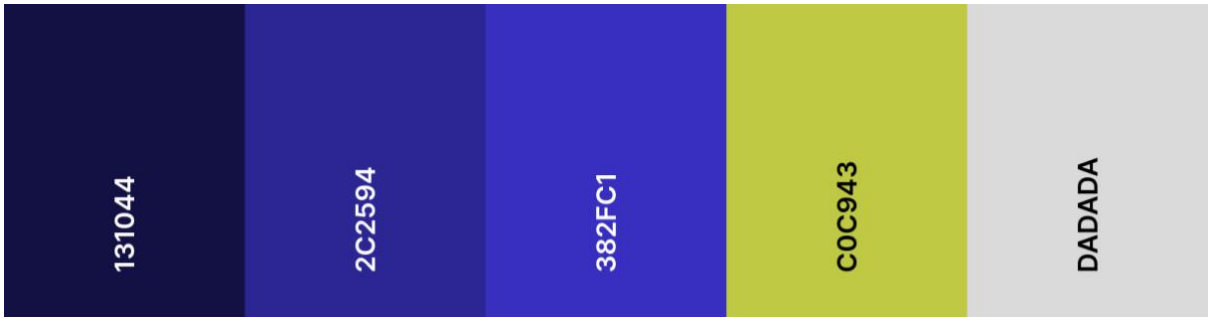


Figure 5.12: Deuteranopia colour blindness

Figure 5.13: Tritanopia colour blindness

Through choosing his colour palette carefully, this has allowed Owen to select a range of colours which satisfy the demands of users that may have a type of colour blindness. It was particularly important for Owen to have two contrasting shades which allowed him to highlight key features within the application. He utilised vivid shades of blue and orange to do so. This was represented correctly through each form of colour blindness as the shades remained contrasting.

5.6 Entity Relationship (ER) Diagram

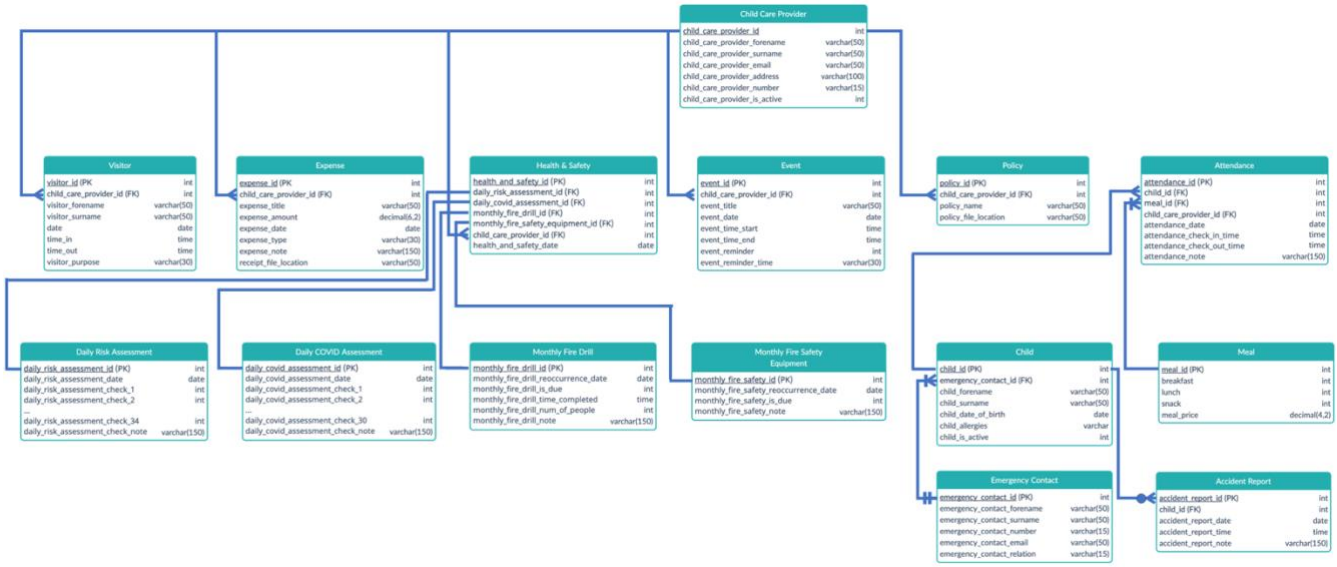


Figure 5.14: ER Diagram

Owen used an ER diagram to establish a structure to the architecture of his application, this helped to shape how the back end would look in terms of the database. An ER diagram helps to visualise the entities within the database, their relationships with one another and the attributes of each entity and the data type which it would be represented with.There were several different relationships used within this ER diagram:

5.6.1 One-to-one

This relationship was used several times to describe that each attendance report would have one meal status connected to it. Additionally, each health

and safety check would have one daily risk assessment, one daily COVID assessment, one monthly fire drill and one monthly fire safety equipment check.

Figure 5.15: One-to-one relationship

5.6.2 One-to-many

This is the most common relationship used in the ER diagram. A childcare provider can have many visitor logs, expense logs, attendance logs, health and safety checks, events and policies. Additionally, each child will be marked in many attendance logs.

Figure 5.16: One-to-many relationship

5.6.3 One-to-one-and-only-one

This relationship is used only once in the ER diagram as there will be one and only one emergency contact for each child.

Figure 5.17: One-to-one-and-only-one relationship

5.6.4 One-to-one-or-many

This relationship is used only once in the ER diagram. Each emergency contact has 1 or many children, should there were multiple siblings with the same childcare provider.

Figure 5.18: One-to-one-or-many relationship

5.6.5 One-to-zero-or-many

This relationship is used only once in the ER diagram as each child can have 0 or many accident reports, depending on how many accidents they are involved in.

Figure 5.19: One-to-zero-or-many relationship

During development, Owens plans had altered when he decided to opt for using Firebase as his database. Firebase uses a no-SQL database that instead of operating with a relational data model, it uses a document data model. Within relational data models, an object can be defined once and used across several different entities. Whereas within document data models, there is only a single instance of an object defined within an entity and each of these are unique and not reused. The initial planning that Owen carried out in his original ER Diagram still helped significantly even though he opted for a no-SQL option. This ER diagram assisted Owen later when structuring his database in terms of understanding the entities required for the application as well as the attributes and data types required for recording data.

6. Implementation & Testing

6.1.1 Requirement 1.1 : Add visitor data to the database

Acceptance Criteria: “The childcare provider can log visitor information which is stored in a database.”

Development

Upon pressing the ‘Log Visitor’ button, the user’s submitted input will be validated before writing their information to a Firestore database table named ‘visitorLogs’.

LogVisitor.js

```
// Initialising connection to visitorLogs database table
const fireDB = app.firestore().collection("visitorLogs");
```

```
await fireDB.add({
  visitor_name: visitorName,
  date_of_visit: dateOfVisit.date,
  time_in: convertTime(timeIn.date),
  time_out: convertTime(timeOut.date),
  visit_purpose: visitPurpose,
});
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addVisitorLog()}>
  <Text style={styles.buttonTextMenu}>Log Visitor</Text>
```

```
</Button>
```

Solution

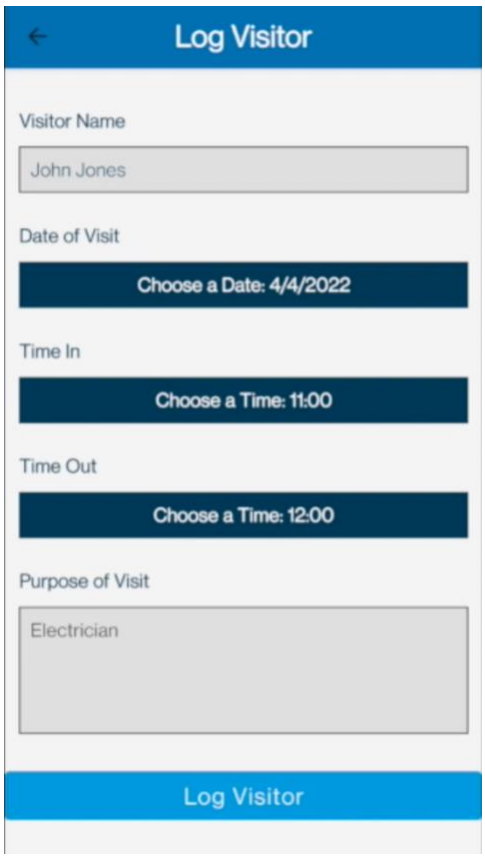


Figure 6.1: Log visitor page

```
date_of_visit: April 4, 2022 at 12:16:36 PM UTC+1
time_in: "11:00"
time_out: "12:00"
visit_purpose: "Electrician"
visitor_name: "John Jones"
```

Figure 6.2: Visitor log database entry

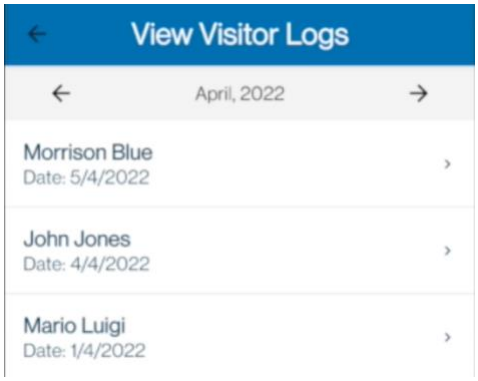


Figure 6.3: Visitor information logged to application successfully

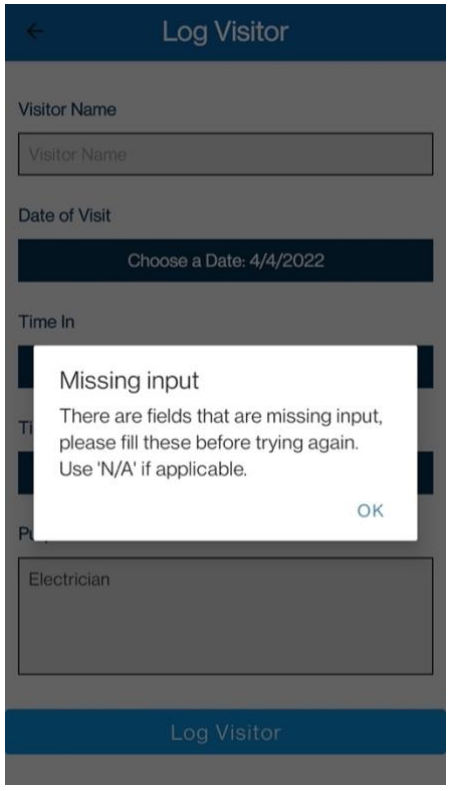


Figure 6.4: Visitor log input validation

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
1	Visitor information is logged successfully	Pass	
2	Details are stored in Firestore database	Pass	
3	Data validity is checked when submitted	Pass	

Table 6.1: Unit Testing for Requirement 1.1

6.1.2 Requirement 1.2 : Add medicine administration data to the database

Acceptance Criteria: “The childcare provider can log medicine administration information which is stored in a database.”

Development

Upon pressing the ‘Log Medicine Administration’ button, the user’s submitted input will be validated before writing their information to a Firestore database table named ‘medicineAdministration’.

LogMedicine.js

```
// Initialising connection to medicineAdministration database table
const fireDB = app.firestore().collection("medicineAdministration");

await fireDB.add({
  child_name: childName,
  medicine_title: medicineTitle,
  medicine_date: medicineDate.date,
  medicine_time: convertTime(medicineTime.date),
  medicine_reason: medicineReason,
  medicine_notes: medicineNotes,
});
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addLog()}>
  <Text style={styles.buttonTextMenu}>
    Log Medicine Administration
  </Text>
</Button>
```

Solution

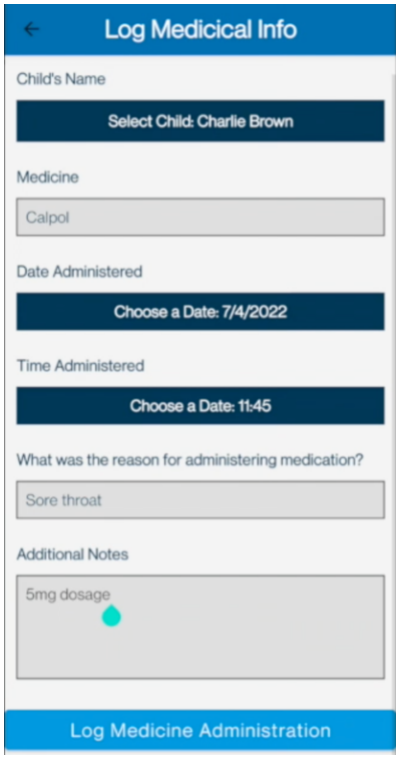


Figure 6.5: Log medical info page

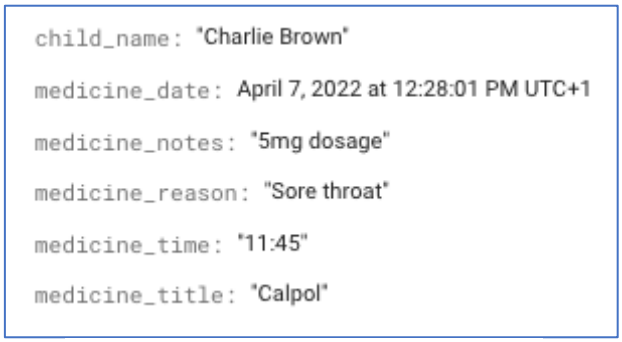


Figure 6.6: Medical log database entry

Figure 6.7: Medical information logged to application successfully

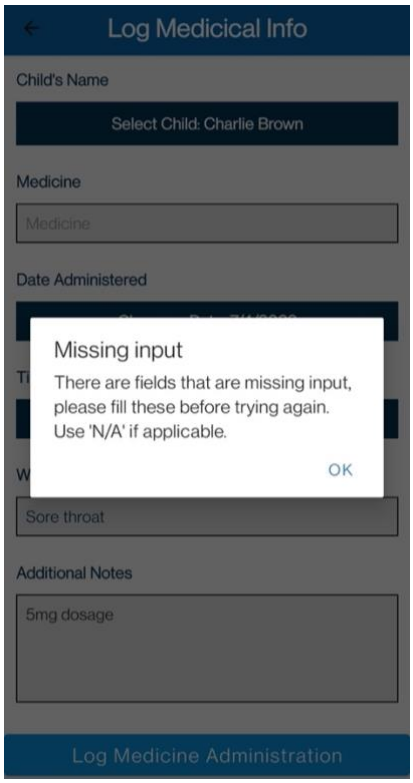


Figure 6.8: Medical log input validation

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
4	Medicine administration information is logged successfully	Pass	

5	Details are stored in Firestore database	Pass	
6	Data validity is checked when submitted	Pass	

Table 6.2: Unit Testing for Requirement 1.2

6.1.3 Requirement 1.3 : Add attendance data to the database

Acceptance Criteria: “The childcare provider can log daily attendance information which is stored in a database.”

Development

Upon pressing the ‘Log Attendance’ button, the user’s submitted input will be validated before writing their information to a Firestore database table named ‘attendanceRegister’.

AttendanceRegister.js

```
// Initialising connection to attendanceRegister database table
const fireDB = app.firestore().collection("attendanceRegister");
```

```
await fireDB.add({
  additional_notes: additionalNotes,
  check_in_time: convertTime(checkInTime.date),
  check_out_time: convertTime(checkOutTime.date),
  child_name: childName,
  collected_by: collectedBy,
  date_of_attendance: convertDate(dateOfAttendance.date),
  dropped_by: droppedBy,
  temperature_checked: temperatureChecked,
});
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addAttendanceLog()}>
  <Text style={styles.buttonTextMenu}>Log Attendance</Text>
</Button>
```

Solution

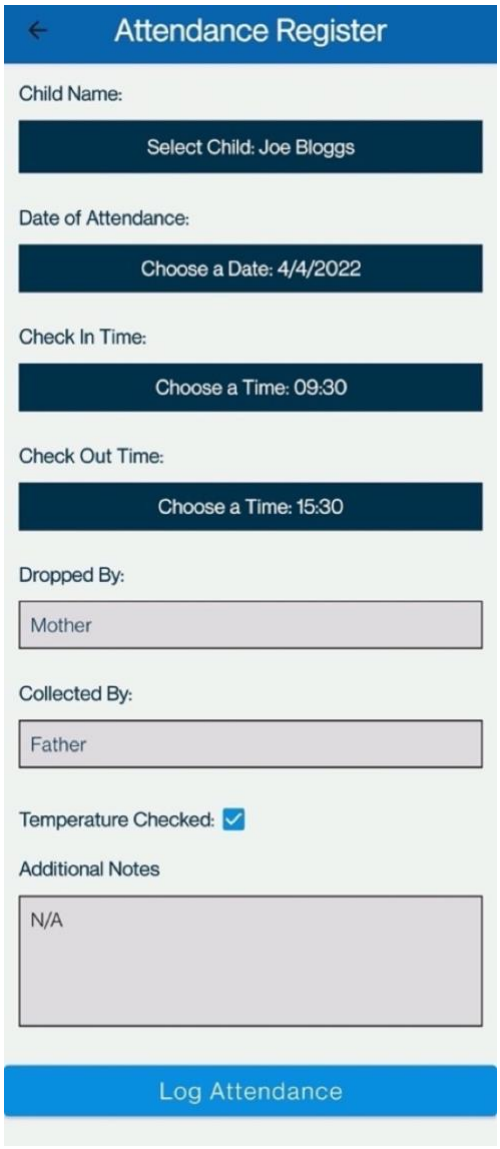


Figure 6.9: Attendance register page

```
additional_notes: "N/A"
check_in_time: "09:30"
check_out_time: "15:30"
child_name: "Joe Bloggs"
collected_by: "Father"
date_of_attendance: "4/4/2022"
dropped_by: "Mother"
temperature_checked: true
```

Figure 6.10: Attendance log database entry

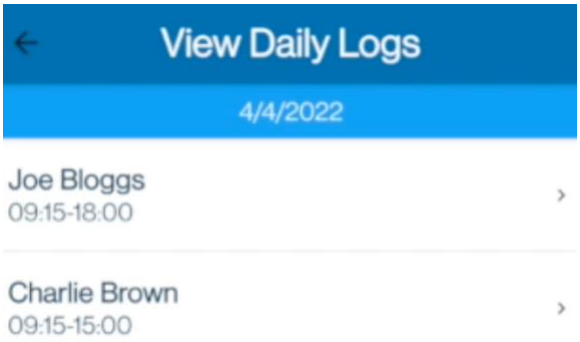


Figure 6.11: Attendance logged to application successfully

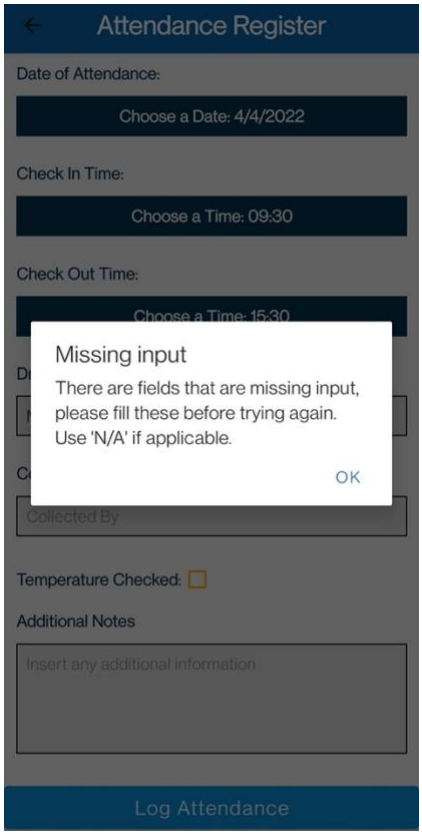


Figure 6.12: Attendance register input validation

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
7	Daily attendance information is logged successfully	Pass	
8	Details are stored in Firestore database	Pass	
9	Data validity is checked when submitted	Pass	

Table 6.3: Unit Testing for Requirement 1.3

6.1.4 Requirement 1.4 : Add accident report data to the database

Acceptance Criteria: “The childcare provider can log accident report information which is stored in a database.”

Development

Upon pressing the ‘Log Accident Report’ button, the user’s submitted input will be validated before writing their information to a Firestore database table named ‘accidentReports’.

LogAccidentReport.js

```
// Initialising connection to accidentReports database table
const fireDB = app.firestore().collection("accidentReports");
```

```
await fireDB.add({
```

```
  child_name: childName,
  accident_date: dateOfAccident.date,
  accident_time: convertTime(timeOfAccident.date),
  accident_notes: accidentNotes,
  accident_location: accidentLocation,
  accident_detail: accidentDetail,
  accident_action: accidentAction,
  accident_medical_attention: accidentMedicalAttention,
});
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addAccidentReport()}>
  <Text style={styles.buttonTextMenu}>Log Accident Report</Text>
</Button>
```

Solution

Figure 6.13: Log accident report page

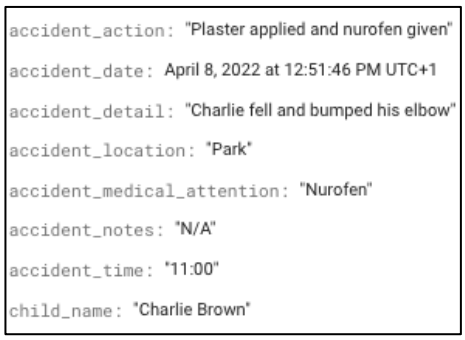


Figure 6.14: Accident report database entry

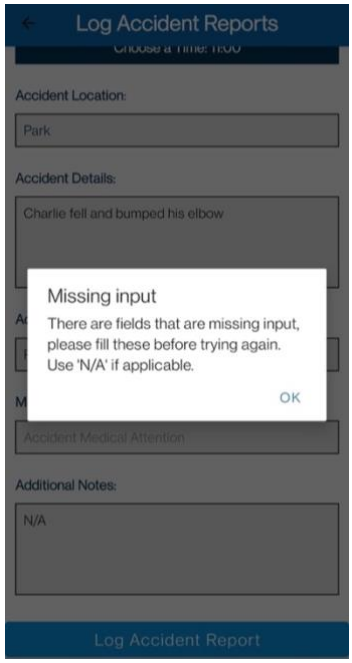


Figure 6.15: Accident report input validation

Figure 6.16: Accident report logged to application successfully

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
10	Accident report is logged successfully	Pass	
11	Details are stored in Firestore database	Pass	
12	Data validity is checked when submitted	Pass	

Table 6.4: Unit Testing for Requirement 1.4

6.1.5 Requirement 1.6, 1.7 : View & update data that has been stored in the database

Acceptance Criteria: "The childcare provider can view the information which they have previously logged."

Acceptance Criteria: "The childcare provider can edit the information which they have previously logged. These changes are implemented on a database."

Development

Upon selecting an existing log that was previously submitted on the application, the user can view its details and update these through providing new inputs.

ViewLogDetails.js

```
// Query the database to gather attendance log data, using userkey as
an identifier
const documentReference = app
  .firestore()
  .collection("attendanceRegister")
  .doc(this.props.route.params.userkey);
// Once the database query has retrieved results, assign them to state
variable values
documentReference.get().then((result) => {
  if (result.exists) {
    const log = result.data();
    this.setState({
      key: result.id,
      childName: log.child_name,
      dateOfAttendance: log.date_of_attendance,
```

```
      checkInTime: log.check_in_time,
      checkOutTime: log.check_out_time,
      droppedBy: log.dropped_by,
      collectedBy: log.collected_by,
      temperatureChecked: log.temperature_checked,
      additionalNotes: log.additional_notes,
    });
  } else {
    console.log("No document found.");
  }
});
```

```
<Text style={styles.bold}>Child Name</Text>
    <View>
      {/* ModalSelector populated with children names from
       childNameArr */}
      <ModalSelector
        style={styles.dropdown}
        data={this.state.childNames}
        onChange={(option) => {
          this.setState({ childName: option.label });
        }}>
      <Text style={styles.dropdownText}>
        Select Child: {this.state.childName}
      </Text>
      </ModalSelector>
    </View>
    <Text style={styles.bold}>Date of Attendance</Text>
    <View>
      <TouchableOpacity style={styles.button} onPress={() =>
        this.showDatepicker()}>
        {this.state.show && (
          <DateTimePicker
            maximumDate={new Date()}
            value={this.state.date}
            mode="date"
            onChange={this.onDateChange}
          />
        )}
        <Text style={styles.buttonText}>
          Choose a Date: {this.state.dateOfAttendance}
        </Text>
      </TouchableOpacity>
    </View>
    <Text style={styles.bold}>Check-in Time</Text>
    <TextInput
      style={styles.input}
      placeholder={"00:00"}
      value={this.state.checkInTime}
      onChangeText={(value) => this.updateStateValue(value,
        "checkInTime")}
```

```
    />
    <Text style={styles.bold}>Check-out Time</Text>
    <TextInput
      style={styles.input}
      placeholder={"00:00"}
      value={this.state.checkOutTime}
      onChangeText={(value) => this.updateStateValue(value,
        "checkOutTime")}
    />
    <Text style={styles.bold}>Dropped By</Text>
    <TextInput
      style={styles.input}
      placeholder={"Dropped By"}
      value={this.state.droppedBy}
      onChangeText={(value) => this.updateStateValue(value,
        "droppedBy")}
    />
    <Text style={styles.bold}>Collected By</Text>
    <TextInput
      style={styles.input}
      placeholder={"Collected By"}
      value={this.state.collectedBy}
      onChangeText={(value) => this.updateStateValue(value,
        "collectedBy")}
    />
    <View style={styles.checkBoxPositioning}>
      <Text style={styles.bold}>Temperature Checked:</Text>
      <CheckBox
        style={styles.checkBox}
        disabled={false}
        value={this.state.temperatureChecked}
        onValueChange={(value) => this.updateStateValue(value,
          "temperatureChecked")}
        tintColors={{ true: "#0B8FDC", false: "orange" }}
      />
    </View>
    <Text style={styles.boldTextCheckbox}>Additional Notes</Text>
    <TextInput
      style={styles.extendedInput}
      multiline={true}
      numberOfLines={4}
      placeholder={"Insert any additional information"}
      value={this.state.additionalNotes}
      onChangeText={(value) => this.updateStateValue(value,
        "additionalNotes")}
    />
```

```
// Set the state variable value to the value supplied from the input
  updateStateValue = (value, prop) => {
    const state = this.state;
    state[prop] = value;
```

```
    this.setState(state);
  };
```

```
const documentUpdate =
app.firestore().collection("attendanceRegister").doc(this.state.key);
      documentUpdate
        .set({
          child_name: this.state.childName,
          date_of_attendance: this.state.dateOfAttendance,
          check_in_time: this.state.checkInTime,
          check_out_time: this.state.checkOutTime,
          dropped_by: this.state.droppedBy,
          collected_by: this.state.collectedBy,
          temperature_checked: this.state.temperatureChecked,
          additional_notes: this.state.additionalNotes,
  })
```

```
<Button
    mode="contained"
    uppercase={false}
    color="#0B8FDC"
    onPress={() => this.editAttendanceLog()}>
    <Text style={styles.buttonTextMenu}>Update</Text>\
</Button>
```

Solution

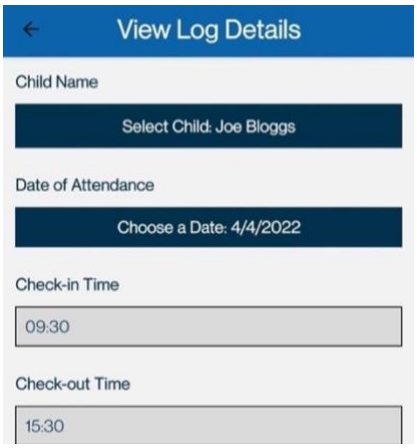


Figure 6.17: Existing log can be viewed

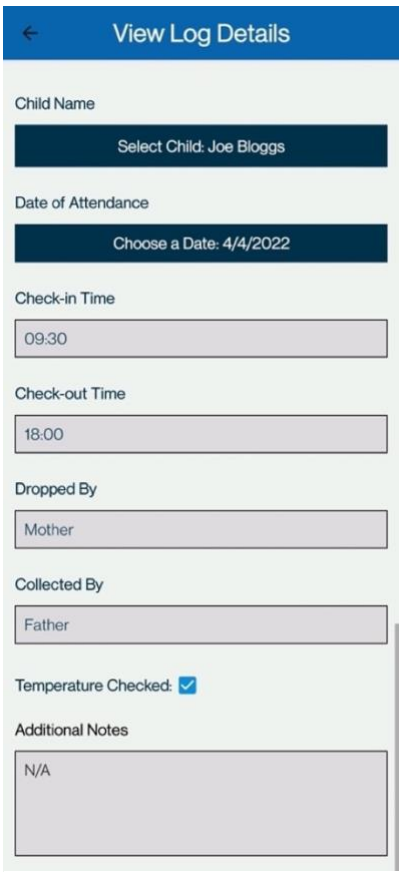


Figure 6.18: Check-out time is updated

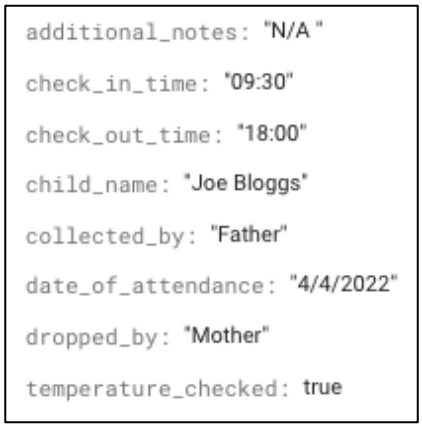


Figure 6.19: Database entry has check_out_time value updated

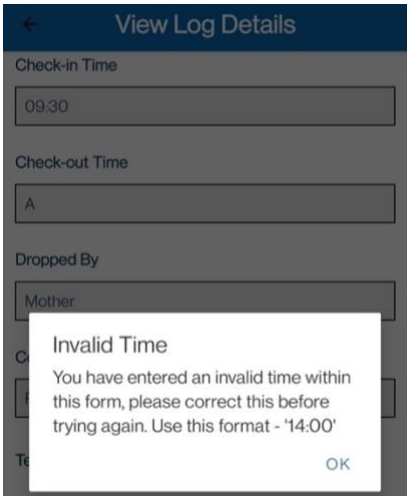


Figure 6.20: Validation rejects invalid non-numerical input for check-out time

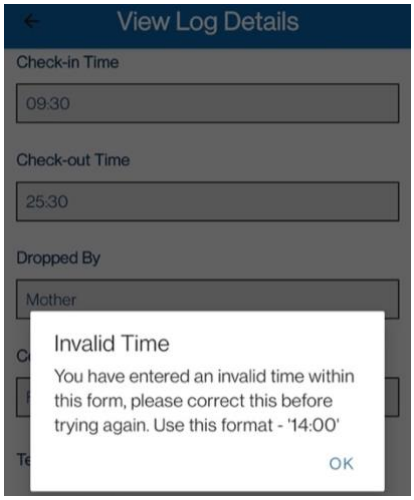


Figure 6.21: Validation rejects invalid time input for check-out time

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
13	Previously entered information can be viewed	Pass	
14	Attendance information that was previously logged can be updated in the application.	Pass	
15	Changes are updated on Firestore database.	Pass	
16	Data validity is checked when submitted.	Fail	Validation is used for time entries to ensure that the input is at least 4 characters in length. However, there are some edge cases that this can be exposed. For example, '14:3' would be accepted as valid. Owen aims to address this in a future development by adding an additional layer of validation.

Table 6.5: Unit Testing for Requirement 1.6, 1.7

6.1.6 Requirement 1.8 : Filtering data

Acceptance Criteria: “The childcare provider can use filters to reduce the results which are returned by the application.”

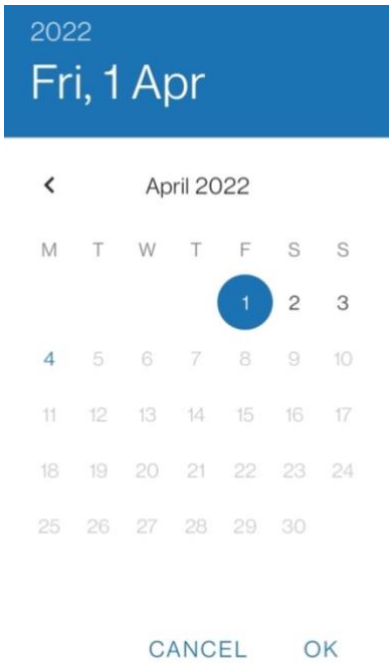
Development

Logs can be filtered within the application through using datepickers, monthpickers and modal selectors to specify the results that should be returned.

ViewLogDetails.js

```
{this.state.show && (
  <DateTimePicker
    maximumDate={new Date()}
    value={this.state.date}
    mode="date"
    is24Hour={true}
    onChange={onChange}
  />
)}
```

Solution



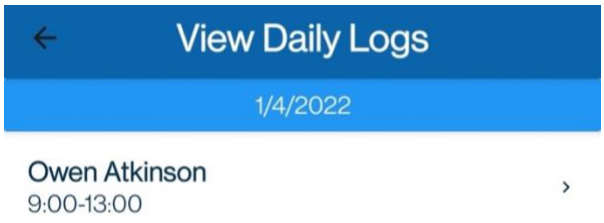


Figure 6.23: Logs only returned for specified date

Figure 6.22: Date picker component

Development

MonthPick.tsx

```
const MonthPick: React.FC<MonthPickerProps> = ({ date, onChange }) =>
{
  // Reduce date by 1 month
```

```
  const handlePrev = () => {
    const newDate = subMonths(date, 1);
    onChange(newDate);
  };

  // Increase date by 1 month
  const handleNext = () => {
    const newDate = addMonths(date, 1);
    onChange(newDate);
  };

  return (
    <View style={styles.row}>
      <IconButton icon="arrow-left" onPress={handlePrev}></IconButton>
      <Text>{format(date, "MMMM, yyyy")}</Text>
      <IconButton icon="arrow-right" onPress={handleNext}>
      </IconButton>
    </View>
  );
};
```

ViewVisitorLogs.js

```
{/* MonthPick used to filter invoice logs into a month & year */}
<SafeAreaView edges={["bottom", "left", "right"]}>
  <FlatList
    ListHeaderComponent={
      <MonthPick
        style={styles.navyStandardText}
        date={this.state.date}
        onChange={(newDate) => {
          this.setState({ date: newDate });
        }}
      />
    }
  />
</SafeAreaView>
```

```
// Only display vistor logs that match the year and month values of
the MonthPick component
if (
  doNumbersMatch(
    getMonth(convertDateCheckType(this.state.date)),
    getMonth(convertDateCheckType(result.date_of_visit))
  ) &&
  doNumbersMatch(
    getYear(convertDateCheckType(this.state.date)),
    getYear(convertDateCheckType(result.date_of_visit))
  )
)
```

Solution

Figure 6.24: Month picker displaying logs from 'March, 2022'

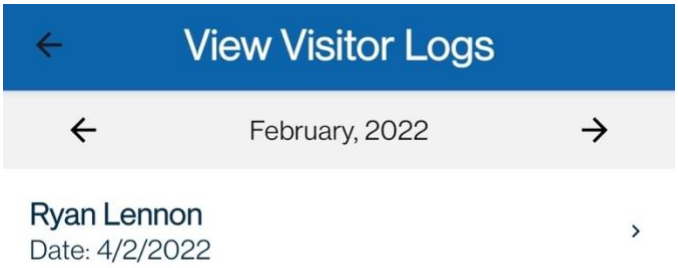


Figure 6.25: From 'March, 2022', pressing the left arrow symbol navigates the user to the previous month

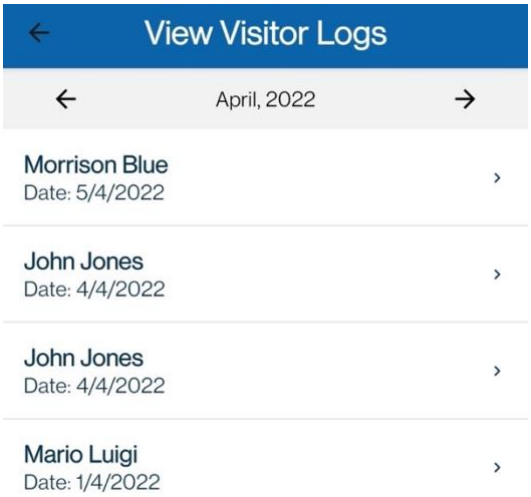


Figure 6.26: From 'March, 2022', pressing the right arrow symbol navigates the user to the next month

Development

ViewMedicalInfo.js

```
// Query the database to gather names of all children and store these
names in childNames array
    const childNames = [];
    let index = 0;
    app
      .firestore()
      .collection("children")
      .orderBy("child_name", "asc")
      .get()
      .then((querySnapshot) => {
```

```
        querySnapshot.forEach((document) => {
          childNames.push({
            key: index++,
            label: document.data()["child_name"],
          });
        });
        this.setState({
          childNames: childNames,
        });
      });
```

```
{/* ModalSelector populated with children names from childNameArr */}
<ModalSelector
 style={styles.navyText}
 data={this.state.childNames}
 initValue="Select Child"
 onChange={(option) => {
  this.setState({ activeChildName: option.label });
 }}
/>
```

Solution

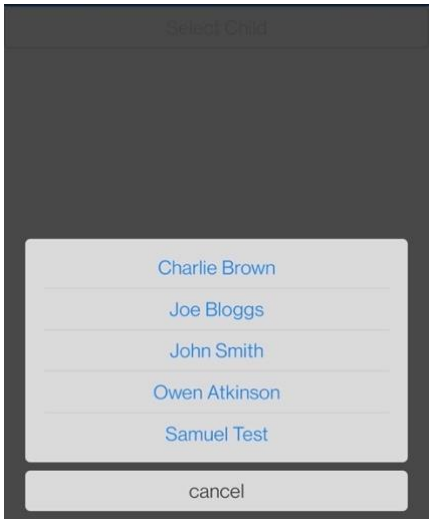


Figure 6.27: Modal selector component is populated with children that are marked as actively in care

Figure 6.28: Modal selector filters results to the specified child's name

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
17	Date picker filters results as expected	Pass	
18	Month picker filters results as expected	Pass	
19	Modal selector filters results as expected	Pass	

Table 6.6: Unit Testing for Requirement 1.8

6.1.7 Requirement 1.9 : Delete data from the database

Acceptance Criteria: “The childcare provider can delete information which has been submitted previously by them, this removes this information from a database.”

Development

ViewLogDetails.js

```
// Delete the attendance log from the database, using userkey as an
identifier & navigate the user back to the ViewLogs page
  deleteAttendanceLog() {
    const documentReference = app
      .firestore()
      .collection("attendanceRegister")
      .doc(this.props.route.params.userkey);

    documentReference.delete().then(() => {
      this.props.navigation.navigate("ViewLogs");
    });
  }
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#EE752E"
  onPress={this.alertDialog}>
  <Text style={styles.buttonTextMenu}>Delete</Text>
</Button>
```

```
// Display alert to confirm if the user wants to delete the item from
the database
  alertDialog = () => {
    Alert.alert(
      "Delete",
      "Really?",
      [
        { text: "Yes", onPress: () => this.deleteAttendanceLog() },
        { text: "No", onPress: () => console.log("Item not deleted"),
          style: "cancel" },
      ],
      {
        cancelable: true,
      }
    );
  };
```

Solution

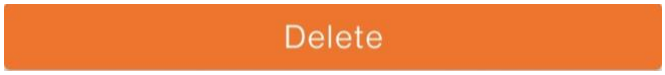


Figure 6.29: Delete button component

Figure 6.30: Alert modal is presented to confirm the delete action

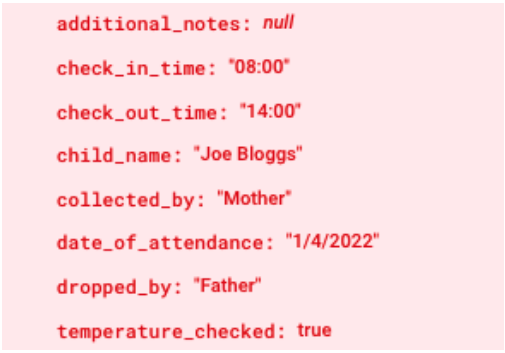


Figure 6.31: Data entry removed from Firestore database

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
20	Logs can be deleted in the application	Pass	
21	Data is removed from Firestore database	Pass	

Table 6.7: Unit Testing for Requirement 1.9

6.1.8 Requirement 2.1 : Upload policy documents

Development

Acceptance Criteria: “The childcare provider can upload documents to a private and secure database through the application.”

Policies.js

```
async function uploadPolicy() {
    // Launch document picker
    let file = await DocumentPicker.getDocumentAsync({});
    // Use a blob to store XML information regarding the document
being uploaded
    if (file.uri != null) {
      const blob = await new Promise((resolve, reject) => {
        const xml = new XMLHttpRequest();
        xml.onload = function () {
          resolve(xml.response);
        };
        xml.onerror = function () {
```

```
          reject(new TypeError("Network request failed"));
        };
        xml.responseType = "blob";
        xml.open("GET", file.uri, true);
        xml.send(null);
      });

      // Provide firebase storage location to store the document
      const ref =
app.storage().ref(`/policies/${removeFileExtension(file.name)}`);
      const snapshot = ref.put(blob);

      snapshot.on(
        "state_changed",
        // Next callback monitors the progress of the document upload
to firebase storage
        function (snapshot) {
          let progress = (snapshot.bytesTransferred /
snapshot.totalBytes);
          setProgress(progress);
          if(progress == 1) {
            setProgress(-1);
          }
        },
        // Error callback used to capture error
        function (error) {
          console.log(error);
          blob.close();
          return;
        },
        // Complete callback used save the document to firebase
realtime database and navigate to the FilePreview page, passing the
downloadURL as fileData.fileURL variable
        function () {
          snapshot.snapshot.ref.getDownloadURL().then(function
(downloadURL) {
            saveFileToRealtimeDatabase(downloadURL, file);
            props.navigation.navigate("FilePreview", {
              fileData: { fileURL: downloadURL}
            })
          });
          blob.close();
          return;
        }
      );
    }
  }
```

```
// Saving to firebase realtime database is required to acquire a URL
for opening & sharing the document
  function saveFileToRealtimeDatabase(downloadURL, file) {
    let trimFileName = removeFileExtension(file.name);
```

```
    app.database().ref(`policies/${trimFileName}`).update({
      fileName: file.name,
      fileURL: downloadURL,
    });
  }
```

```
// Trim the filename into an appropriate format to be stored on
firebase realtime database
  function removeFileExtension(fileWithExtension) {
    return fileWithExtension.replace(/\.[^/.]+$/, "");
  }
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={uploadPolicy}>
  <Text style={styles.buttonTextMenu}>Upload Policy</Text>
</Button>
```

Solution

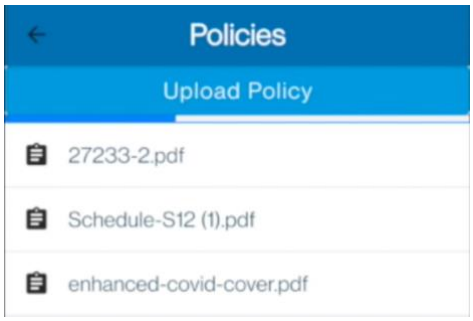


Figure 6.32: Progress bar is displayed when uploading a document

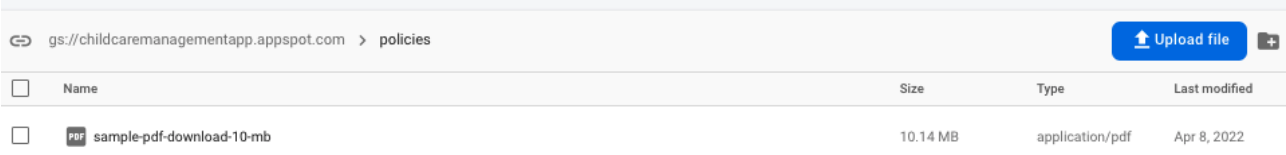


Figure 6.33: Policy uploaded to Firebase storage container

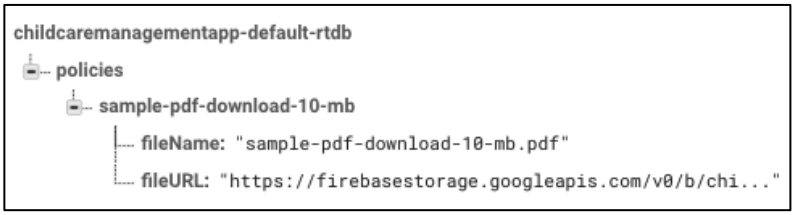


Figure 6.34: Policy details uploaded to Firebase Realtime database

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)

22	Progress bar is displayed when uploading and behaves as expected	Pass	
23	Document is uploaded to Firebase storage container	Pass	
24	Document details uploaded to Firebase Realtime database	Pass	

Table 6.8: Unit Testing for Requirement 2.1

6.1.9 Requirement 2.2 : Delete policy documents

Development

Acceptance Criteria: “The childcare provider can delete policies within the application which removes them from a database.”

Policies.js

```
function deletePolicy(fileName) {
    let deletePolicyDatabase = app.database().ref("policies/" +
removeFileExtension(fileName));
    var storage = app.storage();
    var storageRef = storage.ref();
    var policyRef =
storageRef.child(`policies/${removeFileExtension(fileName)}`);
    // Delete the policy from the firebase realtime database
    deletePolicyDatabase.remove();
    // Delete the policy from the firebase storage container
    policyRef.delete();
    setFileList([]);
    // Refresh the list of policies
    const onChildDeleted = app
      .database()
      .ref(`policies`)
      .on("child_added", (snapshot) => {
        let helperArr = [];
        helperArr.push(snapshot.val());
        setFileList((files) => [...files, ...helperArr]);
      });
    return () => app.database().ref(`policies`).off("child_added",
onChildDeleted);
  }
```

```
// Display alert to confirm if the user wants to delete the policy
from the system
  const alertDialog = (file) => {
    Alert.alert(
      "Delete Policy",
      "Really?",
      [
        { text: "Yes", onPress: () => deletePolicy(file) },
        {
```

```
          text: "No",
          onPress: () => console.log("Item not deleted"),
          style: "cancel",
        },
      ],
      {
        cancelable: true,
      }
    );
  };
```

```
// Swipeable component that reveals an option to delete the policy
  rightContent={
    <FontAwesome.Button
      name="trash"
      backgroundColor="#ee752e"
      alignItems="center"
      justifyContent="center"
      style={styles.swipeableItem}
      alertDialog
      onPress={() => alertDialog(item.fileName)}
    ></FontAwesome.Button>
  }
```

Solution

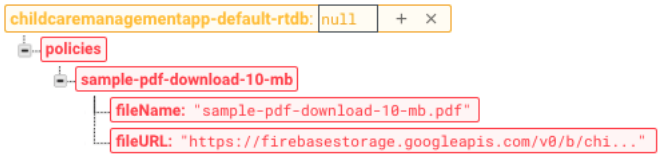


Figure 6.35: Policy details removed from Firebase Realtime database

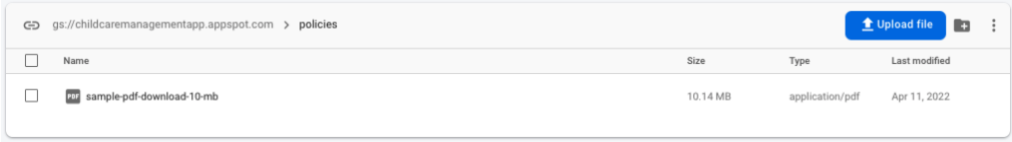


Figure 6.36: Firestore storage container before policy is removed

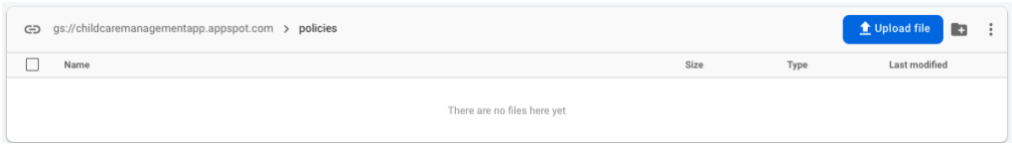


Figure 6.37: Firestore storage container after policy is removed

Figure 6.38: Policy list with previously uploaded document

Figure 6.39: When delete button is pressed, a modal appears

Figure 6.40: Confirming the delete will remove the document from the policy list

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
25	Document is removed from Firebase storage container	Pass	
26	Document is removed from Firebase Realtime database	Pass	
27	Document is removed from policies list	Pass	There seems to be a bug in the scenario within an edge case where a policy is deleted and then immediately after, a policy is uploaded. This results in a duplicate policy being displayed on the list, but this is corrected when the page is refreshed

Table 6.9: Unit Testing for Requirement 2.2

6.1.10 Requirement 2.3 : View policy documents

Acceptance Criteria: “The childcare provider can view their policies through a PDF viewer, allowing them to have features such as navigate and zoom when viewing the document.”

Development

Policies.js

```
key={index}
onPress={() =>
```

```
// Pressing on the file's ListItem component will navigate the user to
the FilePreview page and display the selected policy through passing
the fileData variable
  props.navigation.navigate("FilePreview", {
    fileData: item,
  })
}
```

FilePreview.js

```
// Uses the route fileData variable passed from Policies page
const FilePreview = ({ route }) => {
  return (
// Displays the PDF using PDFReader component, supplying it with a uri
    <PDFReader
      withPinchZoom
      source={{
        uri: route.params.fileData.fileURL,
      }}
    />
  );
};
```

Solution

Figure 6.41: Policy can be opened and viewed

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
28	Policy can be viewed within the application	Pass	

Table 6.10: Unit Testing for Requirement 2.3

6.1.11 Requirement 2.4 : Share policy documents

Acceptance Criteria: "The childcare provider has an option for sharing the document externally."

Development

```
// Using Share to provide a feature that opens the device's sharing
component
  const sharePolicy = async (fileShareableURL) => {
    try {
      const result = await Share.share({
        message: fileShareableURL,
      });
    } catch (error) {
      console.log(error);
    }
  };
```

```
// Swipeable component that reveals an option to share the policy
  leftContent={
    <FontAwesome.Button
      name="share-alt"
      backgroundColor="#0b8fdc"
      alignItems="center"
      justifyContent="center"
      style={styles.swipeableItem}
      onPress={() => sharePolicy(item.fileURL)}
    ></FontAwesome.Button>
  }
```

Solution

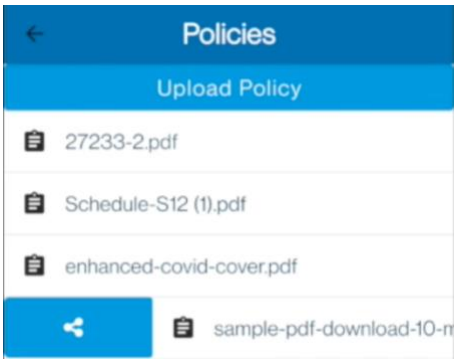


Figure 6.42: Swipe gesture reveals share button

Figure 6.44: URL shared to access resource

Figure 6.43: Native share functionality is launched

Figure 6.45: URL will download the document to a device

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
29	Policy can be shared from the application	Pass	
30	Shareable link can be used to view the policy	Pass	

Table 6.11: Unit Testing for Requirement 2.4

6.1.12 Requirement 3.1, 3.2, 3.3, 3.5: Add child data to the database

Acceptance Criteria: "The childcare provider can add children into the application through submitting details which are stored in a database."

Development

AddChildDetails.js

```
// Initialising connection to children database table
const fireDB = app.firestore().collection("children");
```

```
await fireDB.add({
  child_name: childName,
  child_DOB: convertDate(childDOB.date),
  child_allergies: childAllergies,
  child_allergies_details: childAllergiesDetails,
  child_medical_conditions: childMedicalConditions,
  child_medical_conditions_details: childMedicalConditionsDetails,
  child_is_active: childIsActive,
  child_emergency_contact_name_1: childEmergencyContactName1,
  child_emergency_contact_number_1: childEmergencyNumber1,
```

```
  child_emergency_contact_relation_1: childEmergencyRelation1,
  child_emergency_contact_name_2: childEmergencyContactName2,
  child_emergency_contact_number_2: childEmergencyNumber2,
  child_emergency_contact_relation_2: childEmergencyRelation2,
  child_emergency_contact_name_3: childEmergencyContactName3,
  child_emergency_contact_number_3: childEmergencyNumber3,
  child_emergency_contact_relation_3: childEmergencyRelation3,
  doctor_name: doctorName,
  doctor_address: doctorAddress,
  doctor_number: doctorNumber,
  child_home_address: childHomeAddress
});
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addChild()}>
  <Text style={styles.buttonTextMenu}>Add Child</Text>
</Button>
```

Solution

Figure 6.46: Add new child page

Figure 6.47: Child information is added to a Firestore database

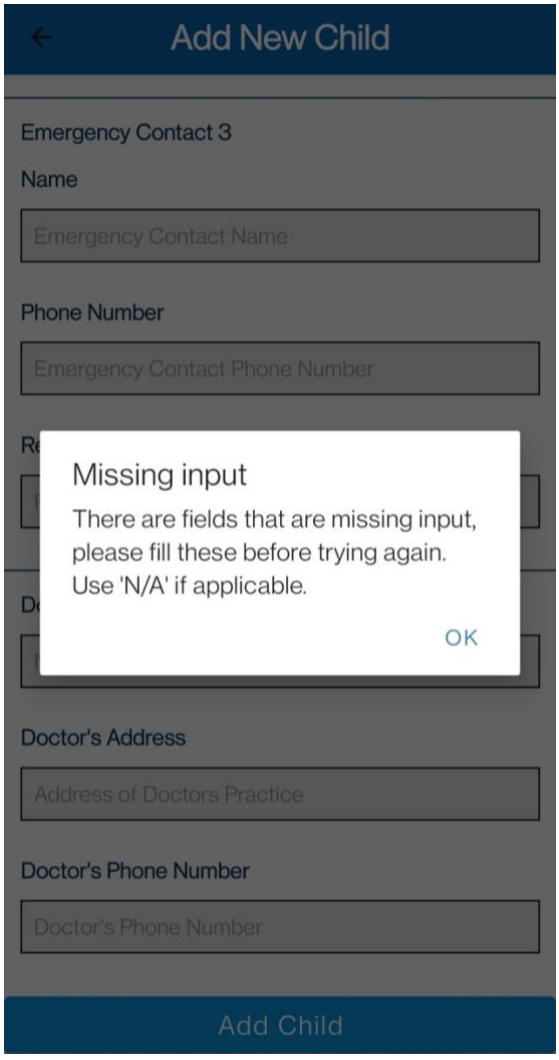


Figure 6.48: Input validation checking for blank inputs

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
31	Childcare provider can add children into the application through submitting details	Pass	
32	Details are stored in a Firestore database	Pass	
33	Data validity is checked when submitted	Pass	

Table 6.12: Unit Testing for Requirements 3.1, 3.2, 3.3, 3.5

6.1.13 Requirement 3.4, 3.7: View & update child data to the database

Acceptance Criteria: "The childcare provider can update information of children in their care, these changes are implemented on a database."

Acceptance Criteria: "The childcare provider can view the child details information which they have previously logged."

Development

UpdateChildDetails.js

```
// Query the database to gather child data, using userkey as an
identifier
    const documentReference = app
      .firestore()
      .collection("children")
      .doc(this.props.route.params.userkey);
    // Once the database query has retrieved results, assign them to
state variable values
    documentReference.get().then((result) => {
      if (result.exists) {
        const data = result.data();
        this.setState({
          key: result.id,
          name: data.child_name,
          dob: data.child_DOB,
          allergies: data.child_allergies,
          allergiesDetails: data.child_allergies_details,
          medicalConditions: data.child_medical_conditions,
          medicalConditionsDetails:
data.child_medical_conditions_details,
          isActive: data.child_is_active,
          emergencyName1: data.child_emergency_contact_name_1,
          emergencyNumber1: data.child_emergency_contact_number_1,
          emergencyRelation1: data.child_emergency_contact_relation_1,
          emergencyName2: data.child_emergency_contact_name_2,
          emergencyNumber2: data.child_emergency_contact_number_2,
          emergencyRelation2: data.child_emergency_contact_relation_2,
          emergencyName3: data.child_emergency_contact_name_3,
          emergencyNumber3: data.child_emergency_contact_number_3,
          emergencyRelation3: data.child_emergency_contact_relation_3,
```

```
          doctorName: data.doctor_name,
          doctorAddress: data.doctor_address,
          doctorNumber: data.doctor_number,
          childAddress: data.child_home_address,
        });
      } else {
        console.log("No document found.");
      }
    });
```

```
<Text style={styles.bold}>Child's Name: {this.state.name}</Text>
        <View style={styles.space}></View>
        <Text style={styles.bold}>Child's Date of Birth</Text>
        <View>
          <TouchableOpacity style={styles.button} onPress={() =>
this.showDatepicker()}>
            {this.state.show && (
              <DateTimePicker
                maximumDate={new Date()}
                value={this.state.date}
                mode="date"
                onChange={this.onDateChange}
              />
            )}
            <Text style={styles.buttonText}>Choose a Date:
{this.state.dob}</Text>
          </TouchableOpacity>
        </View>
        <Text style={styles.bold}>Child's Allergies</Text>
        <TextInput
          style={styles.input}
          placeholder={"List Child's Allergies"}
          value={this.state.allergies}
          onChangeText={(value) => this.updateStateValue(value,
"allergies")}
        />
        <Text style={styles.bold}>Allergy Details</Text>
        <TextInput
          multiline={true}
          numberOfLines={4}
          style={styles.extendedInput}
          placeholder={"Insert details of the child's allergies"}
          value={this.state.allergiesDetails}
          onChangeText={(value) => this.updateStateValue(value,
"allergiesDetails")}
        />
        <Text style={styles.bold}>Child's Medical Conditions</Text>
        <TextInput
          style={styles.input}
          placeholder={"List Child's Medical Conditions"}
          value={this.state.medicalConditions}
```

```
          onChangeText={(value) => this.updateStateValue(value,
"medicalConditions")}
        />
        <Text style={styles.bold}>Medical Condition Details</Text>
        <TextInput
          multiline={true}
          numberOfLines={4}
          style={styles.extendedInput}
          placeholder={"Insert details of the child's medical
conditions"}
          value={this.state.medicalConditionsDetails}
          onChangeText={(value) => this.updateStateValue(value,
"medicalConditionsDetails")}
        />
        <View style={styles.checkBoxPositioning}>
          <Text style={styles.bold}>Child is actively under your
care:</Text>
          <CheckBox
            style={styles.checkBox}
            disabled={false}
            value={this.state.isActive}
            onValueChange={(value) => this.updateStateValue(value,
"isActive")}
            tintColors={{ true: "#0B8FDC", false: "orange" }}
          />
        </View>
        <Text style={styles.boldTextCheckbox}>Child's Home
Address</Text>
        <TextInput
          style={styles.input}
          placeholder={"Child Home Address"}
          value={this.state.childAddress}
          onChangeText={(value) => this.updateStateValue(value,
"childAddress")}
        />
        <View style={styles.horizontalRule}></View>
        <Text style={styles.bold}>Emergency Contact 1</Text>
        <Text style={styles.bold}>Name</Text>
        <TextInput
          style={styles.input}
          placeholder={"Emergency Contact Name"}
          value={this.state.emergencyName1}
          onChangeText={(value) => this.updateStateValue(value,
"emergencyName1")}
        />
        <Text style={styles.bold}>Phone Number</Text>
        <TextInput
          style={styles.input}
          placeholder={"Emergency Contact Number"}
          value={this.state.emergencyNumber1}
```

```
          onChangeText={(value) => this.updateStateValue(value,
"emergencyNumber1")}
        />
        <Text style={styles.bold}>Emergency Contact Relation</Text>
        <TextInput
          style={styles.input}
          placeholder={"Emergency Contact Relation"}
          value={this.state.emergencyRelation1}
          onChangeText={(value) => this.updateStateValue(value,
"emergencyRelation1")}
        />
        <View style={styles.horizontalRule}></View>
        <Text style={styles.bold}>Emergency Contact 2</Text>
        <Text style={styles.bold}>Name</Text>
        <TextInput
          style={styles.input}
          placeholder={"Parent #2 Name"}
          value={this.state.emergencyName2}
          onChangeText={(value) => this.updateStateValue(value,
"emergencyName2")}
        />
        <Text style={styles.bold}>Phone Number</Text>
        <TextInput
          style={styles.input}
          placeholder={"Parent #2 Number"}
          value={this.state.emergencyNumber2}
          onChangeText={(value) => this.updateStateValue(value,
"emergencyNumber2")}
        />
        <Text style={styles.bold}>Relation</Text>
        <TextInput
          style={styles.input}
          placeholder={"Emergency Contact Relation"}
          value={this.state.emergencyRelation2}
          onChangeText={(value) => this.updateStateValue(value,
"emergencyRelation2")}
        />
        <View style={styles.horizontalRule}></View>
        <Text style={styles.bold}>Emergency Contact 3</Text>
        <Text style={styles.bold}>Name</Text>
        <TextInput
          style={styles.input}
          placeholder={"Emergency Contact Name"}
          value={this.state.emergencyName3}
          onChangeText={(value) => this.updateStateValue(value,
"emergencyName3")}
        />
        <Text style={styles.bold}>Emergency Contact Phone
Number</Text>
        <TextInput
          style={styles.input}
```

```
          placeholder={"Emergency Contact Number"}
          value={this.state.emergencyNumber3}
          onChangeText={(value) => this.updateStateValue(value,
"emergencyNumber3")}
        />
        <Text style={styles.bold}>Emergency Contact Relation</Text>
        <TextInput
          style={styles.input}
          placeholder={"Emergency Contact Relation"}
          value={this.state.emergencyRelation3}
          onChangeText={(value) => this.updateStateValue(value,
"emergencyRelation3")}
        />
        <View style={styles.horizontalRule}></View>
        <Text style={styles.bold}>Doctor's Name</Text>
        <TextInput
          style={styles.input}
          placeholder={"Doctor's Name"}
          value={this.state.doctorName}
          onChangeText={(value) => this.updateStateValue(value,
"doctorName")}
        />
        <Text style={styles.bold}>Doctor's Address</Text>
        <TextInput
          style={styles.input}
          placeholder={"Doctor's Address"}
          value={this.state.doctorAddress}
          onChangeText={(value) => this.updateStateValue(value,
"doctorAddress")}
        />
        <Text style={styles.bold}>Doctor's Phone Number</Text>
        <TextInput
          style={styles.input}
          placeholder={"Doctor's Number"}
          value={this.state.doctorNumber}
          onChangeText={(value) => this.updateStateValue(value,
"doctorNumber")}
        />
```

```
// Set the state variable value to the value supplied from the input
  updateStateValue = (value, prop) => {
    const state = this.state;
    state[prop] = value;
    this.setState(state);
  };
```

```
const documentUpdate =
app.firestore().collection("children").doc(this.state.key);
documentUpdate
  .set({
    child_name: this.state.name,
```

```
    child_DOB: this.state.dob,
    child_allergies: this.state.allergies,
    child_allergies_details: this.state.allergiesDetails,
    child_medical_conditions: this.state.medicalConditions,
    child_medical_conditions_details:
this.state.medicalConditionsDetails,
    child_is_active: this.state.isActive,
    child_emergency_contact_name_1: this.state.emergencyName1,
    child_emergency_contact_number_1: this.state.emergencyNumber1,
    child_emergency_contact_relation_1: this.state.emergencyRelation1,
    child_emergency_contact_name_2: this.state.emergencyName2,
    child_emergency_contact_number_2: this.state.emergencyNumber2,
    child_emergency_contact_relation_2: this.state.emergencyRelation2,
    child_emergency_contact_name_3: this.state.emergencyName3,
    child_emergency_contact_number_3: this.state.emergencyNumber3,
    child_emergency_contact_relation_3: this.state.emergencyRelation3,
    doctor_name: this.state.doctorName,
    doctor_address: this.state.doctorAddress,
    doctor_number: this.state.doctorNumber,
    child_home_address: this.state.childAddress
  })
```

```
<Button
    mode="contained"
    uppercase={false}
    color="#0B8FDC"
    onPress={() => this.editChild()}>
    <Text style={styles.buttonTextMenu}>Update</Text>
  </Button>
```

Solution

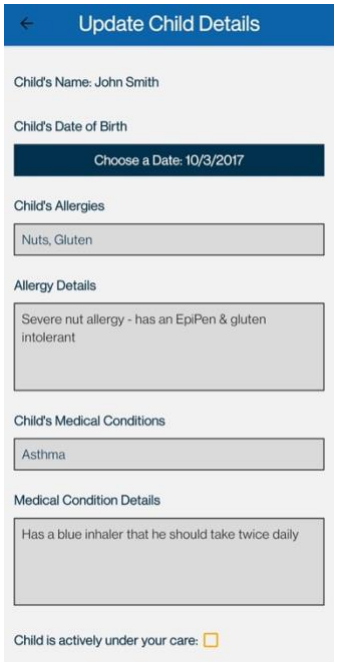


Figure 6.50: Firestore database displays the change in 'child_is_active' status

Figure 6.49: Active status of the child is updated to be false

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
34	Previously entered information can be viewed	Pass	
35	The childcare provider can update information of children in their care	Pass	
36	Changes are updated on Firestore database	Pass	

Table 6.13: Unit Testing for Requirements 3.4, 3.7

6.1.14 Requirement 3.6: De-active children from the database

Initial Acceptance Criteria: "The childcare provider can remove children and their relevant information from the database."

Revised Acceptance Criteria: "The childcare provider can de-active children within the application. This will maintain their relevant information within the database, but data cannot be entered against this child's name until they are re-activated."

Development

UpdateChildDetails.js

```
Text style={styles.bold}>Child is actively under your care:</Text>
<CheckBox
  style={styles.checkBox}
  disabled={false}
  value={this.state.isActive}
  onValueChange={(value) => this.updateStateValue(value, "isActive")}
  tintColors={{ true: "#0B8FDC", false: "orange" }}
/>
```

Solution

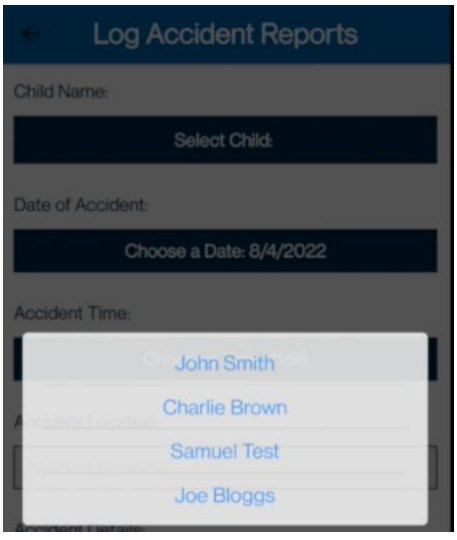


Figure 6.51: Modal selector populated with children that are actively in care

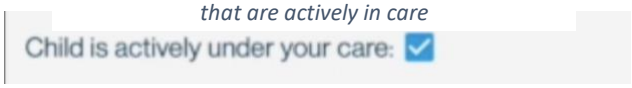


Figure 6.52: The child 'Charlie Brown' has active status set to true

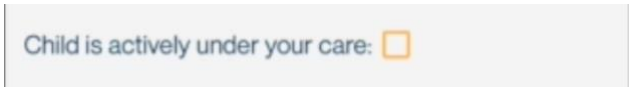


Figure 6.53: The child 'Charlie Brown' has active status changed to false

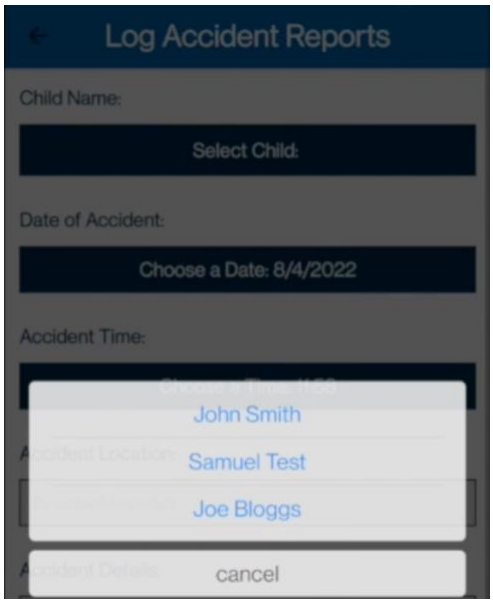


Figure 6.54: The child 'Charlie Brown' is no longer displayed in modal selector

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
37	Child can be marked as inactive	Pass	
38	Child will no longer appear in dropdown elements when logging information	Pass	

Table 6.14: Unit Testing for Requirements 3.6

6.1.15 Requirement 3.8: Call emergency contacts

Acceptance Criteria: "The childcare provider has quick access to make calls to emergency contacts that have been stored on the application."

Development

ViewEmergencyContacts.js

```
// Gather emergency contact data, store it to the contactDetails array
and set the state value
querySnapshot.forEach((result) => {
  const { child_name, child_emergency_contact_name_1,
child_emergency_contact_name_2, child_emergency_contact_name_3,
```

```
child_emergency_contact_number_1, child_emergency_contact_number_2,
child_emergency_contact_number_3, child_emergency_contact_relation_1,
child_emergency_contact_relation_2, child_emergency_contact_relation_3
} = result.data();
  contactDetails.push({
    key: result.id,
    child_name,
    child_emergency_contact_name_1,
    child_emergency_contact_name_2,
    child_emergency_contact_name_3,
    child_emergency_contact_number_1,
    child_emergency_contact_number_2,
    child_emergency_contact_number_3,
    child_emergency_contact_relation_1,
    child_emergency_contact_relation_2,
    child_emergency_contact_relation_3
  });
});
this.setState({
  contactDetails: contactDetails,
});
```

```
// Forward the user to their phonebook, auto-filling the phone with
the contact details that they clicked on
  forwardToPhonebook(source) {
    Linking.openURL(`tel:${source}`);
  }
```

```
{/* For each item in contactDetails */}
{this.state.contactDetails.map((contact, id, i) => {
  // Display the emergency contact info regarding the child name
selected the modal selector
  if (contact.child_name == this.state.activeChildName) {
    return (
      <View key={id}>
        <View style={styles.space}></View>
        <View style={styles.iconPadding}>
          <Text style={styles.navyTextWithLeftMargin}>Name:
            {contact.child_emergency_contact_name_1}
            ({contact.child_emergency_contact_relation_1})
          </Text>
          <MaterialIcons
            name={"phone"}
            size={30}
            color="#EE752E"
            style={styles.phoneIcon}
            onPress={() => {
this.forwardToPhonebook(contact.child_emergency_contact_number_1);
            }}
          />
        </View>
```

```
        <Text style={styles.boldBlueText}
          onPress={() => {
this.forwardToPhonebook(contact.child_emergency_contact_number_1);
          }}>Phone Number: {contact.child_emergency_contact_number_1}
        </Text>
        <View style={styles.horizontalRule}></View>
        <View style={styles.space}></View>
        <View style={{ flexDirection: 'row', justifyContent: 'space-
between' }}>
          <Text style={styles.navyTextWithLeftMargin}>
            Name: {contact.child_emergency_contact_name_2}
            ({contact.child_emergency_contact_relation_2})
          </Text>
          <MaterialIcons
            name={"phone"}
            size={30}
            color="#EE752E"
            style={styles.phoneIcon}
            onPress={() => {
this.forwardToPhonebook(contact.child_emergency_contact_number_2);
            }}
          />
        </View>
        <Text style={styles.boldBlueText}
          onPress={() => {
this.forwardToPhonebook(contact.child_emergency_contact_number_2);
          }}>Phone Number: {contact.child_emergency_contact_number_2}
        </Text>
        <View style={styles.horizontalRule}></View>
        <View style={styles.space}></View>
        <View style={styles.iconPadding}>
          <Text style={styles.navyTextWithLeftMargin}>
            Name: {contact.child_emergency_contact_name_3}
           ({contact.child_emergency_contact_relation_3})
         </Text>
          <MaterialIcons
            name={"phone"}
            size={30}
            color="#EE752E"
            style={styles.phoneIcon}
            onPress={() => {
    this.forwardToPhonebook(contact.child_emergency_contact_number_3);
            }}
          />
        </View>
        <Text style={styles.boldBlueText}
          onPress={() => {
    this.forwardToPhonebook(contact.child_emergency_contact_number_3);
          }}>Phone Number: {contact.child_emergency_contact_number_3}
        </Text>
      </View>
```

```
    );
  } else {
    null;
  }
})}
```

Solution

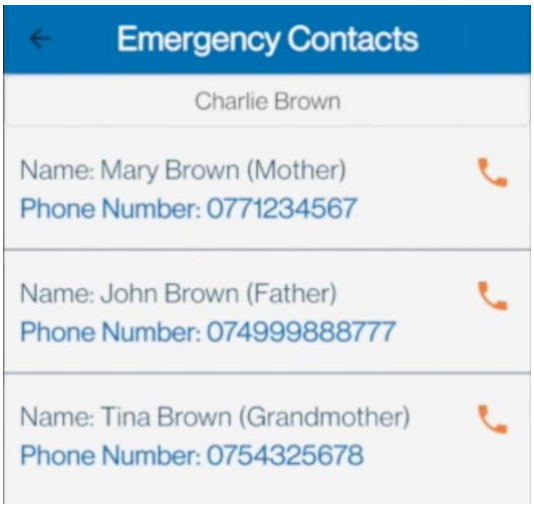


Figure 6.55: Emergency contacts are displayed for the child selected

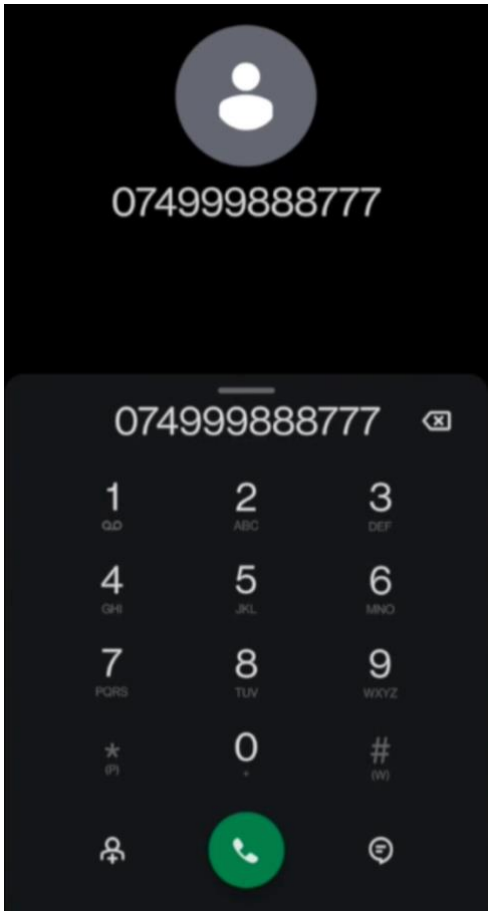


Figure 6.56: Pressing the phone number or phone icon will forward the user to their phone component and autofill the number

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
39	Emergency contact details appear	Pass	
40	When clicked upon, the shortcut to the device's phone functionality works as expected	Pass	

Table 6.15: Unit Testing for Requirements 3.8

6.1.16 Requirement 4.1: Scan food item and view nutritional details

Acceptance Criteria: "The childcare provider can scan the barcode of a product and understand the nutritional values of that product."

Development

AllergyDetection.js

```
const barcodeScan = ({ data }) => {
    // Set scanned variable as true
    setScanned(true);
    // Vibrate the device
    Vibration.vibrate();
    // Fetch product information from openfoodfacts api
  fetch(`https://world.openfoodfacts.org/api/v0/product/${data}.json`)
      .then((response) => response.json())
      .then((json) => {
        // If a product is identified, navigate to the ProductScreen
and include product info using item variable
        if (json.status_verbose === "product found") {
          navigation.navigate("ProductScreen", {
            item: json.product,
          });
        } else alert("Product not found");
      })
      // If an error occurs during this process, print an error
      .catch((error) => {
        console.error(error);
      });
  };
```

```
<BarCodeScanner
        // If scanned variable is false, run barcodeScan const
        onBarCodeScanned={scanned ? undefined : barcodeScan}
        style={StyleSheet.absoluteFillObject}
      />
      {/* While scanned variable is true, display this Button
component */}
```

```
      {scanned && (
        <Button
          mode="contained"
          uppercase={false}
          color="#0B8FDC"
          // When the Button is pressed, set scanned variable to false
which allows barcodeScan const to run
          onPress={() => setScanned(false)}>
          <Text style={styles.buttonTextMenu}>Scan Barcode</Text>
        </Button>
      )}
```

Nutrition.js

```
<Text style={styles.nutritionText}>Per 100g:</Text>
      <View style={styles.space}></View>
      {/* If calorie info is available, display it in Text component
*/}
      {nutriments["energy-kcal_100g"] != null && (
        <View style={styles.allNutrients}>
          <MaterialCommunityIcons
            name={"fire"}
            size={30}
            color="#02314D"
            style={styles.icons}
          />
          <View style={styles.inlineDisplay}>
            <Text style={styles.nutritionText}>Calories:</Text>
            <Text style={styles.nutritionDetail}>
              {nutriments["energy-kcal_100g"]}kcal
            </Text>
          </View>
        </View>
      )}
      <View style={styles.space}></View>
      {/* If fat info is available, display it in Text component */}
      {nutriments.fat_100g != null && (
        <View style={styles.allNutrients}>
          <MaterialIcons
            name={"grain"}
            size={30}
            color="#02314D"
            style={styles.icons}
          />
          <View style={styles.inlineDisplay}>
            <Text style={styles.nutritionText}>Fat:</Text>
            <Text
style={styles.nutritionDetail}>{nutriments.fat_100g}g</Text>
          </View>
        </View>
      )}
      <View style={styles.space}></View>
```

```
      {/* If saturated fat info is available, display it in Text
component */}
      {nutriments["saturated-fat_100g"] != null && (
        <View style={styles.allNutrients}>
          <SimpleLineIcons
            name={"drop"}
            size={30}
            color="#02314D"
            style={styles.icons}
          />
          <View style={styles.inlineDisplay}>
            <Text style={styles.nutritionText}>Saturated Fat:</Text>
            <Text style={styles.nutritionDetail}>
              {nutriments["saturated-fat_100g"]}g
            </Text>
          </View>
        </View>
      )}
      <View style={styles.space}></View>
      {/* If carbohydrate info is available, display it in Text
component */}
      {nutriments.carbohydrates_100g != null && (
        <View style={styles.allNutrients}>
          <MaterialCommunityIcons
            name={"corn"}
            size={30}
            color="#02314D"
            style={styles.icons}
          />
          <View style={styles.inlineDisplay}>
            <Text style={styles.nutritionText}>Carbohydrate:</Text>
            <Text style={styles.nutritionDetail}>
              {nutriments.carbohydrates_100g}g
            </Text>
          </View>
        </View>
      )}
      <View style={styles.space}></View>
      {/* If sugar info is available, display it in Text component */}
      {nutriments.sugars_100g != null && (
        <View style={styles.allNutrients}>
          <MaterialCommunityIcons
            name={"spoon-sugar"}
            size={30}
            color="#02314D"
            style={styles.icons}
          />
          <View style={styles.inlineDisplay}>
            <Text style={styles.nutritionText}>Sugar:</Text>
            <Text style={styles.nutritionDetail}>
              {nutriments.sugars_100g}g
```

```
            </Text>
          </View>
        </View>
      )}
      <View style={styles.space}></View>
      {/* If fibre info is available, display it in Text component */}
      {nutriments.fiber_100g != null && (
        <View style={styles.allNutrients}>
          <MaterialCommunityIcons
            name={"corn"}
            size={30}
            color="#02314D"
            style={styles.icons}
          />
          <View style={styles.inlineDisplay}>
            <Text style={styles.nutritionText}>Fibre:</Text>
            <Text
style={styles.nutritionDetail}>{nutriments.fiber_100g}g</Text>
          </View>
        </View>
      )}
      <View style={styles.space}></View>
      {/* If protein info is available, display it in Text component
*/}
      {nutriments.proteins_100g != null && (
        <View style={styles.allNutrients}>
          <MaterialCommunityIcons
            name={"cow"}
            size={30}
            color="#02314D"
            style={styles.icons}
          />
          <View style={styles.inlineDisplay}>
            <Text style={styles.nutritionText}>Protein:</Text>
            <Text style={styles.nutritionDetail}>
              {nutriments.proteins_100g}g
            </Text>
          </View>
        </View>
      )}
      <View style={styles.space}></View>
      {/* If salt info is available, display it in Text component */}
      {nutriments.salt_100g != null && (
        <View style={styles.allNutrients}>
          <MaterialCommunityIcons
            name={"cube-outline"}
            size={30}
            color="#02314D"
            style={styles.icons}
          />
          <View style={styles.inlineDisplay}>
```

```
            <Text style={styles.nutritionText}>Salt:</Text>
            <Text
style={styles.nutritionDetail}>{nutriments.salt_100g}g</Text>
          </View>
        </View>
      )}
```

Solution

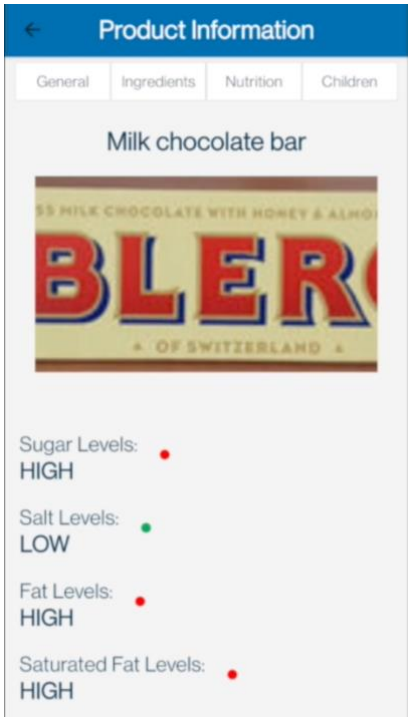


Figure 6.57: Nutritional overview is displayed

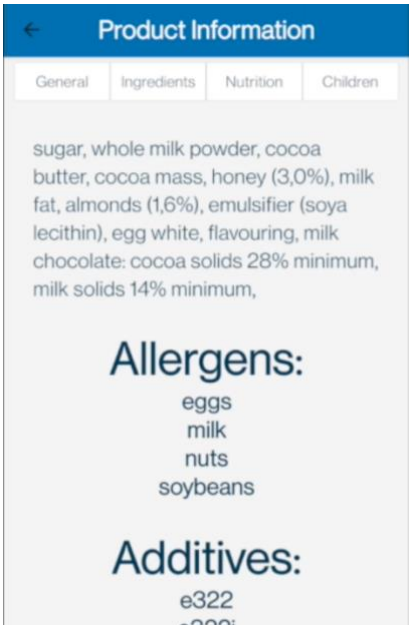


Figure 6.58: Product ingredients list is displayed

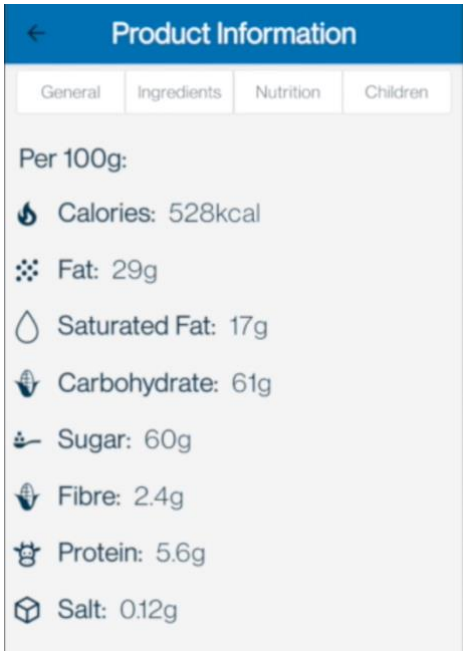


Figure 6.59: Nutritional breakdown is displayed

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
41	Nutritional information returned for a food product	Pass	

Table 6.16: Unit Testing for Requirements 4.1

6.1.17 Requirement 4.2: View the children that are allergic to food

Acceptance Criteria: "The childcare provider can scan the barcode of a product and understand the allergens within the product and which children this product is unsuitable for."

Development

Ingredients.js

```
<Text style={styles.ingredientsList}>
{/* If ingredients info is available, display it in Text component */}
  {ingredients_text ? ingredients_text : "Information Unavailable"
</Text>
<Text style={styles.paddingCenter}>
  Allergens:
</Text>
{/* If allergen info is available, loop over the list of allergens and
display them in Text component */}
{allergens_tags && allergens_tags.length > 0 ? (
allergens_tags.map((allergen, id) => (
    <Text key={id} style={styles.productHeader}>
      {trimAllergyString(allergen)}
    </Text>
  ))
) : (
  <Text style={styles.title}>Information Unavailable</Text>
)}
<Text style={styles.paddingCenter}>
  Additives:
</Text>
{/* If additive info is available, loop over the list of allergens and
display them in Text component */}
{additives_tags && additives_tags.length > 0 ? (
additives_tags.map((additive, id) => (
    <Text key={id} style={styles.productHeader}>
      {trimAllergyString(additive)}
    </Text>
  ))
) : (
  <Text style={styles.title}>Information Unavailable</Text>
```

```
)}
```

ChildAllergies.js

```
<Text style={styles.paddingCenter}>
  Child Allergies
</Text>
<Text style={styles.productName}>
  This Product is unsuitable for:
</Text>
<View style={styles.space}></View>
{ingredients_text ? (
  // For each allergy within the scanned product
  allergens_tags.map((allergen) => (
      // For each child within childNameArr
      childNameArr.map(child => (
          // If the child's allergy is contained within the product,
add the child's name to childArr
child.allergies.toLowerCase().includes(trimAllergyString(allergen)) ?
(addToArr(child.name)) : null )
      )
  ))
) : (
  <Text style={styles.title}>Information Unavailable</Text>
)}
{removeDuplicates()}
{/* For each child's name in childArr, display it in a Text element
*/}
{childArr.map((child) => (
  <Text style={styles.productName}>
      {child}
  </Text>
))}
```

```
// Add name to childArr array
  function addToArr(name){
    childArr.push(name);
  }
```

```
// Remove duplicate names from the array
  function removeDuplicates(){
    childArr = [...new Set(childArr)];
  }
```

Solution

Figure 6.60: A list of children that are allergic to the scanned food item

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
42	Children that a food product is unsuitable for are listed	Pass	

Table 6.17: Unit Testing for Requirements 4.2

6.1.18 Requirement 8.1: Add daily risk assessment data to the database

Acceptance Criteria: "The childcare provider is provided with a form to guide them through the expected checks involved in a daily risk assessment. The details of the completed daily risk assessments will be stored in a database."

Development

AddDailyRiskAssessment.js

```
// Initialising connection to dailyRiskAssessment database table
  const fireDB = app.firestore().collection("dailyRiskAssessment");
```

```
// Add variable values to the database and navigate the user to
HealthSafetyChecks page
  async function addCheck() {
    await fireDB.add({
      daily_risk_assessment_date: changeDate,
      daily_risk_assessment_check_1: dailyRiskAssessmentCheck1,
      daily_risk_assessment_check_2: dailyRiskAssessmentCheck2,
      daily_risk_assessment_check_3: dailyRiskAssessmentCheck3,
      daily_risk_assessment_check_4: dailyRiskAssessmentCheck4,
      daily_risk_assessment_check_5: dailyRiskAssessmentCheck5,
      daily_risk_assessment_check_6: dailyRiskAssessmentCheck6,
      daily_risk_assessment_check_7: dailyRiskAssessmentCheck7,
      daily_risk_assessment_check_8: dailyRiskAssessmentCheck8,
      daily_risk_assessment_check_9: dailyRiskAssessmentCheck9,
      daily_risk_assessment_check_10: dailyRiskAssessmentCheck10,
      daily_risk_assessment_check_11: dailyRiskAssessmentCheck11,
      daily_risk_assessment_check_12: dailyRiskAssessmentCheck12,
      daily_risk_assessment_check_13: dailyRiskAssessmentCheck13,
      daily_risk_assessment_check_14: dailyRiskAssessmentCheck14,
      daily_risk_assessment_check_15: dailyRiskAssessmentCheck15,
```

```
      daily_risk_assessment_check_16: dailyRiskAssessmentCheck16,
      daily_risk_assessment_check_17: dailyRiskAssessmentCheck17,
      daily_risk_assessment_check_18: dailyRiskAssessmentCheck18,
      daily_risk_assessment_check_19: dailyRiskAssessmentCheck19,
      daily_risk_assessment_check_20: dailyRiskAssessmentCheck20,
      daily_risk_assessment_check_21: dailyRiskAssessmentCheck21,
      daily_risk_assessment_check_22: dailyRiskAssessmentCheck22,
      daily_risk_assessment_check_23: dailyRiskAssessmentCheck23,
      daily_risk_assessment_check_24: dailyRiskAssessmentCheck24,
      daily_risk_assessment_check_25: dailyRiskAssessmentCheck25,
      daily_risk_assessment_check_26: dailyRiskAssessmentCheck26,
      daily_risk_assessment_check_27: dailyRiskAssessmentCheck27,
      daily_risk_assessment_check_28: dailyRiskAssessmentCheck28,
      daily_risk_assessment_check_29: dailyRiskAssessmentCheck29,
      daily_risk_assessment_check_30: dailyRiskAssessmentCheck30,
      daily_risk_assessment_check_31: dailyRiskAssessmentCheck31,
      daily_risk_assessment_check_32: dailyRiskAssessmentCheck32,
      daily_risk_assessment_check_33: dailyRiskAssessmentCheck33,
      daily_risk_assessment_check_34: dailyRiskAssessmentCheck34,
    });
    navigation.navigate("HealthSafetyChecks");
  }
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addCheck()}>
  <Text style={styles.buttonTextMenu}>Submit Check</Text>
</Button>
```

Solution

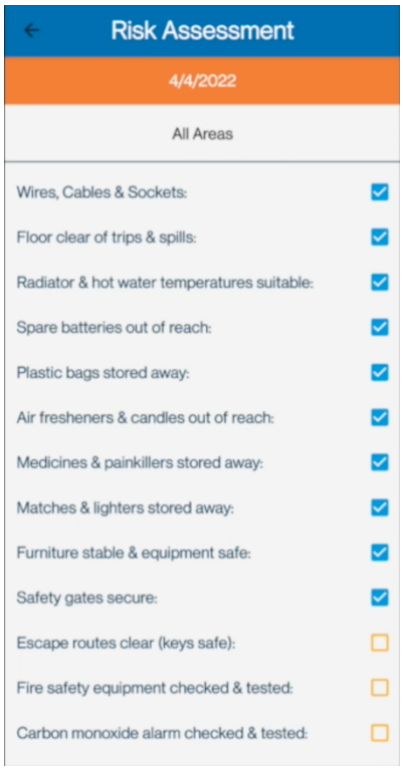


Figure 6.61: Daily risk assessment form

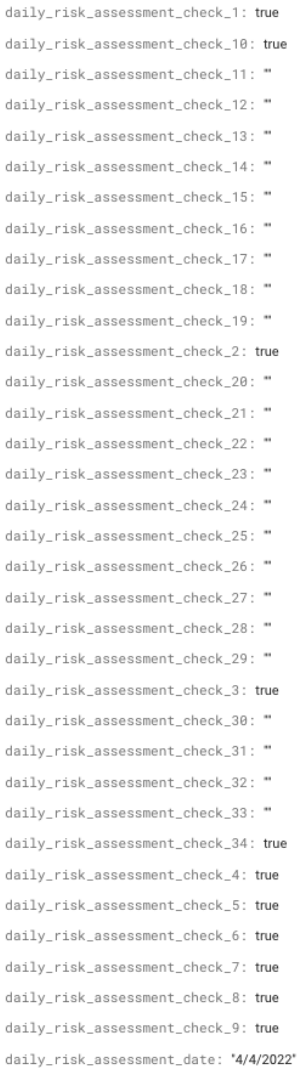


Figure 6.62: Daily risk assessment data submitted to Firestore database

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
43	Daily risk assessment guide can be completed on the application	Pass	
44	Daily risk assessment is stored in Firestore database	Pass	

Table 6.18: Unit Testing for Requirements 8.1

6.1.19 Requirement 8.2: Add daily COVID check data to the database

Acceptance Criteria: “The childcare provider is provided with a form to guide them through daily COVID checks. The details of the completed daily COVID checks will be stored in a database.”

Development

AddDailyCovidAssessment.js

```
  // Initialising connection to dailyCovidAssessment database table
  const fireDB = app.firestore().collection("dailyCovidAssessment");
```

```
// Add variable values to the database and navigate the user to
HealthSafetyChecks page
  async function addCheck() {
    await fireDB.add({
      daily_covid_assessment_date: changeDate,
      daily_covid_assessment_check_1: dailyCovidAssessmentCheck1,
      daily_covid_assessment_check_2: dailyCovidAssessmentCheck2,
      daily_covid_assessment_check_3: dailyCovidAssessmentCheck3,
      daily_covid_assessment_check_4: dailyCovidAssessmentCheck4,
      daily_covid_assessment_check_5: dailyCovidAssessmentCheck5,
      daily_covid_assessment_check_6: dailyCovidAssessmentCheck6,
      daily_covid_assessment_check_7: dailyCovidAssessmentCheck7,
      daily_covid_assessment_check_8: dailyCovidAssessmentCheck8,
      daily_covid_assessment_check_9: dailyCovidAssessmentCheck9,
      daily_covid_assessment_check_10: dailyCovidAssessmentCheck10,
      daily_covid_assessment_check_11: dailyCovidAssessmentCheck11,
      daily_covid_assessment_check_12: dailyCovidAssessmentCheck12,
      daily_covid_assessment_check_13: dailyCovidAssessmentCheck13,
      daily_covid_assessment_check_14: dailyCovidAssessmentCheck14,
      daily_covid_assessment_check_15: dailyCovidAssessmentCheck15,
      daily_covid_assessment_check_16: dailyCovidAssessmentCheck16,
      daily_covid_assessment_check_17: dailyCovidAssessmentCheck17,
      daily_covid_assessment_check_18: dailyCovidAssessmentCheck18,
      daily_covid_assessment_check_19: dailyCovidAssessmentCheck19,
      daily_covid_assessment_check_20: dailyCovidAssessmentCheck20,
      daily_covid_assessment_check_21: dailyCovidAssessmentCheck21,
      daily_covid_assessment_check_22: dailyCovidAssessmentCheck22,
      daily_covid_assessment_check_23: dailyCovidAssessmentCheck23,
      daily_covid_assessment_check_24: dailyCovidAssessmentCheck24,
      daily_covid_assessment_check_25: dailyCovidAssessmentCheck25,
      daily_covid_assessment_check_26: dailyCovidAssessmentCheck26,
      daily_covid_assessment_check_27: dailyCovidAssessmentCheck27,
      daily_covid_assessment_check_28: dailyCovidAssessmentCheck28,
      daily_covid_assessment_check_29: dailyCovidAssessmentCheck29,
      daily_covid_assessment_check_30: dailyCovidAssessmentCheck30,
    });
    navigation.navigate("HealthSafetyChecks");
  }
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addCheck()}>
  <Text style={styles.buttonTextMenu}>Submit Check</Text>
</Button>
```

Solution

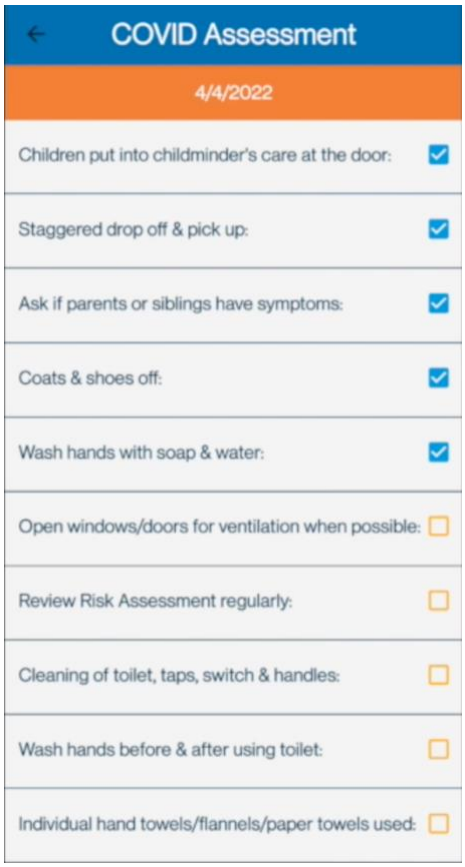


Figure 6.63: Daily COVID assessment form

Figure 6.64: Daily COVID assessment data submitted to Firestore database

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
45	Daily COVID assessment guide can be completed on the application	Pass	
46	Daily COVID assessment is stored in Firestore database	Pass	

Table 6.19: Unit Testing for Requirements 8.2

6.1.20 Requirement 8.3: Add monthly fire drill data to the database

Development

Acceptance Criteria: “The childcare provider is provided with a form to guide them through monthly fire drills. The details of the completed monthly fire drills will be stored in a database.”

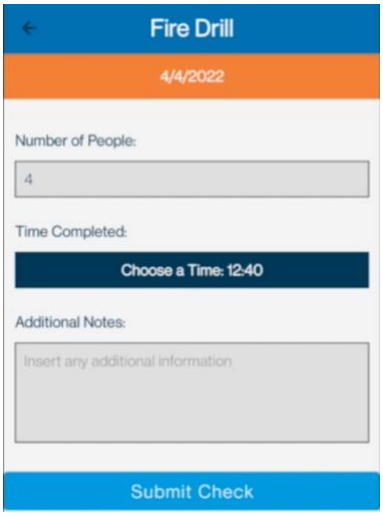
AddMonthlyDrillList.js

```
// Initialising connection to monthlyFireDrill database table
  const fireDB = app.firestore().collection("monthlyFireDrill");
```

```
async function addCheck() {
    // If inputs are valid, add variable values to the database
    if (isInputEmpty(monthlyFireDrillNumberOfPeople)) {
      missingDataAlert();
      return;
    } else if (!isNumeric(monthlyFireDrillNumberOfPeople)){
      numericDataAlert();
  // Add variable values to the database and navigate the user to
HealthSafetyChecks page
    } else {
      await fireDB.add({
        monthly_fire_drill_date: changeDate,
        monthly_fire_drill_num_of_people:
monthlyFireDrillNumberOfPeople,
        monthly_fire_drill_time_completed:
monthlyFireDrillTimeCompleted.date,
        monthly_fire_drill_note: monthlyFireDrillNote
      });
      navigation.navigate("HealthSafetyChecks");
    }
  }
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addCheck()}>
  <Text style={styles.buttonTextMenu}>Submit Check</Text>
</Button>
```

Solution



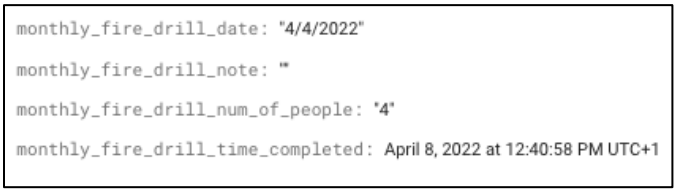


Figure 6.66: Monthly fire drill assessment data submitted to Firestore database

Figure 6.65: Monthly fire drill assessment form

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
47	Monthly fire drill guide can be completed on the application	Pass	
48	Monthly fire drill is stored in Firestore database	Pass	

Table 6.20: Unit Testing for Requirements 8.3

6.1.21 Requirement 8.4: Add fire safety equipment data to the database

Acceptance Criteria: "The childcare provider is provided with a form to guide them through monthly fire safety equipment checks. The details of the completed monthly fire safety equipment checks will be stored in a database."

Development

AddMonthlyFireSafetyEquipmentList.js

```
// Initialising connection to monthlyFireSafetyEquipmentCheck database
table
const fireDB =
app.firestore().collection("monthlyFireSafetyEquipmentCheck");
```

```
// Add variable values to the database and navigate the user to
HealthSafetyChecks page
  async function addCheck() {
    await fireDB.add({
      monthly_fire_safety_date: changeDate,
      monthly_fire_safety_note: monthlyFireSafetyNote
    });
    navigation.navigate("HealthSafetyChecks");
  }
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addCheck()}>
  <Text style={styles.buttonTextMenu}>Submit Check</Text>
</Button>
```

Solution

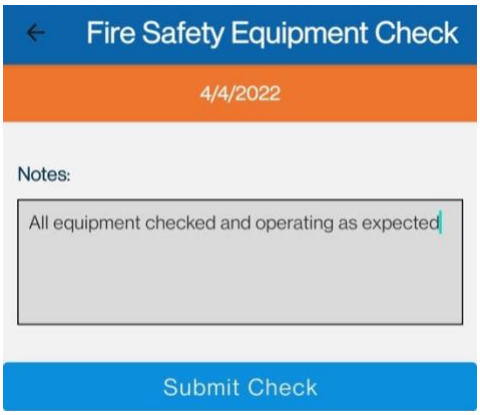


Figure 6.67: Monthly fire drill assessment form

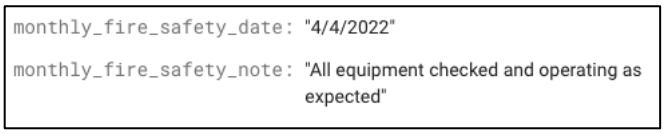


Figure 6.68: Monthly fire drill assessment data submitted to Firestore database

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
49	Monthly fire safety equipment guide can be completed on the application	Pass	
50	Monthly fire safety equipment is stored in Firestore database	Pass	

Table 6.21: Unit Testing for Requirements 8.4

6.1.22 Requirement 8.5: Update health and safety check data

Acceptance Criteria: "The childcare provider has access to previously submitted health and safety checks and has permission to update the data stored within the database."

Development

MonthlyFireDrill.js

```
// Query the database to gather monthly fire drill data, using userkey
as an identifier
const documentReference = app
      .firestore()
      .collection("monthlyFireDrill")
      .doc(this.props.route.params.userkey);
    // Once the database query has retrieved results, assign them to
state variable values
    documentReference.get().then((result) => {
      if (result.exists) {
        const data = result.data();
        this.setState({
          key: result.id,
          monthlyFireDrillDate: data.monthly_fire_drill_date,
          monthlyFireDrillNumberOfPeople:
data.monthly_fire_drill_num_of_people,
```

```
          monthlyFireDrillTimeCompleted:
this.trimTimestamp(data.monthly_fire_drill_time_completed),
          monthlyFireDrillNote: data.monthly_fire_drill_note,
        });
```

```
const documentUpdate =
app.firestore().collection("monthlyFireDrill").doc(this.state.key);
    documentUpdate
      .set({
        monthly_fire_drill_date: this.state.monthlyFireDrillDate,
        monthly_fire_drill_num_of_people:
this.state.monthlyFireDrillNumberOfPeople,
        monthly_fire_drill_time_completed:
this.state.monthlyFireDrillTimeCompleted,
        monthly_fire_drill_note: this.state.monthlyFireDrillNote
      })
```

```
// Set the state variable value to the value supplied from the input
  updateStateValue = (value, prop) => {
    const state = this.state;
    state[prop] = value;
    this.setState(state);
  };
```

```
// Converts a timestamp into time format
  trimTimestamp(dateInput) {
    var timeConvert = dateInput.toDate().toLocaleTimeString();
    timeConvert = timeConvert.split(":");
    return timeConvert[0] + ":" + timeConvert[1];
  }
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => this.editCheck()}>
  <Text style={styles.buttonTextMenu}>Update</Text>
</Button>
```

Solution

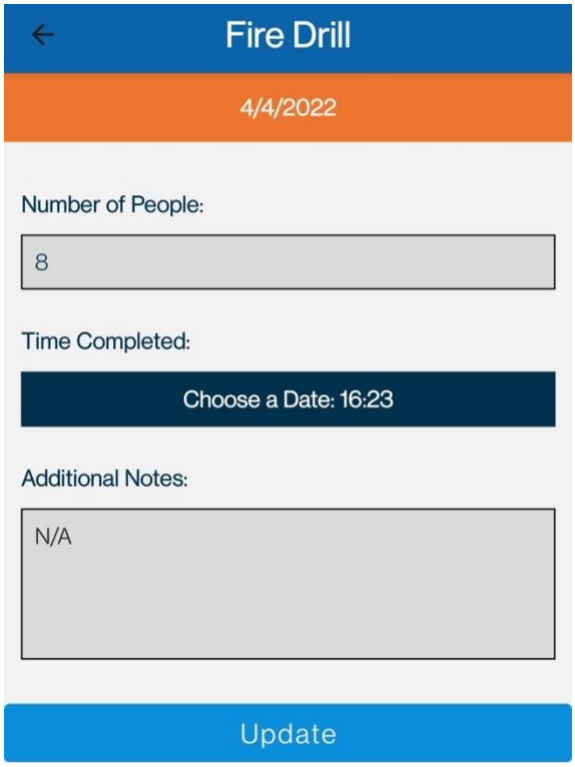


Figure 6.69: Monthly fire drill with initial values

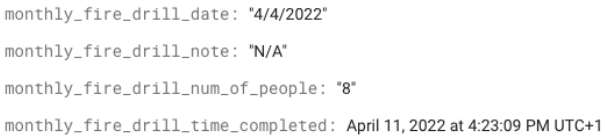


Figure 6.69: Monthly fire drill database log with initial values

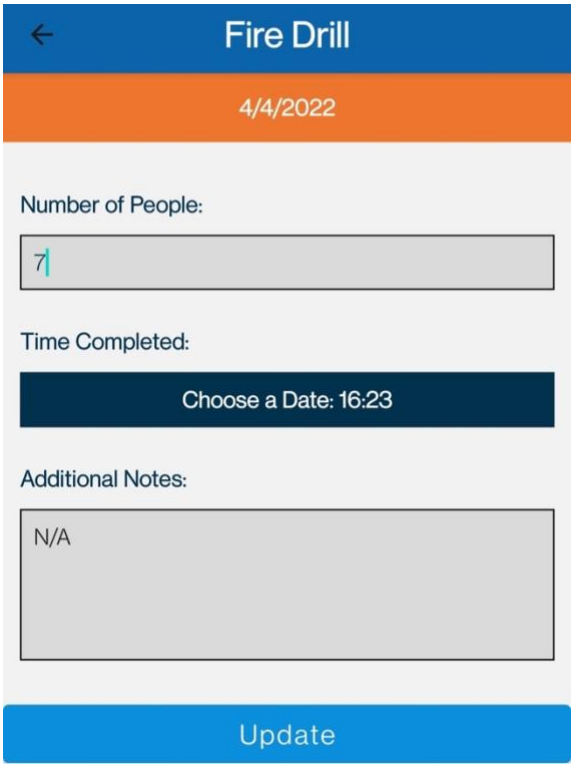


Figure 6.70: Monthly fire drill with updated input

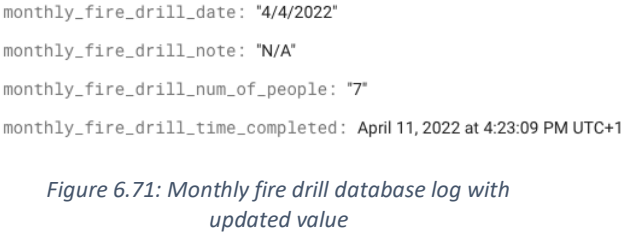


Figure 6.71: Monthly fire drill database log with updated value

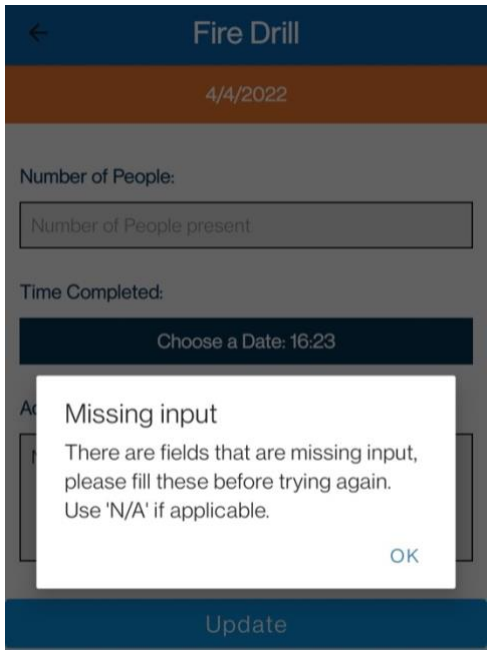


Figure 6.72: Validation used within health and safety check pages

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)

51	Health and safety information that was previously logged can be updated in the application	Pass	
52	Changes are updated on Firestore database	Pass	
53	Data validity is checked when submitted		

Table 6.22: Unit Testing for Requirements 8.5

6.1.23 Requirement 8.6: View statuses of health and safety checks

Acceptance Criteria: "The childcare provider can clearly view the status of each health and safety check per day, this status should display if the check has or hasn't been completed."

Development

DailyCovidList.js

```
// If no dailyCovidAssessment exists, return a blank view
    if (
      this.state.dailyCovidAssessment === undefined ||
      isInputEmpty(this.state.dailyCovidAssessment) ||
      isWeekday(this.props.changeDate)
    ) {
      return <View></View>;
    // Otherwise create a new array and filter dailyCovidAssessment
array to only include records which match the current date
    } else {
      const newArray = this.state.dailyCovidAssessment.filter(
        (item) => item.daily_covid_assessment_date ==
this.props.changeDate
      );
      // If there is a daily covid assessment log present for the
selected date
      if (newArray.length == 1) {
        // Display its info as a ListItem and set its status as
completed
        return (
          <ScrollView>
            {
              // for each daily covid check
              newArray.map((result, id) => {
                // if daily covid check has the selected date, display
in row
                return (
                  <ListItem
                    key={id}
                    onPress={() => {
                      // Navigate to the DailyCovidAssessment page and
populate fields data using the userkey variable as an identifier

this.props.navigation.navigate("DailyCovidAssessment", {
```

```
                        userkey: result.key,
                      });
                    }}
                    bottomDivider
                  >
                    <ListItem.Content>
                      <ListItem.Title
style={styles.navyBoldText}>Daily COVID Assessment</ListItem.Title>
                      <ListItem.Subtitle
style={styles.navyStandardText}>Is Completed: Yes</ListItem.Subtitle>
                    </ListItem.Content>
                    <ListItem.Chevron color="#02314D" />
                  </ListItem>
                );
              })
            }
          </ScrollView>
        );
      // Otherwise, if there is a daily covid assessment log present
for the selected date
      } else {
        // Display its info as a ListItem and set its status as
incompleted
        return (
          <ListItem
            onPress={() => {
              // Navigate to the AddDailyCovidAssessment page and pass
the changeDate parameter with it

this.props.navigation.navigate("AddDailyCovidAssessment", {
                changeDate: this.props.changeDate,
              });
            }}
            bottomDivider
          >
            <ListItem.Content>
              <ListItem.Title style={styles.navyBoldText}>Daily COVID
Assessment</ListItem.Title>
              <ListItem.Subtitle style={styles.navyStandardText}>Is
Completed: No</ListItem.Subtitle>
            </ListItem.Content>
            <ListItem.Chevron color="#02314D" />
          </ListItem>
        );
      }
    }
```

Solution

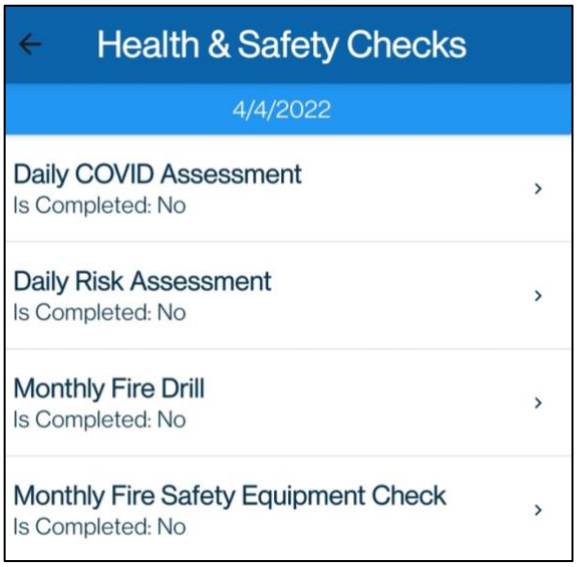


Figure 6.73: Statuses before checks are completed

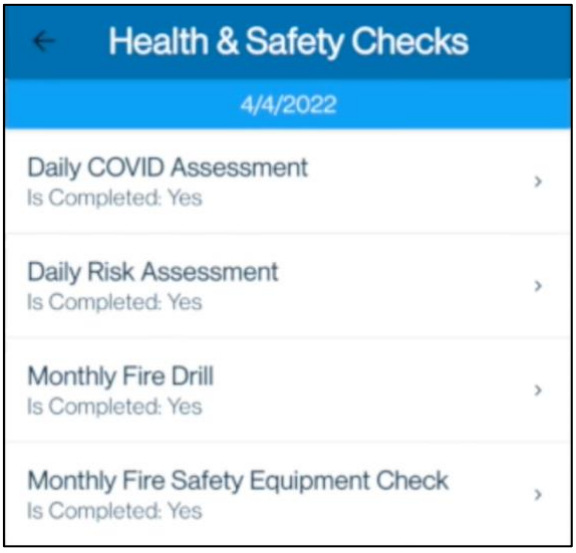


Figure 6.74: Statuses after checks are completed

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
54	Status is incomplete before check is completed	Pass	
55	Status is complete after check is completed	Pass	

Table 6.23: Unit Testing for Requirements 8.6

6.1.24 Requirement 11.1: Add mileage expense data to the database

Acceptance Criteria: “The childcare provider can add mileage expenses within the application, this information is stored within a database.”

Development

LogMiles.js

```
  // Initialising connection to mileageLogs database table
  const fireDB = app.firestore().collection("mileageLogs");
```

```
await fireDB.add({
  mileage_amount: mileageAmount,
  mileage_rate: mileageRate,
  miles_travelled: milesTravelled,
  date_of_mileage: dateOfMileage.date,
});
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => calculateAndAlert()}>
  <Text style={styles.buttonTextMenu}>Log Mileage</Text>
</Button>
```

Solution

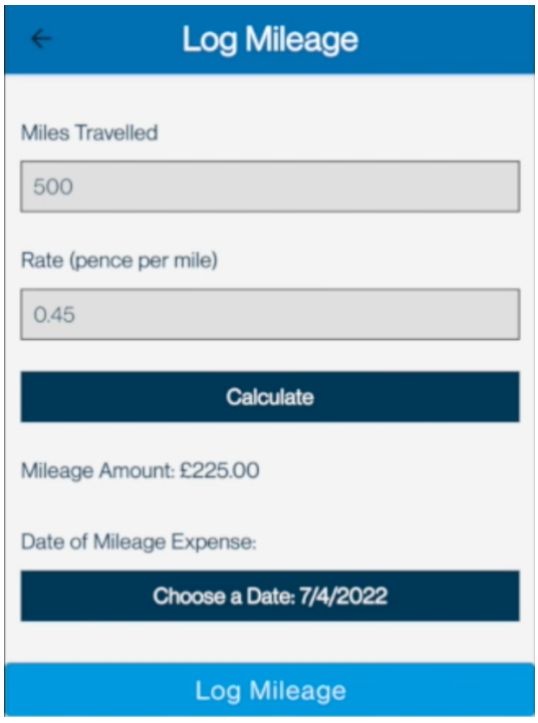


Figure 6.75: Log mileage page

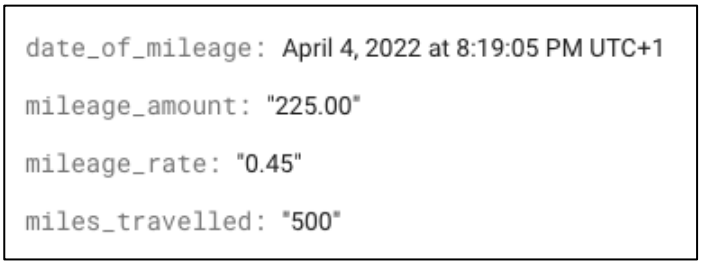


Figure 6.76: Inputs are stored in a mileage Firestore database

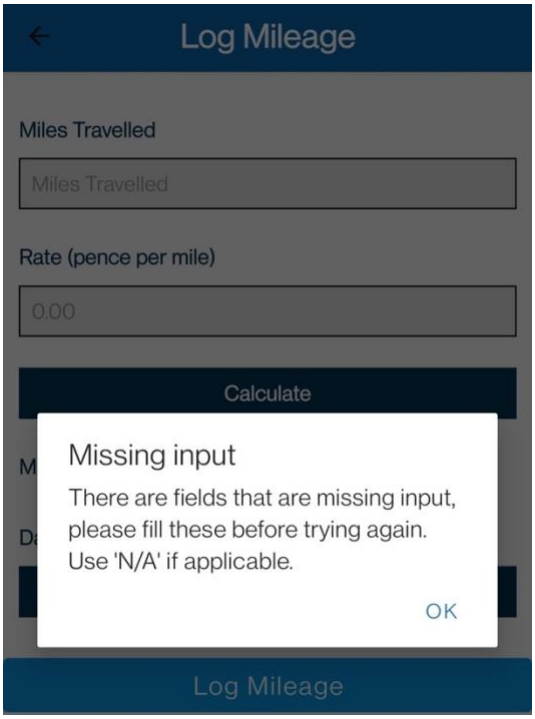


Figure 6.77: Input validation for missing inputs

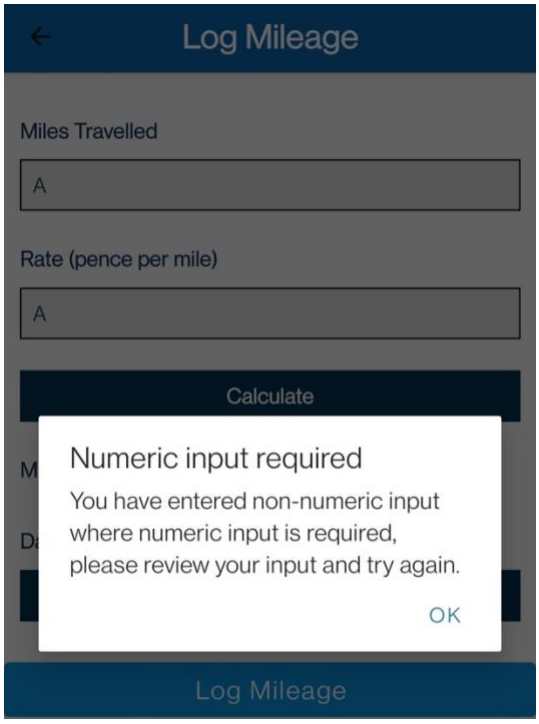


Figure 6.78: Input validation for non-numerical inputs

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
56	Mileage is logged successfully	Pass	
57	Details are stored in Firestore database	Pass	
58	Data validity is checked when submitted	Pass	

Table 6.24: Unit Testing for Requirements 11.1

6.1.25 Requirement 11.2: Add invoice expense data to the database

Acceptance Criteria: "The childcare provider can add invoices within the application, this information is stored within a database."

Development

LogInvoice.js

```
// Initialising connection to invoiceLogs database table
  const fireDB = app.firestore().collection("invoiceLogs");
```

```
await fireDB.add({
  child_name: childName,
  invoice_amount: invoiceAmount,
  date_of_invoice: dateOfInvoice.date,
});
```

```
<Button
  mode="contained"
  uppercase={false}
```

```
  color="#0B8FDC"
  onPress={() => addInvoiceLog()}>
  <Text style={styles.buttonTextMenu}>Log Invoice</Text>
</Button>
```

Solution

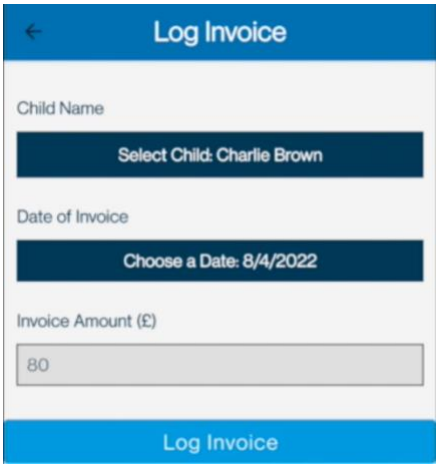


Figure 6.79: Log invoice page

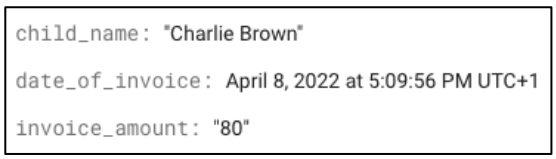


Figure 6.80: Inputs are stored in an invoice Firestore database

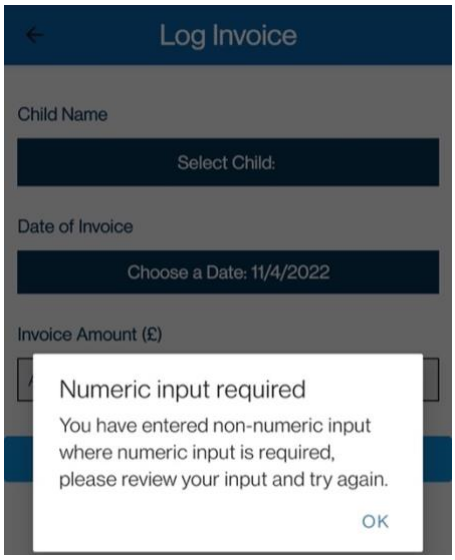


Figure 6.81: Input validation for non-numerical inputs

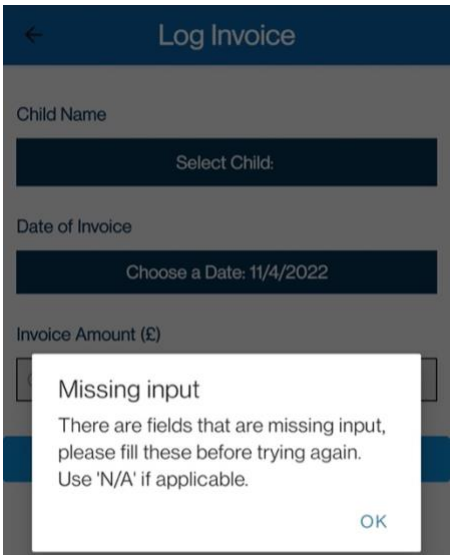


Figure 6.82: Input validation for missing inputs

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
59	Invoice is logged successfully	Pass	
60	Details are stored in Firestore database	Pass	
61	Data validity is checked when submitted	Pass	

Table 6.25: Unit Testing for Requirements 11.2

6.1.26 Requirement 11.3: Add other expense data to the database

Acceptance Criteria: “The childcare provider can add all other relevant expenses within the application, this information is stored within a database.”

Development

LogExpense.js

```
  // Initialising connection to expenseLogs database table
  const fireDB = app.firestore().collection("expenseLogs");
```

```
await fireDB.add({
  expense_title: expenseTitle,
  expense_note: expenseNote,
  expense_amount: expenseAmount,
  date_of_expense: dateOfExpense.date,
  receipt_url: receiptURL,
  expense_category: category,
});
```

```
<Button
  mode="contained"
  uppercase={false}
  color="#0B8FDC"
  onPress={() => addExpenseLog()}>
  <Text style={styles.buttonTextMenu}>Log Expense</Text>
</Button>
```

Solution

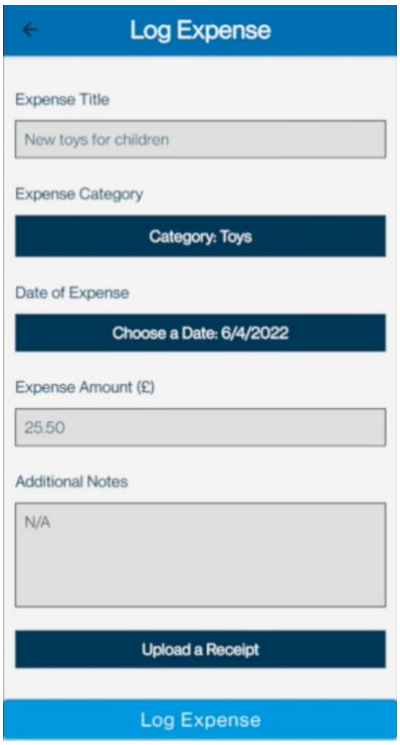


Figure 6.83: Log expense page

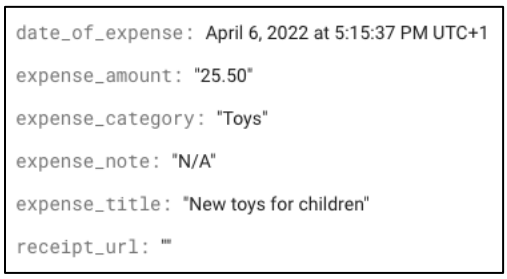


Figure 6.84: Inputs are stored in an expense Firestore database

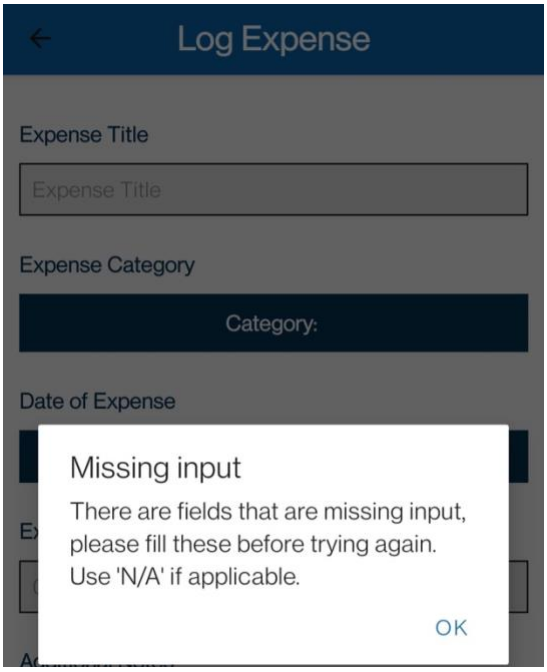


Figure 6.85: Input validation for missing inputs

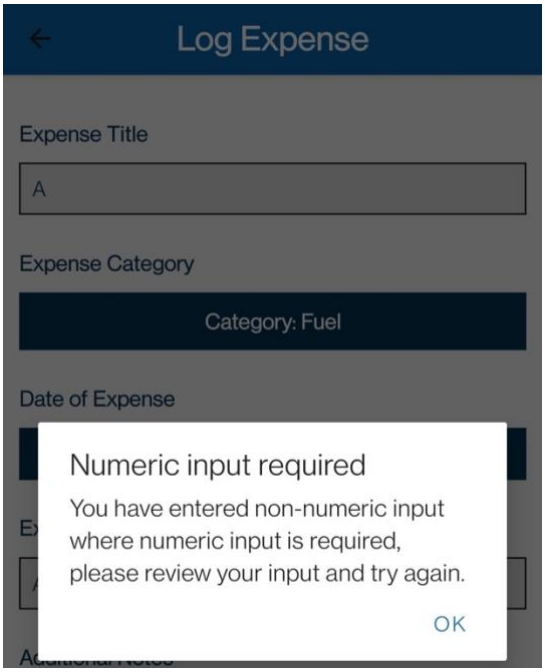


Figure 6.86: Input validation for non-numerical inputs

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
62	Expense is logged successfully	Pass	
63	Details are stored in Firestore database	Pass	
64	Data validity is checked when submitted	Fail	Validation has been used for most areas. However, the category has not been treated correctly for validation as the form accepts this when it is empty. This will be addressed in future developments.

Table 6.26: Unit Testing for Requirements 11.3

6.1.27 Requirement 11.4: View overview of expenses

Acceptance Criteria: "The childcare provider should be presented with an overview of their total expenses across a range of time."

Development

ViewExpenses.js

```
// Increment the expenseTotal by accumulatively adding the
expense_amount of each expense
this.state.expenseTotal += parseFloat(result.expense_amount);
```

```
<Text style={styles.boldLargeText}>
  {/* Display the expenseTotal in a Text component */}
  Monthly Total: £{parseFloat(this.state.expenseTotal).toFixed(2)}
</Text>
```

ViewInvoice.js

```
// Increment the invoiceAmount by accumulatively adding the
invoice_amount of each expense
this.state.invoiceAmount += parseFloat(result.invoice_amount);
```

```
<Text style={styles.boldLargeText}>
  {/* Display the invoiceAmount in a Text component */}
  Monthly Total: £{parseFloat(this.state.invoiceAmount).toFixed(2)}
</Text>
```

ViewMiles.js

```
// Increment the mileageAmount by accumulatively adding the
mileage_amount of each expense
this.state.mileageAmount += parseFloat(result.mileage_amount);
```

```
<Text style={styles.boldLargeText}>
  {/* Display the invoiceAmount in a Text component */}
  Monthly Total: £{parseFloat(this.state.mileageAmount).toFixed(2)}
</Text>
```

Solution

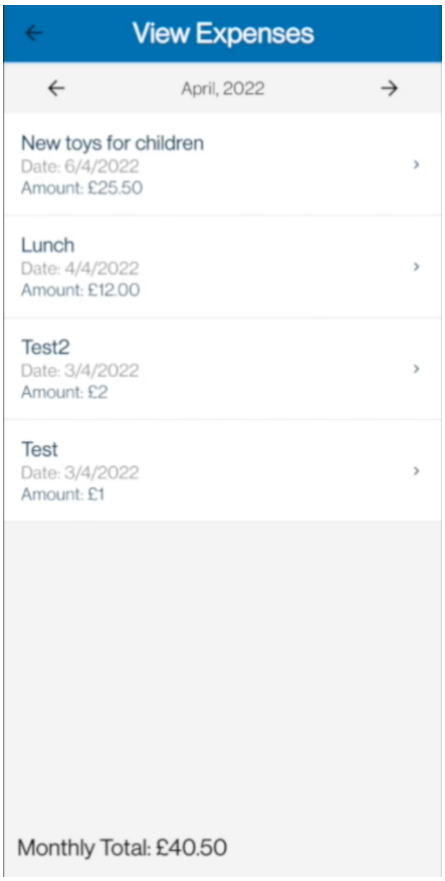


Figure 6.87: View expense page includes a monthly total overview

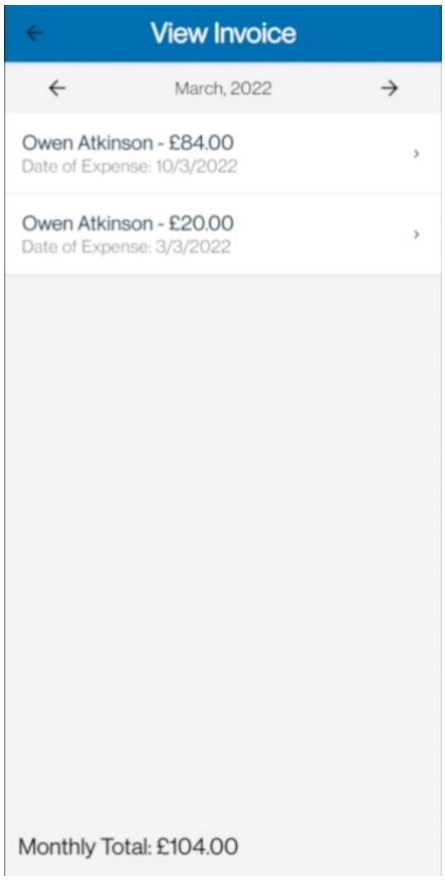


Figure 6.88: View invoice page includes a monthly total overview

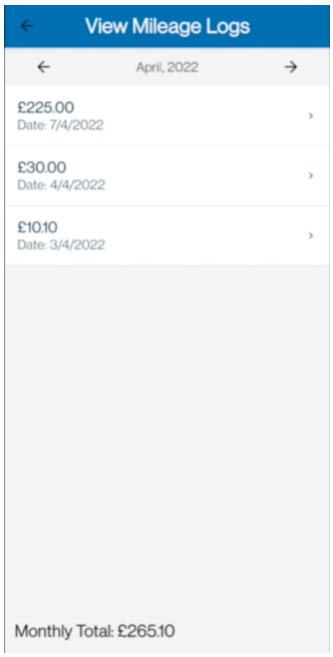


Figure 6.89: View invoice page includes a monthly total overview

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
65	Total amount can be viewed for expenses logs	Pass	
66	Total amount can be viewed for invoice logs	Pass	
67	Total amount can be viewed for mileage logs	Pass	

Table 6.27: Unit Testing for Requirements 11.4

6.1.28 Requirement 11.5: View overview of expenses

Acceptance Criteria: "The childcare provider has the option to upload an image of a receipt when submitting an expense."

Development

LogExpense.js

```
const pickImage = async () => {
    // Launch image picker
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.All,
      allowsEditing: true,
      quality: 1,
    });

    // If the image picker isn't cancelled, set the image variable to
equal the uri of the selected image
    if (!result.cancelled) {
      setImage(result.uri);
```

```
    }

    uploadImage(result.uri);
  };
```

```
async function uploadImage(file) {
    // Use a blob to store XML information regarding image being
uploaded
    const blob = await new Promise((resolve, reject) => {
      const xml = new XMLHttpRequest();
      xml.onload = function () {
        resolve(xml.response);
      };
      xml.onerror = function () {
        reject(new TypeError("Network request failed"));
      };
      xml.responseType = "blob";
      xml.open("GET", file, true);
      xml.send(null);
    });

    // Trim the filename into an appropriate format to be stored on
firebase storage
    let trimFileName = /[^/]*$/.exec(file)[0];
    // Provide firebase storage location to store the receipt
    const ref = app.storage().ref(`/receipts/${trimFileName}`);
    const snapshot = ref.put(blob);

    snapshot.on(
      "state_changed",
      // Next callback can be blank here
      function () {
      },
      // Error callback used to capture error
      function (error) {
        console.log(error);
        blob.close();
        return;
      },
      // Complete callback used set the ReceiptURL variable value
      function () {
        snapshot.snapshot.ref.getDownloadURL().then(function
(downloadURL) {
          setReceiptURL(downloadURL);
        });
        blob.close();
        return;
      }
    );
  }
```

```
<View>
  <TouchableOpacity style={styles.button} onPress={pickImage}>
    <Text style={styles.buttonText}>Upload a Receipt</Text>
  </TouchableOpacity>
</View>
{image && <Image source={{ uri: image }} style={styles.receiptPreview}
/>}
```

Solution

Figure 6.90: Upload receipt functionality

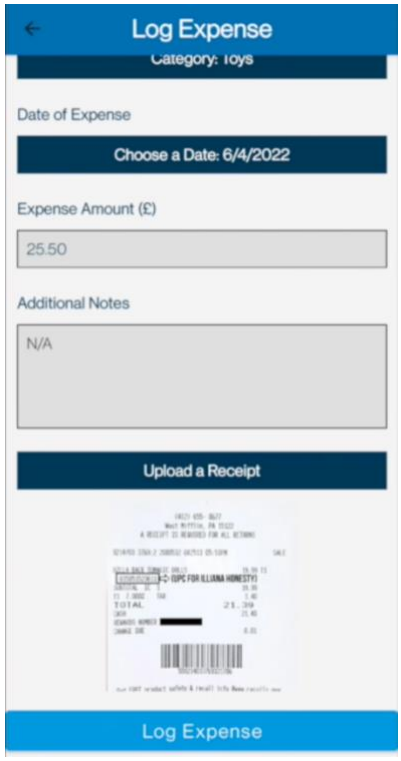


Figure 6.91: Receipt preview after uploading

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
68	Receipt is uploaded to Firebase storage container	Pass	

Table 6.28: Unit Testing for Requirements 11.5

6.1.29 Requirement 11.6: View receipts within expenses

Acceptance Criteria: “The childcare provider can view the receipts which they have previously uploaded within the application.”

Development

UpdateExpense.js

```
// If a receipt has been attached to the expense, display a 'View
Receipt Button'
```

```
if (this.state.receiptUrl !== "") {
  receiptButton = (
    <Button
      mode="contained"
      uppercase={false}
      color="#02314D"
      onPress={() =>
        // Navigate the user to the 'ReceiptPreview' page and pass the
        receiptUrl
        this.props.navigation.navigate("ReceiptPreview", {
          receiptImage: this.state.receiptUrl,
        })
      }>
      <Text style={styles.buttonTextMenu}>View Receipt</Text>
    </Button>
  );
}
```

ReceiptPreview.js

```
const ReceiptPreview = ({ route }) => {
  return (
    <View style={styles.imageView}>
      {/* Displays the receipt image in full screen, using the passed
      receiptUrl */}
      <Image
        source={{ uri: route.params.receiptImage }}
        style={styles.receiptImage}
      />
    </View>
  );
};
```

Solution

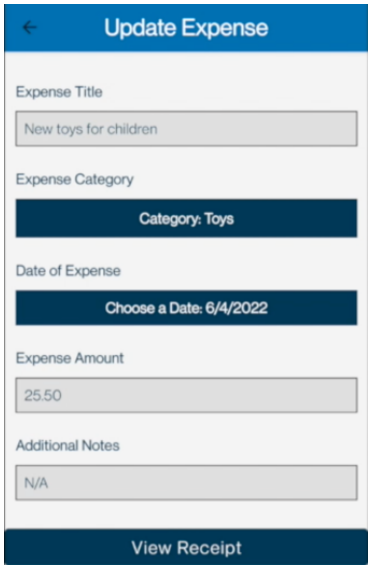


Figure 6.92: View receipt button is available when viewing an expense

Figure 6.93: View receipt functionality

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
69	Previously uploaded receipt can be viewed	Pass	

Table 6.29: Unit Testing for Requirements 11.6

6.1.30 Requirement 11.7: Calculate mileage amount

Acceptance Criteria: "The childcare provider has a feature within the mileage expense page to calculate the total expense for mileage based on miles driven and the variable rate."

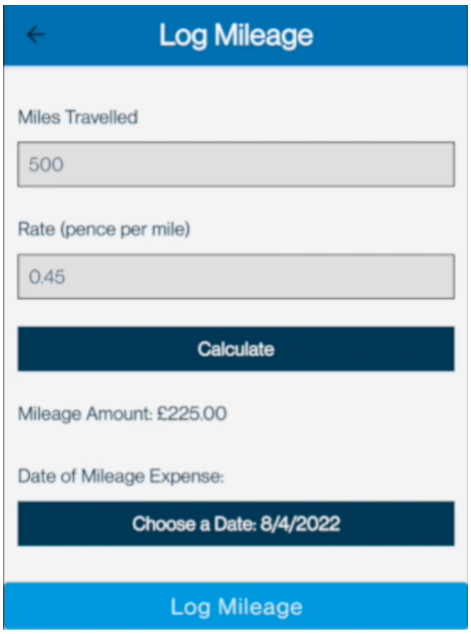
Development

LogMiles.js

```
// calculates the mileage amount based on the milesTravelled and
mileageRate variables
 function calculateMileageAmount(){
   setMileageAmount(
     parseFloat(milesTravelled * parseFloat(mileageRate)).toFixed(2)
   );
 }
```

```
<Text style={styles.bold}>Mileage Amount: £{mileageAmount}</Text>
```

Solution



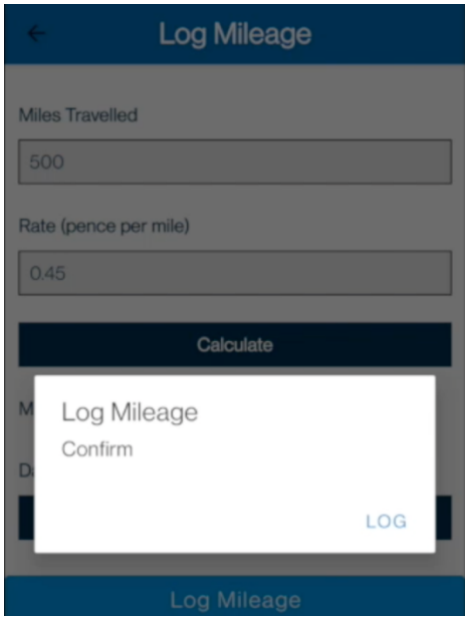


Figure 6.95: 'Log mileage' button can also be used to generate a value for mileage amount

Unit Testing

Test #	Testing Criteria	Pass/Fail	Notes (if failed)
70	Mileage is calculated when the 'calculate' button is pressed	Pass	
71	Mileage is calculated when the user submits the log information without pressing the 'calculate' button	Pass	

Table 6.30: Unit Testing for Requirements 11.7

6.2 Development Challenges

There were several challenges that Owen encountered during his development, these were largely unforeseen, but Owen managed to overcome them through perseverance.

Figure 6.94: 'Calculate' button can be used to generate a value for mileage amount

6.2.1 Database connection

Owen found it challenging to structure his database systems on Firebase and to establish an initial connection through the application. Through reading Firebase documentation and guidance, Owen got the database successfully connected with read and write access from the application.

6.2.2 Health & Safety checks

The health and safety checks required a lot of logic to be applied which Owen found complex at first. This included health and safety checks only being displayed on weekdays and monthly checks only being displayed on the first Monday of each month. Owen successfully implemented this component through breaking the work down into smaller tasks which were much more manageable to achieve.

6.2.3 Policy management

When developing the policy component of the application, Owen had to utilise a Firebase Storage container to upload PDF documents to and a Firebase Realtime Database to store the sharable URL connected to the PDF document. Owen was largely unfamiliar with both of these Firebase services but through utilising developer guides, he successfully achieved this functionality.

6.2.4 Receipt management

To achieve the ability of uploading receipts with expenses, Owen had to familiarise himself with how XML requests and promises operated. He did so

by reading online guides which explained how to use them and allow him to upload receipts to Firebase Storage.

6.2.5 Drop-down menus populated by a database query

Within many parts of the application, there are drop-down menus populated with names of children that the childcare provider has added into their system. Owen was unsure how this could be achieved and couldn't find any online guidance to assist this. Instead, he had to utilise his existing knowledge and experiment with his code. Owen successfully achieved this functionality through a Firestore query which was stored as an array and populated the drop-down component.

6.3 User Testing

Owen carried out a session of user testing where he provided several users with access to his application. After their session of testing, Owen provided them with a short questionnaire to understand their thoughts on multiple aspects of the application.

Childcare Management Application User Survey

1. Has your occupation ever been a childcare provider? Yes No

2. How would you describe your technical ability?

1	2	3	4	5
(Not technical)		(Average)		(Very technical)

3. What age are you?

4. What is your gender? Male Female

5. How easy was it to understand the features of the application and how they worked?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

6. How useful was the user guide at explaining the features of the application?

1	2	3	4	5
(Not useful)		(Average)		(Very useful)

7. Was there any key information missing from the user guide? If so, what was missing?

8. How easy was it to understand the navigation of the application?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

9. How would you rate the layout, design and overall appearance of the application?

1	2	3	4	5
(Very poor)		(Average)		(Very good)

10. What do you feel is the most valuable feature of the application?

11. Are there any improvements that you would make to the application? If so, what changes would you make?

12. Do you believe this application is useful for a childcare provider? Yes No

6.4 Survey Insights

After gaining the insights of several individuals through user testing, Owen analysed the results and concluded a list of statistics and conclusions which relate to his application.

- Has your occupation ever been a childcare provider?

Yes	No
50%	50%

Table 6.31: User testing occupation breakdown

- How would you describe your technical ability?

1	2	3	4	5
25%	0%	25%	25%	25%

Table 6.32: User testing technical ability breakdown

- What age are you?

16-25	26-35	36-45	46-55	56+
50%	0%	0%	25%	25%

Table 6.33: User testing age breakdown

- What is your gender?

Male	Female
50%	50%

Table 6.34: User testing gender breakdown

- How easy was it to understand the features of the application and how they worked?

1	2	3	4	5
0%	0%	0%	25%	75%

Table 6.35: User testing understanding features breakdown

- How useful was the user guide at explaining the features of the application?

1	2	3	4	5
0%	0%	0%	0%	100%

Table 6.36: User testing user guide usefulness breakdown

- How easy was it to understand the navigation of the application?

1	2	3	4	5
0%	0%	0%	0%	100%

Table 6.37: User testing understanding navigation breakdown

- How would you rate the layout, design and overall appearance of the application?

1	2	3	4	5
25%	0%	0%	25%	75%

Table 6.38: User testing appearance rating breakdown

- Do you believe this application is useful for a childcare provider?

Yes	No
100%	0%

Table 6.39: User testing application usefulness breakdown

6.4.1 Conclusions

Overall, Owen was pleased with the feedback that he received about his application. He felt that it was valuable to target a mixed demographic of those with and without experience as a childcare provider. He also felt that it was valuable to receive feedback from a range of age groups. Given additional time, Owen would have addressed the 26-45 age group to get a better understanding of their thoughts on the application. The user testing phase was also beneficial for Owen to gain additional feedback on his application and to note these points within his future developments.

These points for future developments included:

- Including a feature to allow for parents to sign documents via electronic signatures.
- Allowing the user to upload additional consent forms, contracts and parent confirmation of policies.
- Keyboard automatically changes to numeric input when numeric values will be entered.
- Increase the size of checkboxes within health and safety checks to make them easier to click on.
- Additional input options such as postcode look up for addresses and an autofill search component to assist the user when adding child medications.

6.5 User Guide

Requirement 25 states that a user guide will be included in the documentation that explains to new users how to use the systems core functionality. Below is the resulting guide:

What is this application?

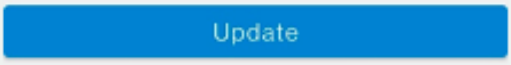
The goal of this application is to support the day-to-day tasks of childcare providers by offering a digital alternative. This aims to reduce the amount of time spent completing paperwork and administrative tasks, allowing the childcare provider to focus on providing a high level of care to children.

Logging Information

You can input information regarding health and safety checks, children in your care, attendance records, expenses, mileage logs, invoices, medicine administration and visitor records. This information is submitted within forms by clicking the 'Log' button, this will be stored within a secure database.

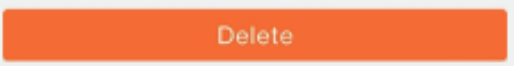
Updating Information

When viewing your information within the app, there is an option at the bottom of all logs that allows you to update the information stored within that specific log entry. By changing the inputs and clicking the 'Update' button, the information stored within that log entry will be updated.



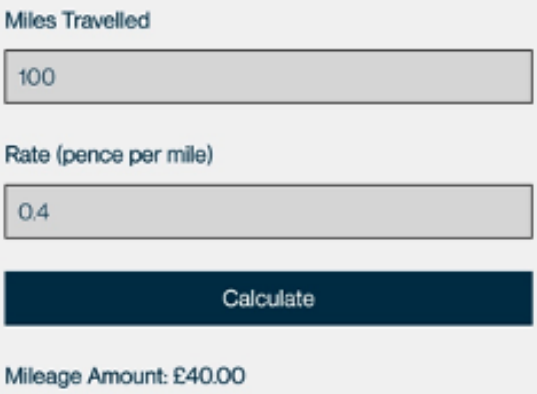
Deleting Information

When viewing your information within the app, there is an option at the bottom of most logs that allows you to delete the information stored within that specific log.



Calculating mileage

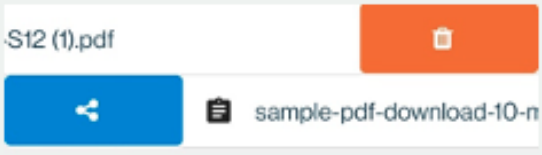
Mileage expenses are based on a calculation of miles travelled and the rate of pence per mile. Inserting values for each of these inputs and clicking the 'Calculate' button will automatically generate a mileage amount to be expensed.



Policies

Policies can be added by clicking the 'Upload Policy' button. This will open your device's document library. From here, you can select the policy that you want to upload. A progress bar will be displayed to show the upload progress of the file.

After it's uploaded, it will appear in the list of policies. Sliding each of these items on the list to the left or right will reveal additional functionality for either sharing or deleting.



Emergency Contacts

Emergency contact information for each child can be viewed as a summary within this page of the application. Clicking on either the phone number or phone icon will forward you to the phone keypad component of your phone and autofill the input.

Name: Rachel Bloggs (Mother)
Phone Number: 123456789

Deactivating Children

Children can be deactivated when they are no longer in your care. Deactivating a child will ensure that their name is no longer included within forms across the application. However, any historic data on this child will always remain. This ensures that information regarding any child that has ever been in your care will be preserved securely.

Child is actively under your care:

Activating children

Children can be activated to represent when they are actively in your care. Activating a child will ensure that their name can be included within forms across the application.

Child is actively under your care:

Adding Receipts

Clicking on the 'Upload a Receipt' button will open your device's photo library. From here, you can select the image that you wish to attach to the expense that you are logging. This receipt can be viewed in full screen later when viewing your logged expenses.

Upload a Receipt

Allergy Detection

Placing a barcode in front of your camera and pressing the 'Scan Barcode' button will capture the product information of a food item.

Scan Barcode

You will be presented with a general overview, ingredients list, nutrition details and a list of children in your care that the food item is unsuitable for due to allergies.

General | Ingredients | Nutrition | Children

Filters

When viewing information in the app, various types of filters can be used:

Date pickers which only display information connected to a date that you select.

30/3/2022

Dropdown selectors which only display information connected to specific children that you select.

Select Child

Monthly date navigators which only display information connected to that specific month and year that you navigate to.

February, 2022

March, 2022

7. Evaluation

7.1 Requirements Validation

Completed	Partially completed	Incomplete
70%	**4%**	**26%**
36/53	2/53	14/53

Table 7.1: Requirements completion rates

Must Have	Should Have	Could Have	Will Not Have
47%	**15%**	**11%**	**26%**
25/53	8/53	6/53	14/53

Table 7.2: Requirements completion rate per MoSCoW category

7.2 Requirements Completion Overview

ID	MoSCoW Category	Completion Status	Comments
1.1	Must Have	Completed	
1.2	Must Have	Completed	
1.3	Must Have	Completed	
1.4	Must Have	Completed	
1.5	Will Not Have	Incomplete	This was a low priority requirement
1.6	Must Have	Completed	
1.7	Should Have	Completed	
1.8	Should Have	Completed	
1.9	Must Have	Completed	
2.1	Should Have	Completed	
2.2	Should Have	Completed	
2.3	Should Have	Completed	
2.4	Should Have	Completed	
3.1	Must Have	Completed	
3.2	Must Have	Completed	
3.3	Must Have	Completed	
3.4	Must Have	Completed	
3.5	Must Have	Completed	
3.6	Must Have	Completed	
3.7	Should Have	Completed	
3.8	Could Have	Completed	
4.1	Could Have	Completed	
4.2	Could Have	Completed	
5.1	Will Not Have	Incomplete	This was a low priority requirement
5.2	Will Not Have	Incomplete	This was a low priority requirement
6.1	Will Not Have	Incomplete	This was a low priority requirement
6.2	Will Not Have	Incomplete	This was a low priority requirement
7.1	Will Not Have	Incomplete	This was a low priority requirement
7.2	Will Not Have	Incomplete	This was a low priority requirement
8.1	Must Have	Completed	

8.2	Must Have	Completed	
8.3	Must Have	Completed	
8.4	Must Have	Completed	
8.5	Must Have	Completed	
8.6	Must Have	Partially completed	The health and safety checks are marked as completed or not completed depending on if the user has submitted one. However, there is no status for in progress. This is the scenario where the user submits a partially complete health and safety check.
9.1	Will Not Have	Incomplete	This was a low priority requirement
10.1	Will Not Have	Incomplete	This was a low priority requirement
10.2	Will Not Have	Incomplete	This was a low priority requirement
11.1	Must Have	Completed	
11.2	Must Have	Completed	
11.3	Must Have	Completed	
11.4	Could Have	Partially completed	There is an overview of individual expenses, mileage and invoices per month. However, there isn't a total overview of expenses across a financial year.
11.5	Could Have	Completed	
11.6	Should Have	Completed	
11.7	Could Have	Completed	
12.1	Will Not Have	Incomplete	This was a low priority requirement
12.2	Will Not Have	Incomplete	This was a low priority requirement
12.3	Will Not Have	Incomplete	This was a low priority requirement
12.4	Will Not Have	Incomplete	This was a low priority requirement
13.1	Must Have	Completed	
14.1	Must Have	Completed	
15.1	Must Have	Completed	
16.1	Must Have	Completed	

Table 7.3: Requirements completion overview

7.3 Changing Requirements

7.3.1 Requirement 3.3: As a childcare provider, I want to record emergency contact information about each child.

During a show and tell session with his stakeholder, it was brought to Owen's attention that there would be three emergency contacts required for each child, rather than just one. This change was implemented within the application to allow for this.

7.3.2 Requirement 3.6: As a childcare provider, I want to have the ability to remove children from my application, should they no longer be in my care.

After discussing this requirement further with his stakeholder, Owen realised that removing a child from the application wasn't the best practice. Instead, a child's details should be kept inside the application to maintain the integrity of all historic data. This is because all data that a childcare provider holds regarding children within their care should be held until the child turns 22 years old. When a child is marked as inactive in the application, it prevents their name from appearing when submitting forms but allows the user to access historic data regarding the child.

7.3.3 Requirement 4.1: As a childcare provider, I want to have the ability to scan a food item and understand if it is healthy for a child to consume.

During a show and tell session, Owen was demonstrating the allergy detection component of the application. His stakeholder made a point that it would be beneficial to include an overview for the nutrition of the food item on the general tab. This would include salt, sugar, fat and saturated fat levels to give childcare providers valuable insights at a quick glance. This addition was implemented and added to the allergy detection component.

7.4 Objectives Validation

At the beginning of his project, Owen established objectives that he hoped to achieve throughout.

1. To understand and identify the key requirements of the stakeholder.

Owen met with his stakeholder on several occasions to build up an understanding of the requirements and vision that the stakeholder had for this solution. This helped Owen to understand the most important requirements that brought the most benefits to the stakeholder.

2. To refine a list of functional and non-functional requirements.

Owen refined the user's requirements into functional and non-functional requirements. He also converted them into user stories and prioritised them through using a value x complexity system and assigned them a category as per the MoSCoW method.

3. To evaluate and identify the most suitable software methodology for the project.

Owen evaluated several software methodologies including Scrum, Waterfall and XP. After balancing the pros and cons of each of these, Owen identified Scrum as the most suitable software methodology for his project.

4. To conduct a risk analysis on potential problems which may emerge during development, establishing a back-up plan or mitigating the risks.

Owen conducted a risk analysis to identify potential problems that he may have encountered during his project. Through identifying mitigations for each potential problem, this helped to reduce the likelihood of a problem occurring and minimise the severity that it may cause.

5. To research and identify the most suitable technology options to achieve the stakeholder's requirements.

Owen evaluated several technology options for his project. This included frameworks, IDEs, APIs and external modules. After completing his research, this allowed Owen to have a good understanding of his options and make suitable selections.

6. To construct a Gantt chart which will include details of sprint durations, key milestones and deadlines.

Owen constructed a Gantt chart which helped him to plan start and end dates for specific project tasks. This included preparation before development, sprint dates, show and tell meetings and other deliverables.

7. To build an application which serves as a central resource and reduces the complexities surrounding the administrative and maintenance tasks involved in the role of a childcare provider.

Owen successfully built an application that serves as a central resource. The application can be used to add, view, update and remove key information that childcare providers access daily. Through digitalising a wide range of tasks that would previously be completed on paper, Owen has built a tool that simplifies the administrative and maintenance tasks required of a childcare provider.

8. To carry out user testing on a range of individuals and record their feedback for future improvements

Owen carried out user tests on several people, this helped him to gain insights on his application and identify areas that he could improve upon through future development.

9. To produce a user guide that will enable the user to harness the application with as little reading as possible.

Owen created a user guide to provide guidance for users of his application. This includes a summary of what the application does and guides the user through how to use a range of different features.

7.5 Future Developments

There are several future developments that Owen has identified, should he have additional time to implement them. These suggestions have varied from a variety of sources incomplete user stories, stakeholder ideas within show and tell meetings and user testing feedback.

7.5.1 iOS concept

The concept that Owen developed runs on Android devices but through using React Native, it's possible to also release the same code onto iOS devices. Given additional time, Owen would like to also launch an iOS concept of this same application.

7.5.2 Notifications

One of the user stories that Owen didn't get the chance to complete due to its priority was to establish notifications within the application. These would support reminders for the health and safety checks that the user would complete on a daily or monthly basis. Given additional development time, Owen would like to implement these notifications.

7.5.3 Upload consent forms, contracts and parent confirmation of policy

This feedback was raised to Owen's attention through a show and tell meeting. Given additional development time, he would add components to the application which would allow the user to upload consent forms, contracts and parent confirmation of policy for each child in their care.

7.5.4 Additional form input types

Given additional development time, Owen would implement additional types of form input to replace plain text entry. Some examples of this include a postcode lookup when entering address information and a searchable text drop-down for selecting a medication.

7.5.5 Automatic numerical keyboard

Given additional development time, Owen would add functionality to input fields that automatically change the user's keyboard to numerical format when a numerical input is required.

7.5.6 Increase checkbox size

This feedback was raised to Owen's attention through user testing. Given additional development time, he would increase the size of checkboxes within the health and safety checks. This would make it easier for users to click on these and improve the speed that checks can be completed digitally.

7.5.7 'In Progress' status for health and safety checks

This feedback was raised to Owen's attention through user testing. Given additional development time, he would add an 'in progress' status to partially completed health and safety checks. This would help to inform the user that there are remaining checks to be completed.

7.5.8 Custom date range for financial overview

Given additional development time, Owen would add a feature to provide an overview of financial information over a date range specified by the user.

7.5.9 Address validation edge case scenarios

There were some areas where validation had been missed and this wasn't detected in initial testing. Owen aims to add extra layers to his validation to ensure that these edge cases are covered.

7.6 Innovative and Creative Features

There are several features that Owen developed for his application which he believes to be innovative and creative:

- Owen feels that the allergy detection component within his application is innovative as it allows the user to gain insights into a food item within seconds through only scanning a barcode. It cross-references the allergies of children actively in care and notifies the user if the food item is unsuitable for any children to consume.

- Owen feels that the receipt upload component is innovative as childcare providers no longer need to maintain receipts throughout a financial year. It also eliminates the risk of receipts going missing and the childcare provider being unable to complete some expenses. This is a paperless solution to replace a manual and tedious task.

- Owen feels that the emergency contact shortcut within the application is innovative as it improves the speed that a childcare provider can access details in the case on an emergency. Through separating these contacts

from the user's individual contacts list, it eliminates the need to store work-related contacts amongst personal contacts.

7.7 Performance Review

Overall, Owen is pleased with the overall outcome of his project and feels that he has gained a new level of understanding through applying the theory that he has covered throughout his course, in the form of this project. There were some technologies required during his project that required for Owen to upskill within key areas. Through utilising online documentation and guidance, this gave Owen an opportunity to be self-taught and put his learning into practice. Owen is delighted to have upskilled in a range of areas such as React Native, Firebase and mobile app development. He believes that going ahead, he could continue to apply the knowledge gained within future projects.

Owen felt that he planned his project extensively which allowed him to establish start and end dates for each component and provided him with a good routine of contributing work. Through detailed planning such as the tasks required in each sprint, this ensured that Owen knew exactly what he was working on throughout each sprint and the order of development.

Owen met with his project supervisor frequently at the beginning of the project, this began as a weekly meeting. This provided Owen with an opportunity to ask any questions regarding the project and gain valuable feedback on components that he had been working on. After moving onto the development stage, Owen became more independent with this work at so the frequency of the meetings with his supervisor became bi-weekly. This was mainly due to Owen focusing more on the feedback that he was gaining through show and tell meetings with his stakeholder.

7.8 Entrepreneurial & Commercial Opportunities

With over 2000 registered childcare providers currently operating in Northern Ireland, the demand for a solution to these issues increases continues to rise (familysupportNI, 2022). There is a huge commercial opportunity which remains untouched and idle. The childcare industry is in dire need to adjust to paperless and adopt technology to help support the fantastic work that these self-employed childcare providers carry out. Through speaking with multiple childcare providers, there is without doubt, a huge interest in an application such as this. Owen believes that this concept proves that there is value in an application such as this and with a little more work on future developments and looking into scalability, this application could be made available for public release.

8. Conclusion

As Owen reflects on upon the progress that he has made throughout the year, he is incredibly proud of the project that he has produced. Not only has Owen demonstrated the years of theory and understanding that he has gained through completing his Ulster University course, but he has also demonstrated his ability as a developer. Through learning how to use new technologies, upskilling in key areas when required and persevering through difficult challenges, this has helped Owen to gain knowledge in multiple areas and become a much more well-rounded developer. This is a project that allowed Owen to gain experience in mobile app development which was something that he previously had very little experience with. Going ahead, he feels much more confident working mobile app development such as this and would feel comfortable to be involved with similar projects in the future.

References

AppBrain, 2022. *React Native.* [Online]
Available at:
https://www.appbrain.com/stats/libraries/details/react_native/react-native
codemagic, 2021. *Choosing the right database for your React Native app.* [Online]
Available at: https://blog.codemagic.io/choosing-the-right-database-for-react-native-app/
Cohn, M., 2007. *Differences Between Scrum and Extreme Programming.* [Online]
Available at: https://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming
Edamam, n.d. *Food Database API Documentation.* [Online]
Available at: https://developer.edamam.com/food-database-api-docs
Expo, 2022. *Introduction to Expo.* [Online]
Available at: https://docs.expo.dev/
familysupportNI, 2022. *Search Results in Childminder Category.* [Online]
Available at:
https://www.familysupportni.gov.uk/Search/Results?FirstSearch=1&sTypeID=138&serviceID=153
Firebase, 2022. *Cloud Firestore.* [Online]
Available at: https://firebase.google.com/docs/firestore
Firebase, 2022. *Cloud Storage for Firebase.* [Online]
Available at: https://firebase.google.com/docs/storage
Firebase, 2022. *Firebase Realtime Database.* [Online]
Available at: https://firebase.google.com/docs/database
Lucid Content Team, n.d. *Scrum for one: A tutorial on adapting Agile Scrum methodology for individuals.* [Online]
Available at: https://www.lucidchart.com/blog/scrum-for-one
Mozilla, n.d. *Progressive web apps (PWAs).* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
Native, R., 2016. *DECO, THE REACT NATIVE IDE: NOW FREE AND OPEN SOURCE!.* [Online]
Available at: https://www.reactnative.com/deco-the-react-native-ide-now-free-and-open-source
Open Food Facts, n.d. *Open your food and know what you eat.* [Online]
Available at: https://world.openfoodfacts.org/discover
React Native, 2022. *Who's using React Native?.* [Online]
Available at: https://reactnative.dev/showcase
Rehkopf, M., n.d. *User stories with examples and a template.* [Online]
Available at: https://www.atlassian.com/agile/project-management/user-stories
Stevenson, D., 2018. *What is Firebase? The complete story, abridged..* [Online]

Available at: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

Appendices

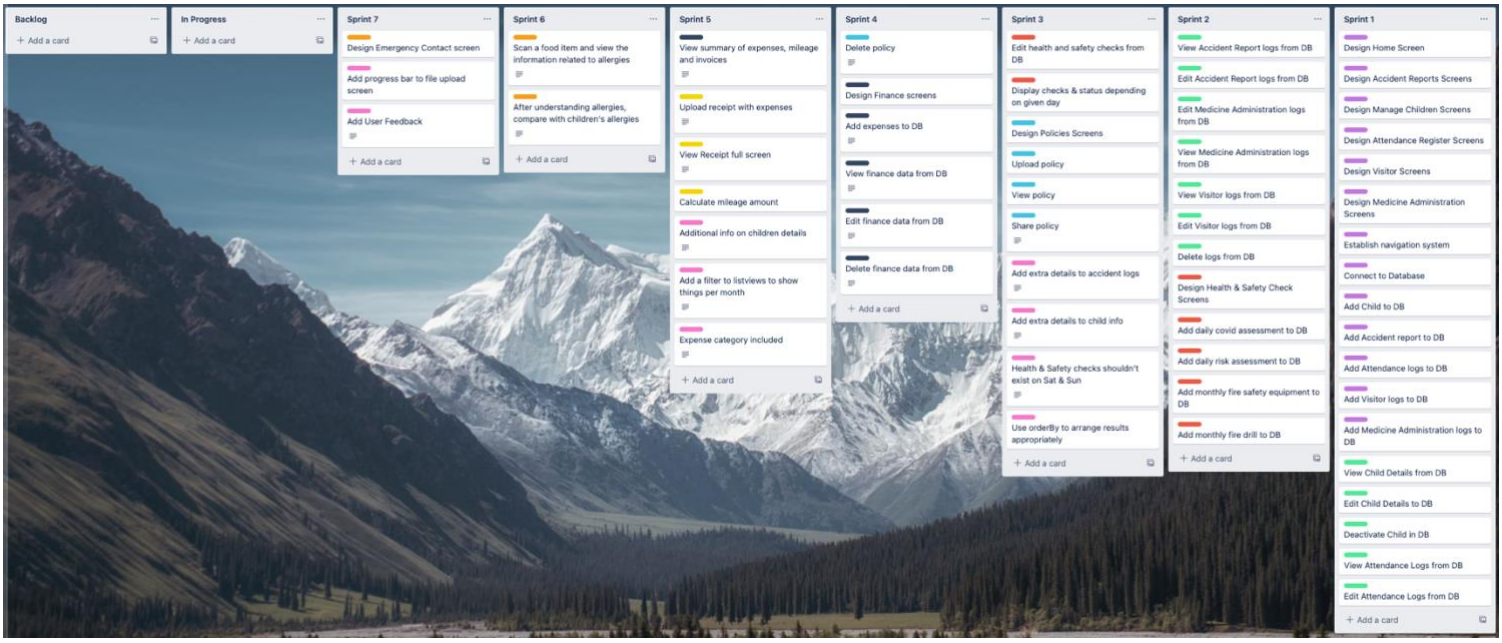
A. GitHub repository

Owen's GitHub repository can be accessed at this URL:
https://github.com/owenatkinson/childcare-management-app

The repository is currently private though access can be requested and granted through adding additional users as collaborators.

B. GitHub network graph

C. Trello Board



D. Full Requirements List

ID	Requirement	Complexity	Benefit	Total
1.1	As a childcare provider, I want to log visitor information. This will ensure that there is a digital record created for all visitor logs, should it need to be referenced in the future.	5	5	25
1.2	As a childcare provider, I want to log medicine administration about each child. This will ensure that there is a digital record created for all medicine administration logs, should it need to be referenced in the future.	5	5	25
1.3	As a childcare provider, I want to log daily attendance information about each child. This will ensure that there is a digital record created for all daily attendance logs, should it need to be referenced in the future.	5	5	25
1.4	As a childcare provider, I want to log accident reports if an incident occurs. This will ensure that there is a digital record created for all accident report logs, should it need to be referenced in the future.	5	5	25
1.5	As a childcare provider, I want to have the ability to allow parents to sign forms electronically. This will allow the digitalisation of form signing entirely, avoiding the need to print and acquire a hand-written signature from the parent which must then be stored physically.	2	4	8
1.6	As a childcare provider, I want to have the ability to edit the information that I have logged. This will	4	5	20

	allow me to correct any errors that may have been entered to ensure that information is maintained as accurately as possible.			
1.7	As a childcare provider, I want to have the ability to view any of the information that I have submitted via forms. This will allow me to review details of recorded logs from the past.	3	5	15
1.8	As a childcare provider, I want to have the ability to view my logged data through filters. This will allow me to easily identify data, when navigating through large quantities.	3	5	15
1.9	As a childcare provider, I want to have the ability to delete the information that I have submitted via forms. This will allow me to remove any unwanted details from the application.	5	5	25
2.1	As a childcare provider, I want to have the ability to upload policies from my device onto the application. This will allow me to upload and store documents for future access.	3	5	15
2.2	As a childcare provider, I want to have the ability to delete my policies. This will allow me to remove any documents which no longer need to be stored for reference.	3	5	15
2.3	As a childcare provider, I want to have the ability to view my policies. This will allow me to digitally view documents at any time with easy access.	3	5	15
2.4	As a childcare provider, I want to have the ability to share my policies. This will allow me to have a simple method of sharing relevant documents externally with others.	4	4	16
3.1	As a childcare provider, I want to record dietary requirements about each child. This ensures that an accurate record has been made for dietary requirement information, should it need to be referenced in the future.	5	5	25
3.2	As a childcare provider, I want to record medical information about each child. The childcare provider can record medical information for children in their care, this data is stored in a database.	5	5	25
3.3	As a childcare provider, I want to record emergency contact information about each child. This ensures that an accurate record has been made for emergency contact information, should it need to be referenced in the future.	5	5	25
3.4	As a childcare provider, I want to have the ability to update this information about each child should	4	5	20

	their circumstances change throughout their time in my care. This allows me to correct any errors that may have been entered to ensure that information is maintained as accurately as possible.			
3.5	As a childcare provider, I want to have the ability to add additional children to the application. This ensures that I can keep the application updated with all children currently in my care.	5	5	25
3.6	As a childcare provider, I want to have the ability to remove children from my application, should they no longer be in my care. This ensures that I can keep the application updated with all children currently in my care.	5	5	25
3.7	As a childcare provider, I want to have the ability to view child details should I need to retrieve this information. This allows me to review details of recorded logs from the past.	3	5	15
3.8	As a childcare provider, I want to have the ability to phone emergency contacts directly through the application. This allows me to have quick access to emergency contact information and avoids having to mix these details with my personal contacts.	4	3	12
4.1	As a childcare provider, I want to have the ability to scan a food item and understand if it is healthy for a child to consume. This will improve the speed that I can carry out these checks to ensure the children in my care are eating food which is healthy.	3	4	12
4.2	As a childcare provider, I want to have the ability to scan a food item and understand if it is unsuitable for a child to consume due to their allergies. This will improve the speed that I can carry out these checks for the safety of the children in my care.	3	4	12
5.1	As a childcare provider, I want to have the ability to upload images from day trips. This simplifies the process of manually messaging each family a copy of these images, instead only one upload is required.	2	2	4
5.2	As a childcare provider, I want to have an automated tool that shares images from day trips with parents. This simplifies the existing process of having to manually select which photos to share with the relevant parents.	2	2	4
6.1	As a childcare inspector, I want to have read-only access of the childcare provider's health & safety	2	3	6

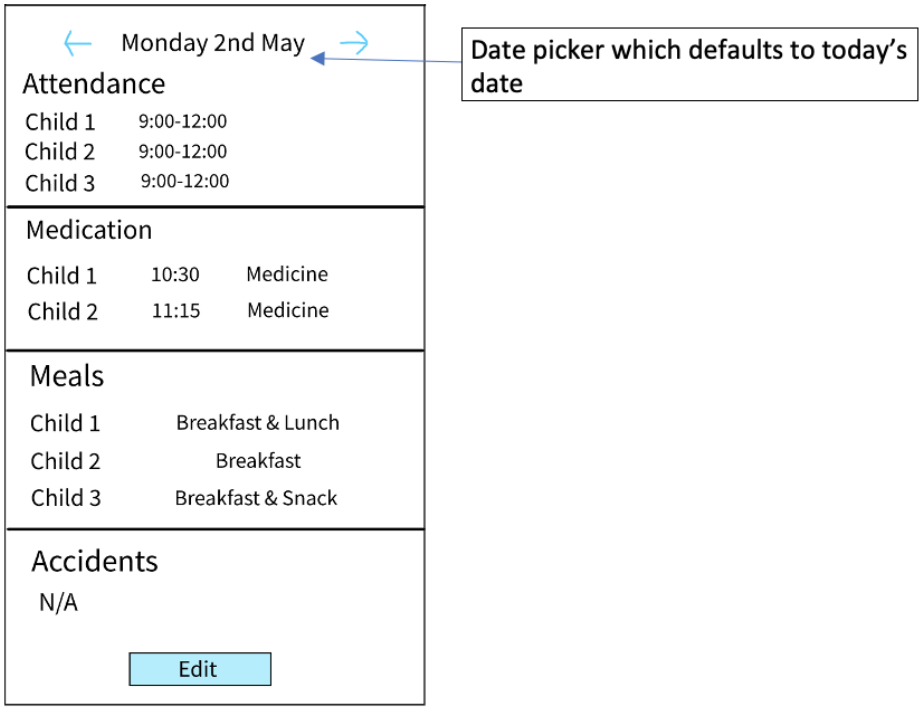
	information. This allows me to complete components of the annual assessment before arriving at the premises.			
6.2	As a childcare inspector, I want to have read-only access of the childcare provider's attendance information. This allows me to complete components of the annual assessment before arriving at the premises.	2	3	6
7.1	As a parent, I want to have read-only access of my child's personal information. This allows me to have a record of the childcare provider's details of my child, should I need to raise any areas that need updated.	2	2	4
7.2	As a parent, I want to have read-only access of my child's daily attendance reports. This allows me to have a record of relevant detail for my child throughout a given day.	2	2	4
8.1	As a childcare provider, I want to be provided with a form to guide me through my daily risk assessments. This will improve the accuracy and speed at which I can carry out daily risk assessments, ensuring that the safety of the children in my care is maximised.	4	5	20
8.2	As a childcare provider, I want to be provided with a form to guide me through my daily COVID checks. This will improve the accuracy and speed at which I can carry out daily COVID checks, ensuring that the safety of the children in my care is maximised.	4	5	20
8.3	As a childcare provider, I want to be provided with a form to guide me through my monthly fire drills. This will improve the accuracy and speed at which I can carry out monthly fire drills, ensuring that the safety of the children in my care is maximised.	4	5	20
8.4	As a childcare provider, I want to be provided with a form to guide me through my monthly fire safety equipment check. This will improve the accuracy and speed at which I can carry out monthly fire safety equipment checks, ensuring that the safety of the children in my care is maximised.	4	5	20
8.5	As a childcare provider, I want to have the ability to edit information that I have logged in my health and safety checks. This allows the childcare provider to correct any errors that may have been entered during health and safety checks to ensure that information is maintained as accurately as possible.	4	5	20

8.6	As a childcare provider, I want to view the status of my health and safety checks. This will help me to understand and identify any checks that I may have missed.	4	5	20
9.1	As a childcare provider, I want to be provided with reminders for my daily and monthly & safety checks. This will ensure that I have carried out the correct safety measures that I am required to complete.	2	4	8
10.1	As a childcare provider, I want to increase the speed that repetitive logs can be entered by using previous information to predict my input. This will reduce the time that it takes for logs to be submitted.	1	3	3
10.2	As a childcare provider, I want to reduce the error rate within my logged information. The application should detect potential errors from the information that I have logged. This will prevent invalid data from being submitted within the application and stored within a database.	1	3	3
11.1	As a childcare provider, I want to have the ability to track mileage expenses. Having a digital record of mileage expenses will make it much easier to calculate my total expenses at the end of the financial period.	4	5	20
11.2	As a childcare provider, I want to have the ability to track invoices. Having a digital record of invoices will make it much easier to calculate my total income at the end of the financial period.	4	5	20
11.3	As a childcare provider, I want to have the ability to track other relevant expenses (equipment, gifts, fuel etc). Having a digital record of all other expenses will make it much easier to calculate my total expenses at the end of the financial period.	4	5	20
11.4	As a childcare provider, I want to have access to an overview of my expenses. This will provide me with an overall understanding and breakdown of my expenses.	3	4	12
11.5	As a childcare provider, I want to have the ability to upload an image of a receipt. This will allow me to have a digital reference to the expense that I have made and means that I won't need to hold onto large amounts of physical receipts.	3	4	12
11.6	As a childcare provider, I want to have the ability to view the receipts that I have previously uploaded. This will allow me to view the details of expenses that I have made.	4	4	16

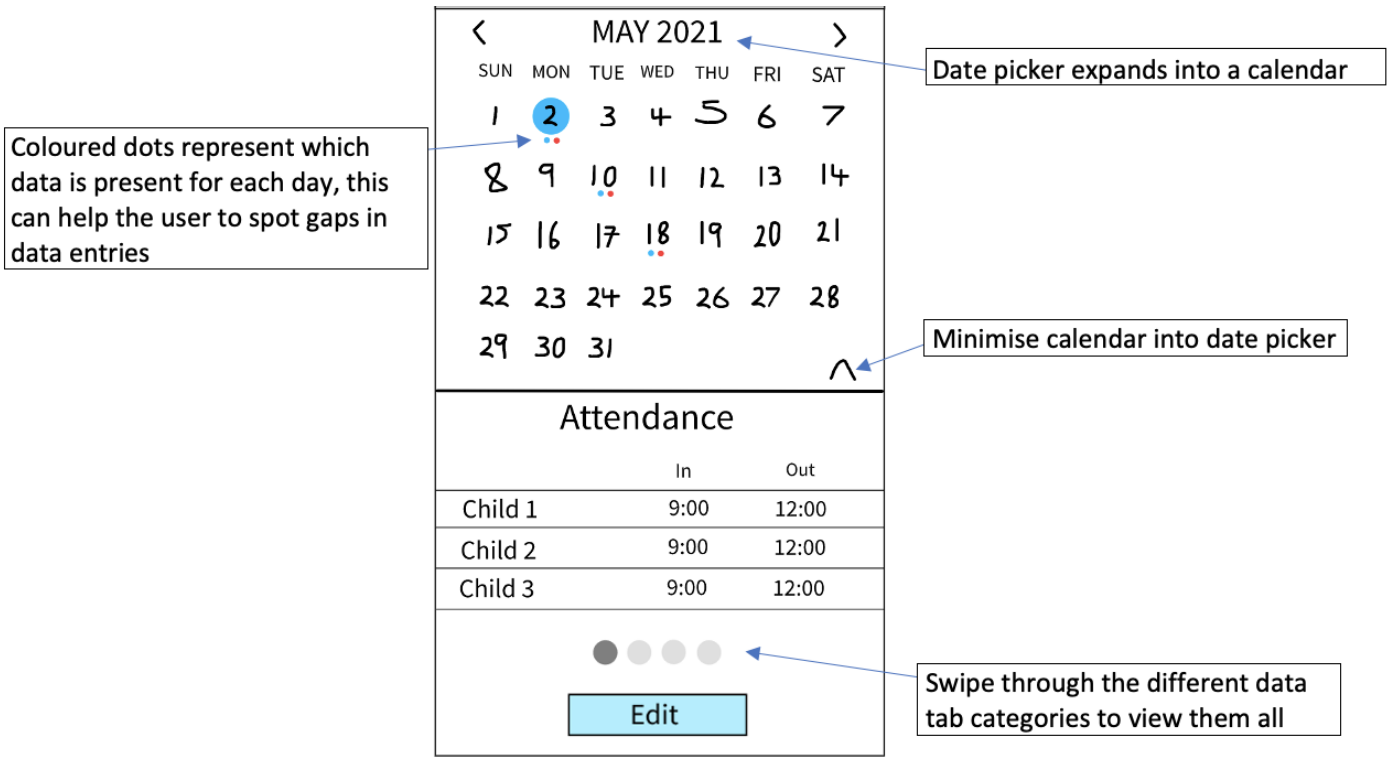
11.7	As a childcare provider, I want to have the ability to calculate my mileage expenses within the application. This will improve the speed that I can complete expenses, allowing me to focus more attention on other areas of my role.	3	4	12
12.1	As a childcare provider, I want to have a planning feature that allows me to add notable events such as children's birthdays, inspection dates, renewal dates, etc. This will help me to prepare for these events with the required preparation and help me to maintain a separation between my private calendar used for non-work-related events.	2	3	6
12.2	As a childcare provider, I want to have the ability to edit events within the planning feature. This will allow me to amend the details of the event should the date, time or details need to be changed.	2	3	6
12.3	As a childcare provider, I want to have the ability to delete events within the planning feature. This will allow me to remove any events planner should I require to do so.	2	3	6
12.4	As a childcare provider, I want to receive notifications for these planned events. This will notify me when these events are upcoming, ensuring that I have considered the required preparation.	2	3	6

E. Additional wireframe designs

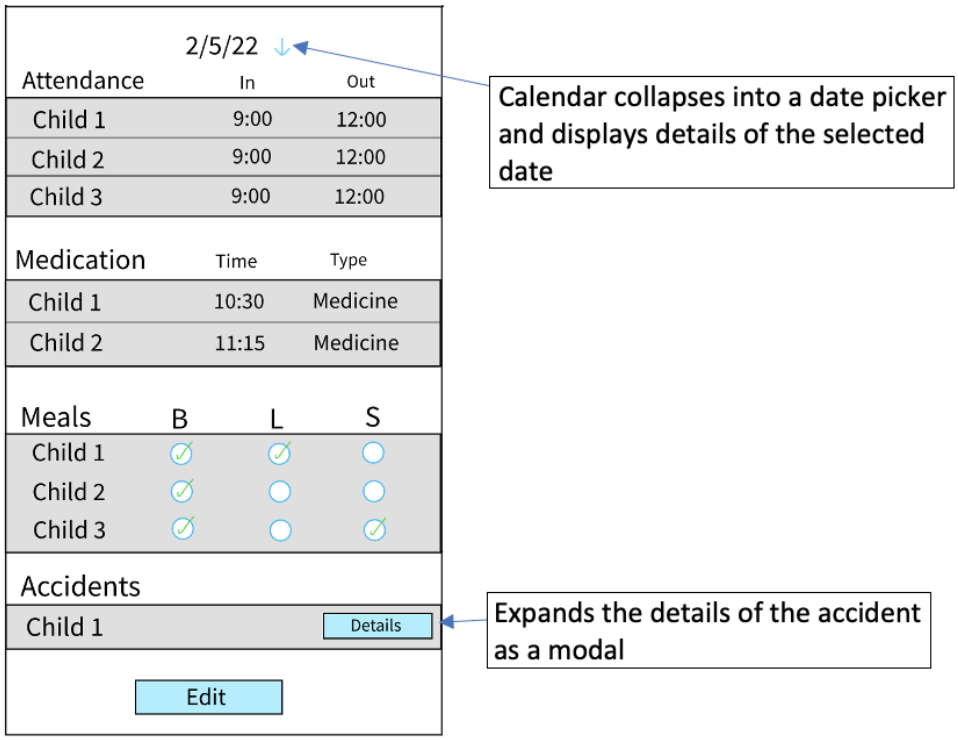
Design 1



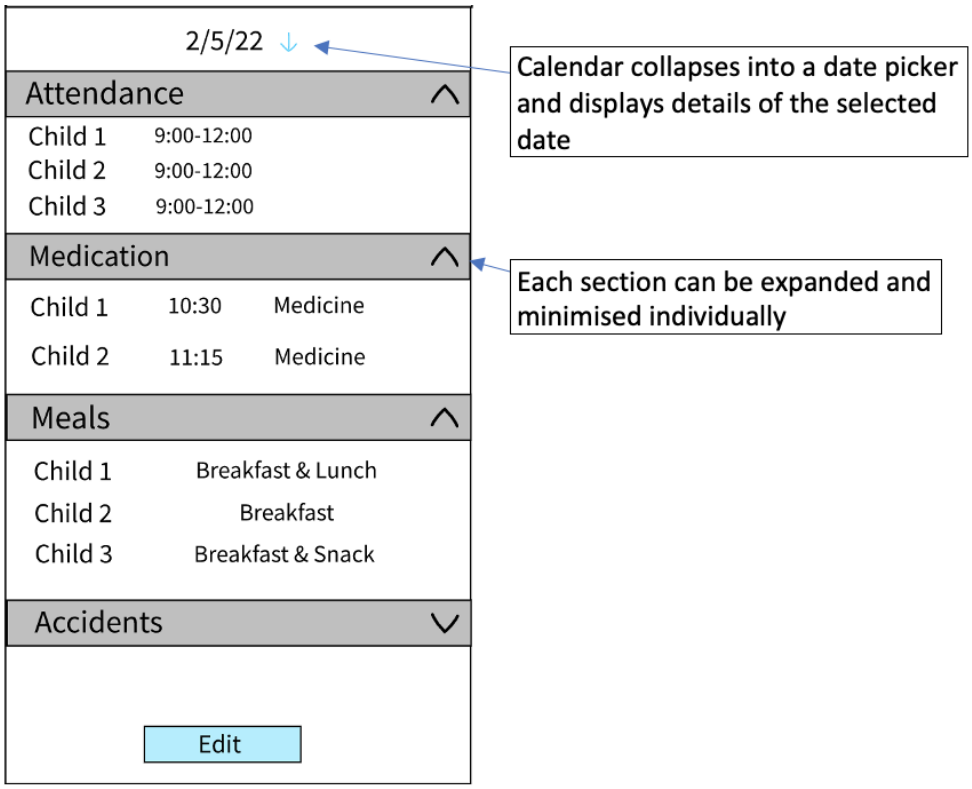
Design 2



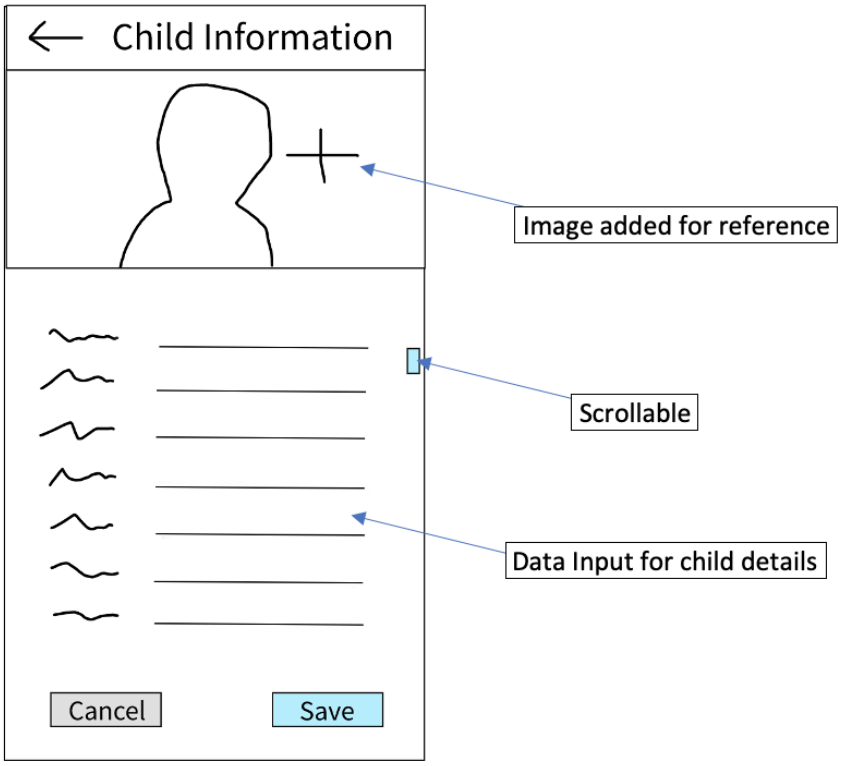
Design 3


2/5/22
Attendance
In
Out
Child 1
9:00
12:00
Child 2
9:00
12:00
Child 3
9:00
12:00
Medication
Time
Type
Child 1
10:30
Medicine
Child 2
11:15
Medicine
Meals
B
L
S
Child 1
Child 2
Child 3
Accidents
Child 1
Details
Edit
Calendar collapses into a date picker and displays details of the selected date
Expands the details of the accident as a modal

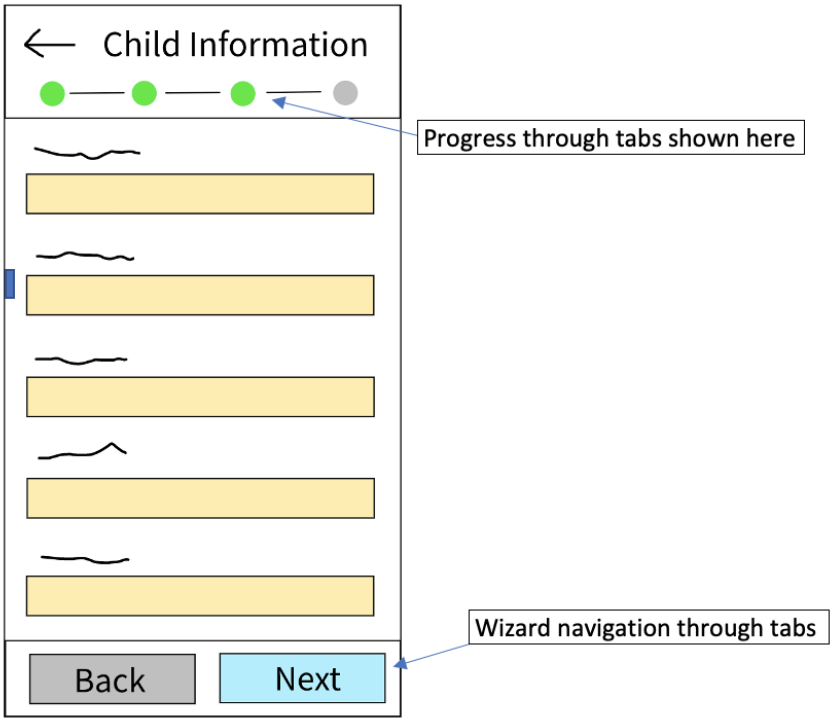
Design 4


2/5/22
Attendance
Child 1 9:00-12:00
Child 2 9:00-12:00
Child 3 9:00-12:00
Medication
Child 1 10:30 Medicine
Child 2 11:15 Medicine
Meals
Child 1 Breakfast & Lunch
Child 2 Breakfast
Child 3 Breakfast & Snack
Accidents
Edit
Calendar collapses into a date picker and displays details of the selected date
Each section can be expanded and minimised individually

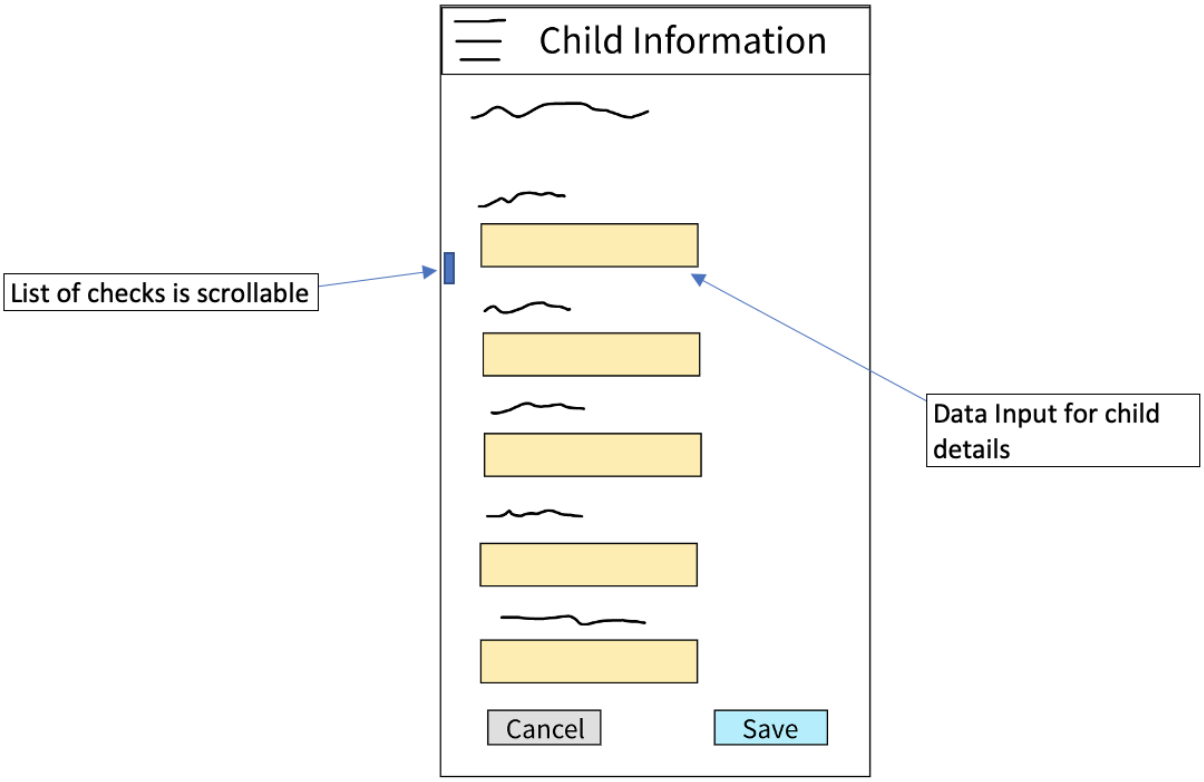
Design 1



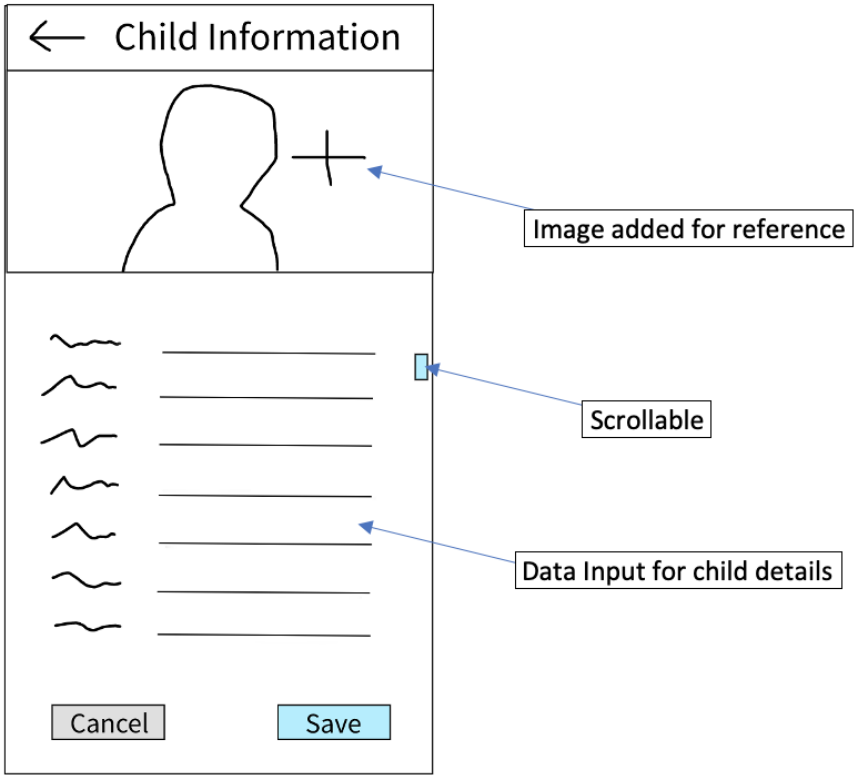
Design 2



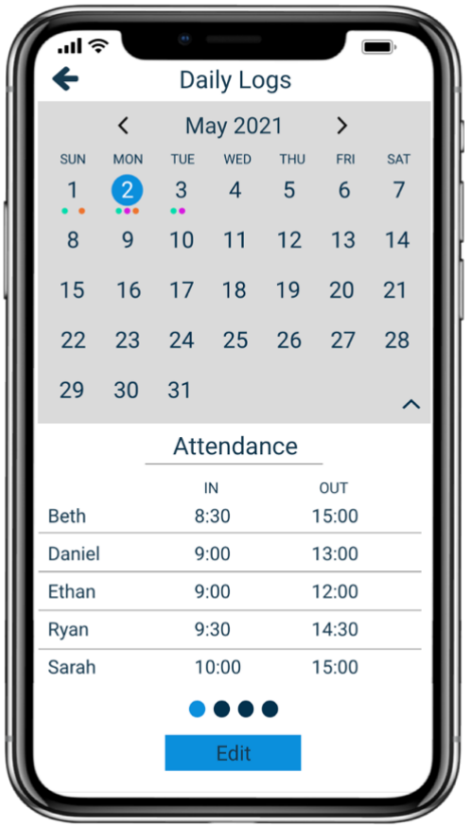
Design 3

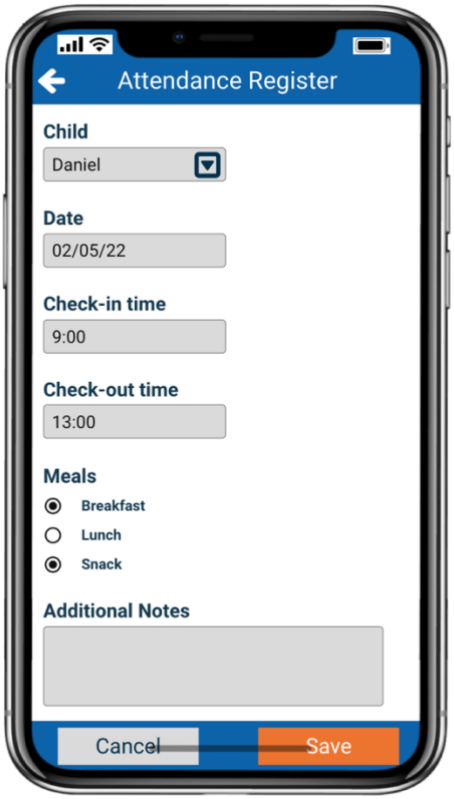


Design 4



F. Additional advanced designs





Allergy Detection

Chocolate Milkshake

Allergens

Milk

Suitable for

Beth
Ethan
Daniel
Ryan
Stuart

Not suitable for

Sarah
Helen

View Visitor Log

Edit Log
Remove Log

Visitor Name & Company
Dennis Sanders

Date
02/05/22

Start Time of Visit
14:00

End Time of Visit
14:20

Purpose of Visit
Electrician

data-protection.pdf

Adobe Acrobat PDF Files

Adobe® Portable Document Format (PDF) is a universal file format that preserves all of the fonts, formatting, colours and graphics of any source document, regardless of the application and platform used to create it.

Adobe PDF is an ideal format for electronic document distribution as it overcomes the problems commonly encountered with electronic file sharing.

- ***Anyone, anywhere*** can open a PDF file. All you need is the free Adobe Acrobat Reader. Recipients of other file formats sometimes can't open files because they don't have the applications used to create the documents.
- PDF files ***always print correctly*** on any printing device.
- PDF files always display ***exactly*** as created, regardless of fonts, software, and operating systems. Fonts, and graphics are not lost due to platform, software, and version incompatibilities.
- The free Acrobat Reader is easy to download and can be freely distributed by anyone.
- Compact PDF files are smaller than their source files and download a page at a time for fast display on the Web.

Attendance Register

Child
Daniel

Date
02/05/22

Check-in time
9:00

Check-out time
13:00

Meals
- (•) Breakfast
- ( ) Lunch
- (•) Snack

Additional Notes

Cancel | Save

G. User surveys answers

Childcare Management Application User Survey

1. Has your occupation ever been a childcare provider? Yes No

2. How would you describe your technical ability?

1	2	3	4	5
(Not technical)		(Average)		(Very technical)

3. What age are you? 49

4. What is your gender? Male Female

5. How easy was it to understand the features of the application and how they worked?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

6. How useful was the user guide at explaining the features of the application?

1	2	3	4	5
(Not useful)		(Average)		(Very useful)

7. Was there any key information missing from the user guide? If so, what was missing?
Icons used to categorise the different sections of the user guide to correspond to the home page icons of the app.

8. How easy was it to understand the navigation of the application?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

9. How would you rate the layout, design and overall appearance of the application?

1	2	3	4	5
(Very poor)		(Average)		(Very good)

10. What do you feel is the most valuable feature of the application?

The fact that instead of going through child records, I can access anything at the click of the button, especially in the scenario where an emergency occurs. The ability to extract information from specific dates or for a specific child made it very easy to access. I would usually need to find my files and look through pages of information to find what I need. The speed that I can get my hands on this information makes my job much easier.

11. Are there any improvements that you would make to the application? If so, what changes would you make?

Including a section that allows me to upload consent forms, contracts and parent confirmation of policies. These would be valuable to have on hand.

12. Do you believe this application is useful for a childcare provider? Yes No

Childcare Management Application User Survey

1. Has your occupation ever been a childcare provider? Yes **No**

2. How would you describe your technical ability?

1	2	3	4	5
(Not technical)		(Average)		(Very technical)

3. What age are you? **24**

4. What is your gender? Male Female

5. How easy was it to understand the features of the application and how they worked?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

6. How useful was the user guide at explaining the features of the application?

1	2	3	4	5
(Not useful)		(Average)		(Very useful)

7. Was there any key information missing from the user guide? If so, what was missing? **No**

8. How easy was it to understand the navigation of the application?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

9. How would you rate the layout, design and overall appearance of the application?

1	2	3	4	5
(Very poor)		(Average)		(Very good)

10. What do you feel is the most valuable feature of the application?

The allergy detection is a great tool for someone to find out if a child is allergic to a certain food item. It's a quick and easy way for the child carer to receive an answer.

11. Are there any improvements that you would make to the application? If so, what changes would you make?

It would be great if the keyboard automatically changed to numerical values when the forms require a numerical input, this would speed up the process. It would also be useful to increase the size of the checkboxes within the health and safety checks as it would be easier to click on these.

12. Do you believe this application is useful for a childcare provider? Yes No

Childcare Management Application User Survey

1. Has your occupation ever been a childcare provider? Yes No

2. How would you describe your technical ability?

1	2	3	4	5
(Not technical)		(Average)		(Very technical)

3. What age are you? **62**

4. What is your gender? Male Female

5. How easy was it to understand the features of the application and how they worked?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

6. How useful was the user guide at explaining the features of the application?

1	2	3	4	5
(Not useful)		(Average)		(Very useful)

7. Was there any key information missing from the user guide? If so, what was missing?
N/A

8. How easy was it to understand the navigation of the application?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

9. How would you rate the layout, design and overall appearance of the application?

1	2	3	4	5
(Very poor)		(Average)		(Very good)

10. What do you feel is the most valuable feature of the application?
Having one main place for all documents is a big positive. I also really like that it's possible to upload your receipts, so that you could refer back to them for future reference e.g. when calculating your expenses at the end of the tax year. I also like how it would cut down the paper wastage.

11. Are there any improvements that you would make to the application? If so, what changes would you make?
I think that this app is very good! It could potentially benefit from having a system in place that would allow parents/guardians to electronically sign the medications administered to verify that parental consent was received.

12. Do you believe this application is useful for a childcare provider? Yes No

Childcare Management Application User Survey

1. Has your occupation ever been a childcare provider? Yes **No**

2. How would you describe your technical ability?

1	2	3	4	5
(Not technical)		(Average)		(Very technical)

3. What age are you? **23**

4. What is your gender? Male Female

5. How easy was it to understand the features of the application and how they worked?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

6. How useful was the user guide at explaining the features of the application?

1	2	3	4	5
(Not useful)		(Average)		(Very useful)

7. Was there any key information missing from the user guide? If so, what was missing?
N/A

8. How easy was it to understand the navigation of the application?

1	2	3	4	5
(Very difficult)		(Average)		(Very easy)

9. How would you rate the layout, design and overall appearance of the application?

1	2	3	4	5
(Very poor)		(Average)		(Very good)

10. What do you feel is the most valuable feature of the application?
The concept of having all information in one location and not requiring any paperwork to be printed and stored. The ability to share across multiple devices.

11. Are there any improvements that you would make to the application? If so, what changes would you make?
I would add an address drop down based on the postcode for users to select the appropriate address (less room for error when entering in an address). I would also potentially add an autofill feature to the medications administered section, again to avoid error in name of the medicine.

12. Do you believe this application is useful for a childcare provider? Yes No