CM2020 Agile Software Projects
Final Project Report

Lucille: A To-Do App for the Elderly

Tutor Group 3, Team 18, students nos.:

- 200406440
- 200422923
- 200464026
- 200510091
- 200521248

09/05/2022

Contents

Introduction

Background

During the first half of this module we recognized a gap in two overlapping markets: the to-do app market and the elderly care app market,[1] which we chose to fill by designing *Lucille*, a "to-do" app for elderly users.[2] We analyzed available literature on usability design for the elderly, and extracted a series of design recommendations; using these recommendations we designed several prototypes, the last of which we proceeded to produce during the latter half of the semester.[3] In this report we explain how we approached the planning, production and evaluation of this prototype; analyze the process and its outcomes; and suggest steps for future improvement.

Assumptions

We assume familiarity with the midterm report (Sharma *et al.*, 2022), Agile Manifesto and principles (Beck *et al.*, 2001), Scrum Guide (Schwaber and Sutherland, 2020), as well as the principles of user-centered design as laid out by Gould and Lewis (Gould and Lewis, 1985), and elaborated on by Sharp, Rogers and Preece (Sharp, Rogers and Preece, 2019).

[1] Here we use the terms "seniors", "elderly" and "older adults" interchangeably to denote persons aged 65 years and older.
[2] The name "Lucille" was chosen both for its sound and age-appropriateness, and as a tongue-in-cheek reference to Lucille Bluth, the overbearing mother figure played by Jessica Walter in the TV series *Arrested Development* ('Arrested Development', 2003).
[3] For the complete market analysis, literature review and design series, see (Sharma *et al.*, 2022).

Project planning, management and execution

Introduction

The purpose of this section is to present the tools and processes that were used during the development of the application; explain the planning and management considerations that underpinned the project; trace the path and timeline that the project took; and analyze the reasons and dynamics that lead to it.

For the initial project plan and associated timeline, see (Sharma *et al.*, 2022, chap. Project plan).

Team background

Our decisions throughout the process were heavily influenced by our respective backgrounds, and our understanding of each other's abilities. We provide our academic and professional details below, so that the reader may better understand our thinking at various stages of the project:

- Rodi Ali, B. Eng. Engineering design, a Cloud Engineer at TryDig AS
- Takahisa Hashimoto, BSc. Mechanical Engineering, a Senior Expert at the BMW Group
- Isa Phuyuthanon, BSc. Business Administration, a full time student
- Morag Scheinwald, a full time student
- Anugya Sharma, BArch Architectural Engineering, a full time student

Tracing the project's evolution

In retrospect, the project timeline can be divided into four periods: towards the midterms (05-10-22–06-27-22); immediately after the midterms (06-28-22–07-23-22); the realignment period (07-24-22–08-15-22) and the production period (08-16-22–09-04-22).

Towards the midterms (05-10-22–06-27-22)

The goal at this stage of the work was to produce a viable project proposal for the midterms. The proposal was to be written as follows:

- Primary author: Ms. Sharma
- Literature research and secondary author: Mr. Scheinwald
- Preliminary testing and ethical review: Mr. Phuyuthanon
- Summary and analysis of market research: Mr. Hashimoto
- Use cases, technical and security specification, and project plan: Mr. Ali

At this point in time the project had no clear leadership; decisions were reached by vote, and work allocation was on an ability-and-availability basis. We were meeting once a week for discussion, and keeping in touch on Slack in-between.

Some of the key decisions made at this point were to create a “to-do” app, and to use a software stack based on React and Amazon Web Services (AWS). Both of these decisions were informed by the prevalence of the technologies, the availability of online resources (documentation, tutorials, project templates, etc.), and the professional experience of one of the team members. Our unique “twist” on this theme, however, came as a result of brainstorming on the societal and environmental problems faced in our respective countries; we recognized the need to find technological solutions for the needs of an aging population - a realization that was corroborated by a subsequent market analysis and literature review.

Shortly before submission it became clear that the proposal lacked a project plan and key parts of the specification, which were then hastily written by the principal authors. Nevertheless, the proposal was well accepted, and provided a solid basis for production.

An important lesson from this phase of the work was that we should increase our use of Jira, a task management system for teams, to better allocate and follow-up on tasks.

After the midterms (06-28-22–07-23-22)

The period immediately following the midterms was challenging. Having agreed to use a software stack based on React and AWS, members made significant efforts learning the technologies through tutorials, books, API documentation, and other such materials. An effort was made to locate a suitable “starter template” for the project, but the one that was chosen proved too complicated. Mr. Ali, the developer most experienced with React, was on vacation until mid-July, so was unavailable to assist.

Towards the end of this period two important decisions were made: the first was to accept a clear division of labor, for the first time since the project began:

- Product owner: Mr. Scheinwald
- Documentation and design: Ms. Sharma
- Tech lead: Mr. Ali
- Testing and analysis: Mr. Hashimoto
- User testing: Mr. Phuyuthanon

The thinking underlying this division was to provide members with clear areas of expertise that suit their respective skill sets, so as to:

1. Optimize time and effort spent
2. Give members a sense of ownership and pride at their work
3. Improve accountability
4. Allow for easier task allocation and management

The second decision was to continue working with React. After researching and discussing alternatives such as Svelte (a React alternative), Mithril.js (a minimalist client-side SPA framework) and Chakra-UI (an easy-to-use React component library, which eventually formed the basis of our prototype), we decided that despite the initial difficulties, React might still prove viable and valuable. It required, however, that we leave the bulk of the “business logic” to our

most experienced developer. This was a risky decision, but if it had worked it would have left the rest of the team free to focus on our main value proposition, which is the user experience.
At this point in time it was also decided to switch to a different template that used a different set of AWS services; it deviated from the original design which prohibited login screens,[4] but was supposed to make development easier and faster, so was accepted.

Realignment (07-24-22–08-15-22)

This period began with analysis of the project structure and goals (see fig. 8 - Work breakdown on 07/27/2022), and the presentation of a work plan by the product owner. This included an order of priorities for development, and a Jira "roadplan" with various milestones (for a summary of the roadplan, see Fig. 0 below). We began working in "sprints" 7-14 days long (in some cases less), with weekly or twice-weekly follow-up, depending on the need.

Having divided responsibilities and configured the template for development, the team started re-working the code. Unfortunately, this too proved too complex: not only did the new template use React, but it also used TypeScript and Redux - two technologies that only the tech lead was familiar with - and he was away on business until mid-August.

Towards the end of this period, seeing as no progress was being made with the current model, the product owner presented a new analysis (see fig. 9 - Work breakdown on 08/10/2022), and a plan for pivoting away from the then-current tech stack towards one that is simpler and more feasible (see Appendix G: Technology stack proposal). To explore alternatives, two prototypes were demonstrated: one that relied on React, Chakra-UI and PouchDB (a simple database library); and another based on "vanilla" JS and client-side templating with the EJS library, which was familiar to team members from a previous module. Member then took a few to experiment, with Ms. Sharma working on the Chakra-based prototype, and Mr. Hashimoto working on the EJS-based prototype. Eventually the former won on technical grounds (it was more suitable for rendering dynamic apps), and became the basis for our current codebase.

Production (08-16-22–09-04-22)

Production began in earnest in mid-August, with the tech lead refactoring the existing codebase into a more colloquial React form. A contingency was put in place in case development would stall, with one of the other members who was familiar with React being asked to review the code base as it was being worked on, in case they would need to take over and continue development on their own.

The initial prototypes deviated quite significantly from the original design, but after some intense discussions, re-prioritization of features, several cycles of testing led by Mr. Hashimoto and Mr. Phuyuthanon, and some design solutions by Ms. Sharma, the current version of the app comes close to realizing the intent of the design.

---

[4] See Literature review and design recommendations in (Sharma *et al.*, 2022)

Work division, methodology, tools and artifacts

Work division

As of the writing of this report, the team is structured as follows:

- Product owner: Mr. Scheinwald
- Documentation and design: Ms. Sharma
- Programming: Mr. Ali
- User testing and analysis: Mr. Hashimoto
- User testing and design: Mr. Phuyuthanon

Development methodology

The team used a tailored development process loaning heavily from the Scrum methodology (Schwaber and Sutherland, 2020) and user-centered design (Sharp, Rogers and Preece, 2019, sec. What Is a User-Centered Approach?). The process consisted of the following:

- A flexible team structure where members have different specialities, but may engage in different kinds of tasks as needed
- A product owner that plans and prioritizes development goals, and assigns tasks in agreement with team members
- An iterative, user-centered design process that divides the long term goals of the project into short production cycles, each followed by an evaluation stage with real users
- An emphasis on the human aspects of development: clear and frequent communication, collaborative ownership, developer happiness and sense of fulfillment

An analysis of the relation between our chosen process and “traditional” development methodologies is given in the section “Lessons learned and reflections on Agile methodology” below.

Tools and artifacts

Lucille

The team produced *Lucille*, a “to-do” app for older users. Lucille was written using JavaScript, React (‘React’, 2022), Chakra UI (Adebayo, 2022) and Dexie.js (Fahlander, 2022) as its main components, and constitutes roughly 1,800 lines of code (LOC).

A live instance of Lucille is available here.

Git / GitHub

Git (‘Git’, 2022) and GitHub (‘GitHub’, 2022) were used to track and review changes, and deploy code. A total of 125 commits were made to the repository.

A complete Git log was attached in field 2 of this submission.

Jira

Atlassian's Jira ('Jira', 2022) was used to structure the project, schedule and assign tasks, and follow up on completion. A total of 138 issues were created, of which 92 were completed.

A complete Jira task record was attached in field 2 of this submission.

Slack

A private Slack channel ('Slack', 2022) was used for frequent communication on various matters. A total of 1,165 messages and 104 attachments were exchanged.

Zoom

Zoom ('Zoom', 2022) was used for weekly or twice-weekly meetings, 7 hours and 44 minutes of which were recorded and uploaded to the team's Google Drive.

Google Drive

Google Drive ('Google Drive', 2022) was used for storage of documents, recordings and memo throughout the lifetime of the project. A total of 143 files constituting some 1.16 GB were stored on the drive.

Lessons learned and reflections on Agile methodology

Throughout the development of this project the team encountered many challenges, some of which have no clear answer. We reflect on these challenges and their relation to Agile methodologies and user-centered design.

Planning and management

Project planning

Properly planning a project is no doubt key to its success. Several elements of the project that we did not plan came to "haunt" us later:

- Our design was incomplete, and so when it was time to program certain elements of the user interface, we did not have the right design prototypes to hand to the developer, and had to stop to prepare them. No doubt, we should have spent more time specifying use cases and designing prototypes to suit.
- Our technology vetting process was lacking, and so we found ourselves switching through several stacks, until we assembled a stack that was both feasible and usable for our use case. A thorough evaluation and specification of our needs, and a market analysis of available technologies, would have prevented this. In particular, we should have chosen a software stack that is familiar to several team members, to avoid having

to rely on a single developer for all complex work (a "bus factor" of one), as well as to lower the costs of learning new technologies.[5]
- We did not assign roles until late into the project, presumably due to members' apprehension from taking leadership roles. While an understandable human tendency, it can - and has - resulted in poor planning and follow-up, and significant deviations from the module schedule. A team must have some member willing to *at least* organize and coordinate activity, if not plan, assign tasks, and follow-up on their completion.

Project management

Project planning is intimately tied to project management, in the sense that the latter is a continuous extension of the former: every input from users, technical difficulty, or team member unavailability required re-evaluation and re-planning - adjusting scope, re-prioritizing goals, or shifting resources - flexibility that was highly valuable in bringing the project to conclusion.

And of course - planning and management themselves requires planning, since they have a very real overhead in terms of time and resources, and these need to be scheduled and allocated - but also limited, so as to not spend more time planning than actually producing software…

Risk management

On several occasions we found ourselves facing a risk that was both likely and significant, and required a contingency to be put in place:
- Having experienced difficulties with some complex technology stacks, we conducted two technology surveys to find easier, but similarly capable alternatives - and it is from one of those that we eventually developed the current program.
- Having had problems with developer availability, we prepared for another team member - then engaged with other tasks - to take over their responsibilities, and for *another* team member to take over *their*s.

While imperfect, these contingencies proved crucial in overcoming the technological challenge and moving ahead with development.

Relation with Agile methodology

We adopted several ideas from the Scrum method (Schwaber and Sutherland, 2020): a "product owner" role, flexible role assignments, 1-2 week-long "sprints", and routine meetings of different kinds (eg. planning meeting once a week, follow-up several times a week). These were mixed with tools from other methodologies, such as Kanban boards and Gantt charts. We did not require a "Scrum master", nor "ceremonies" per se.

With regards to planning, it seems that - on the surface - agile methodologies eschew planning in favor of rapid progress (see for example Beck *et al.*, 2001) - but this need not be the case.

---

[5] This is especially true late in the project, where "onboarding" new developers in addition or in place of existing ones can actually be less cost-effective than continuing as-is (Brooks, 1995).

The first step in the Scrum process is for the product owner to “[order] the work for a complex problem” (Schwaber and Sutherland, 2020), which suggests that a complex problem has *already* been defined and decomposed into small “chunks” of work, with the goal of the Scrum process to then *organize* that work to deliver “value”.

The human factor

Managing expectations and communicating intent

Communication between shareholders proved to be one of the most important aspects of our work process:

- Communicating expectations from course staff to the team
- Communicating impressions from users to the product owner and designers
- Communicating design intent from the product owner and designers to the developer
- Communicating usage intent from the team to our users
- Communicating work expectations within the team

We used a variety of tools to facilitate communication, including Slack, Zoom, Jira, Figma, Photoshop and draw.io, as well as notepads and a digital whiteboard. We learned two lessons in this context:

1. Everything has to be written: tasks, use cases, obligations, design specifications, flow charts, concept maps, resources, etc. etc. If it is not written, it might as well not exist.
2. Any item that is meant to be communicated needs to be clear, specific, verifiable, and complete (cf. SMART specification criteria by Doran and others, 1981).

Given how these problems and solutions presented in our small team, it’s eminently clear why larger development teams place such an emphasis on using shared vocabularies, defining well-formed requirements, maintaining thorough documentation, and following a clear development methodology.

As an aside, Agile methodologies seem to assume and depend on the availability of fast and reliable means of communication - and in some cases, like Extreme Programming, on physical presence (Wells, 2009). In a remote work environment where such means are not available, the risk of miscommunication and lost work is higher, and so is the need to over-specify work so as to resist communication errors.[6]

Maintaining a supportive social environment

Agile methodologies place great emphasis on the importance of developer agency and happiness - emphasis which we believe is right. We strove to respect each other, enjoy our work and enable others to enjoy theirs, and resolve disagreements amicably through persuasion and consensus, rather than imposition.

---

[6] More generally, where the expected cost of change mid-project is higher than the initial cost of planning, planning trumps “agility” (Voytko, 2020).

Assessing skill, suitability and quality of work

Given an *ex nihilo* team like ours, how does one assess individuals' skills, abilities and motivations so as to optimize their work assignments, and by extension their efficiency and satisfaction? Asking may yield inaccurate results, and not everyone possesses a portfolio, so this remains an open question with but one answer: time and familiarity.

Users

We attempted to fully adopt the principles of user-centered design: early focus on users and tasks, empirical measurement, and iterative design (Sharp, Rogers and Preece, 2019). For more information on the tools and methods we employed, see User Survey and Testing later in this document.

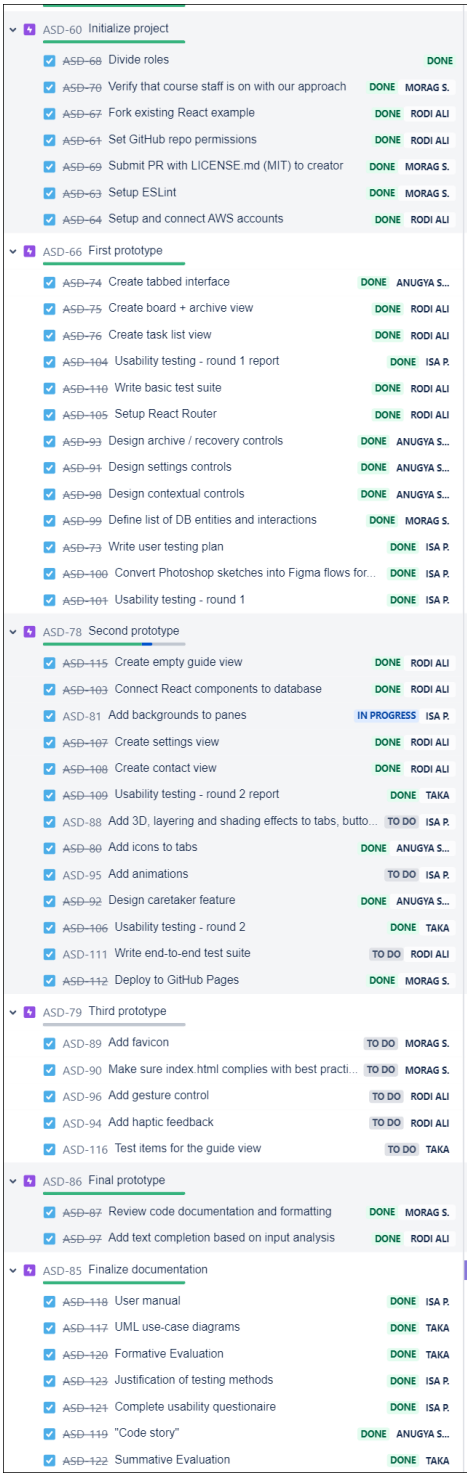
Features	Sprint no.	Start Date	End Date	Span
	1	07/03/2022	07/09/2022	7 days
	2	07/10/2022	07/16/2022	7 days
	3	07/17/2022	07/23/2022	7 days
	4	07/24/2022	07/30/2022	7 days
	5	07/31/2022	08/06/2022	7 days
	6	08/07/2022	08/13/2022	7 days
	7	08/14/2022	08/20/2022	7 days
	8	08/21/2022	08/27/2022	7 days

Figure 0: Summary of our Jira roadmap

Design

Lifecycle of the functional prototype

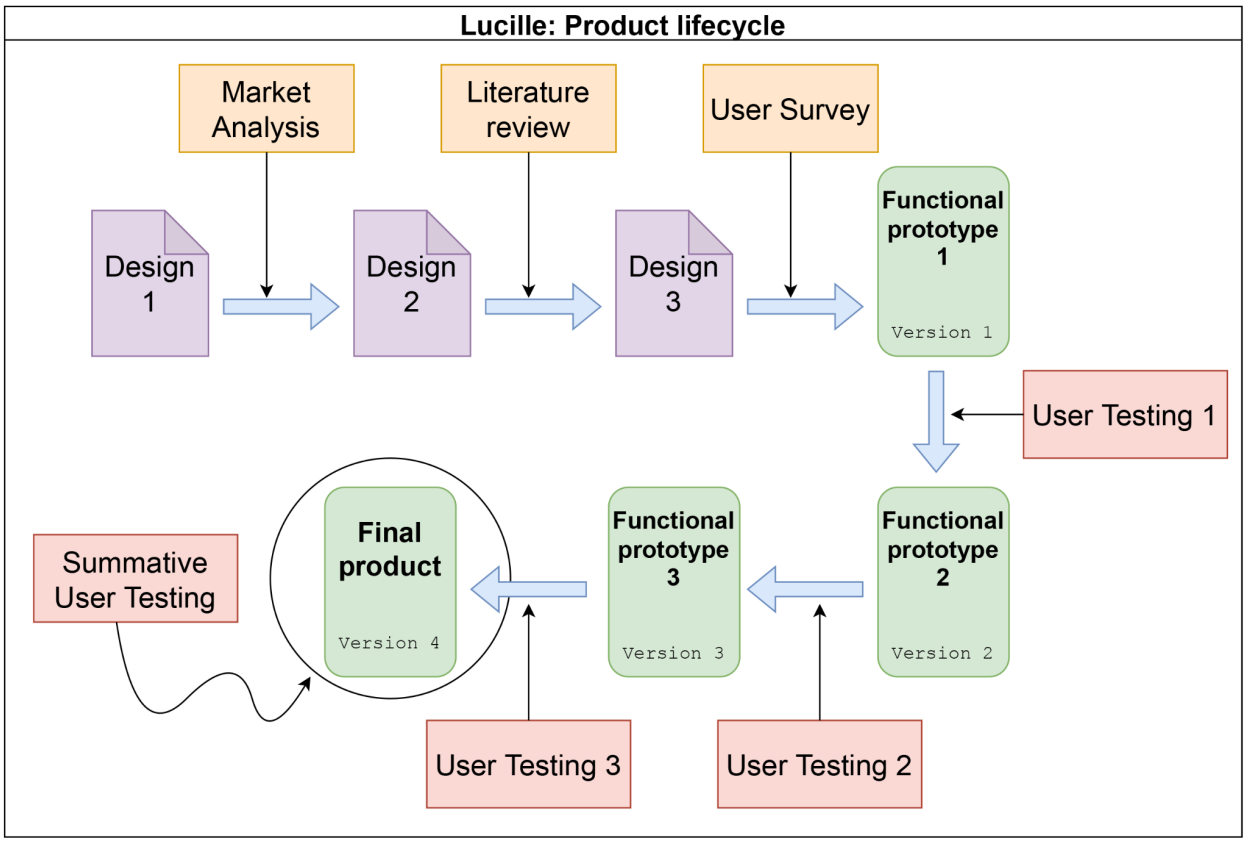


Figure 1: Lucille's lifecycle and evolution

Functional prototype 1

For the first functional prototype of the application, we started with the final version of the "Notabook" prototype (Sharma *et al.*, 2022, sec. Third prototype: Notabook). Individual elements of the prototype were designed following literature recommendations on these subjects (Sharma *et al.*, 2022, chap. Literature review):

1. Age-related changes to perception, cognition and motor function
2. Elderlies' personal and societal attitudes to technology and technology use
3. User experience design recommendation for elderly users
4. Models of technology acceptance among the general population and older users

These were accompanied by a market survey of Apps designed for older users, and "to-do" apps (Sharma *et al.*, 2022, chap. Market analysis).

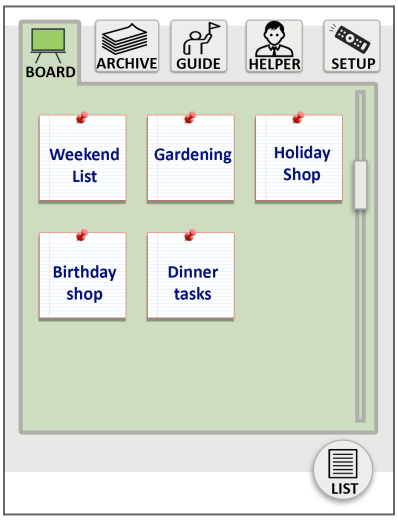
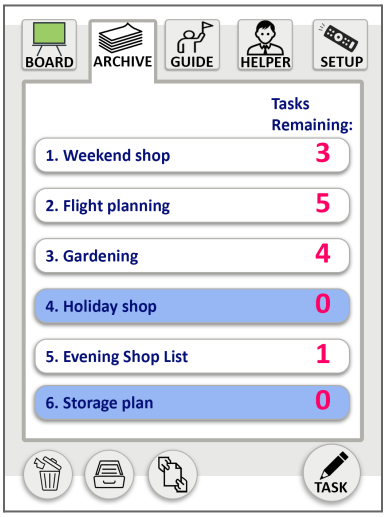
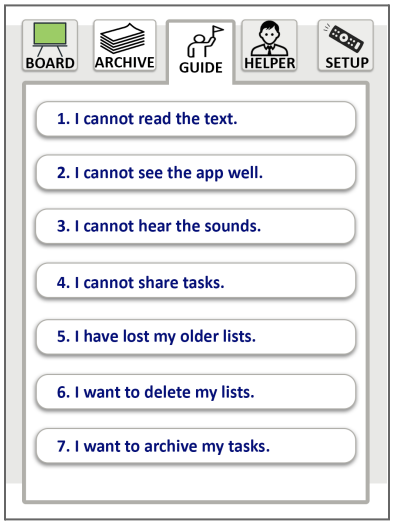
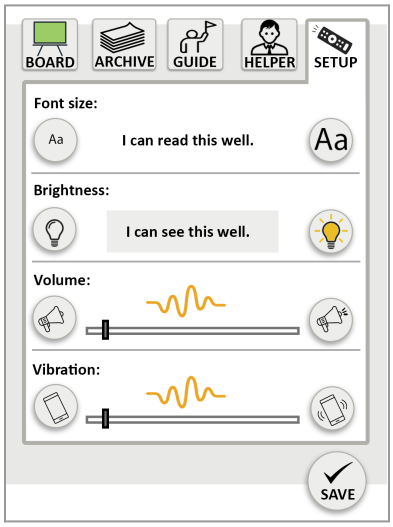
Functional prototype 1:		
Board Page	List View	Archive Page
		
Guide Page	Setup Page	Helper page
		

Table1: Different pages of the Functional Prototype 1.

Color, texture and size: designing interface for the elderly

Seniors, as they grow old, experience a loss of visual acuity, visual focus on nearby objects(presbyopia), compromised peripheral vision, lower light sensitivity, and lower sensitivity to color contrasts(Nunes, Silva and Abrantes, 2010; Pinheiro and da Silva, 2012) . Keeping this in mind, we have included the following things in our design:

- Maximum contrast: Dark text on white backgrounds.
- Large fonts
- Large icons

- Light colored paper with dark blue ink.
- Sound feedback to compensate for visual loss.(Clicking sounds when you click button)
- Haptic feedback to compensate for visual loss. (Vibration when you click buttons)

Minimum memory load on the users:

Older adults go through deteriorating changes in short-term memory, retrieval of semantic memories (general knowledge), prospective memory (schedule of future actions) and they take longer to form new procedural memories(implicit knowledge of how to perform certain actions)(Nunes, Silva and Abrantes, 2010). Hence, it becomes of paramount importance to develop a minimum-memory-load application. These are some of our attempts to reduce the memory and cognitive load of the user when using our app:

- Eye catching buttons with not only an instructive icon but also labeled with large upper scale letters.
- Use of color to signify the main landing page. (The only colorful button is the board-button)
- Editable texts in blue signifying “Writing ink” and different from the inbuilt texts that are black.
- Differentiating “action buttons” from “Navigation buttons”. Action buttons are circular and navigation buttons are rectangular. “Action buttons” on the footer bar or in pop-up and “Navigation buttons” always on the top bar.

Navigability:

- Single branch navigation
- Consistent conspicuous location of the navigation bar
- Layering: When a modal has to pop up, the background is faded so that the users always know where they are. Additionally, they can press anywhere on the faded background to go back. (As seen in Figure 2 below)

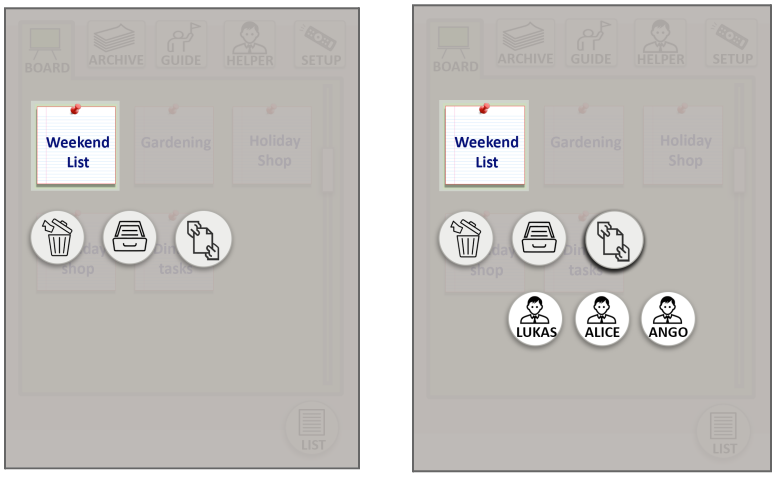


Figure 2: Pop-ups, layering and faded background.

Interaction design for the elderly:

Older adults experience increased motor response times, difficulties with hand coordination, and with maintaining continuous movement. Prevalence of arthritis can further reduce movement speed and accuracy. This has psychological impacts on our users such as the “fear of misclicking”. Interactions such as double clicking become problematic (Nunes, Silva and Abrantes, 2010). In regards to this, our design is very mindful of using senior-friendly interactions and avoiding problematic interactions such as double clicking, and sliding gestures.

- “Single click” Vs “long-press click”: The app’s basic functionalities are all covered without ever using any interaction but the “single click”. However “long-press click” on some elements gives up additional functionalities like sharing, archiving and deleting tasks/lists directly. We use “Long- press click” in place of double click and the amount of holding-click-time comes from our user testing. Similarly, these 2 user-interactions have passed through user testing as well.

- Sliders with controlling buttons on the sides: Our user testing and literature review has shown that a sliding-gesture is not appropriate when designing for elderly users. Hence, we use controlling buttons on the side of the sliders as shown in Figure 3 below.

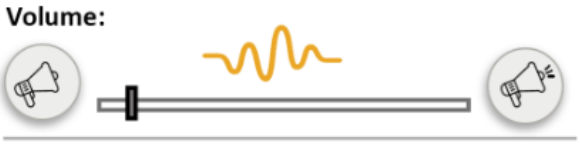


Figure 3: Sliders accompanied with controlling buttons on the side.

Designing for acceptance and adoptability:

Replicating the natural act of “Note-taking”:

We try to replicate familiar actions such as “note taking”,and “pinning lists on board”, so that users can rely on their intuition, recognition and existing knowledge to navigate a software. Users do not have to learn a new “software vocabulary” to be able to use our app. This makes the possibility of adoption and acceptance of the software by our users very high. (Leonardi *et al.*, 2008; Neves and Amaro, 2012; Williams *et al.*, 2013)

- Notebook-parchment texture in the “list” page.
- Pen icons.
- Strike animation
- Green board
- Blue ink
- Pinned paper images
- Write by clicking on the paper , rather than pressing a button.

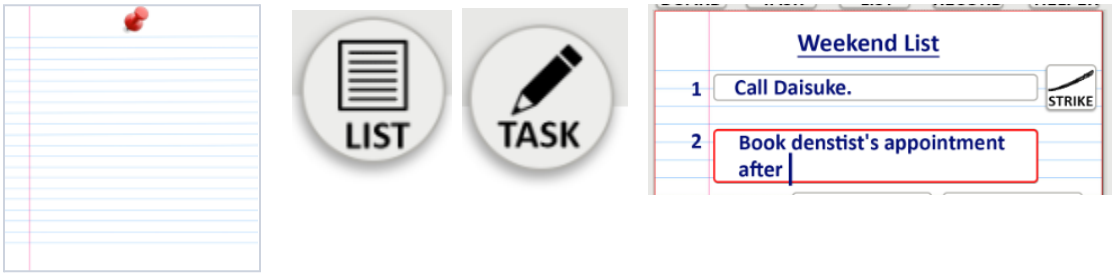


Figure 4: Elements in the interface design that contribute towards making the note taking experience as close to the real world as possible.

Animation:

To make it more acceptable to our target base, we make sure our animations are slow and easy to perceive and that they are neither much longer nor much shorter than one second lit review about animation (Card, Robertson and Mackinlay, 1991).

- “Animated feedback resembles that which is provided by a physical interaction.
- Gives a sense of accomplishment when completing tasks, which improves satisfaction and acceptance” (Sharma *et al.*, 2022, sec. Literature review and design recommendations).

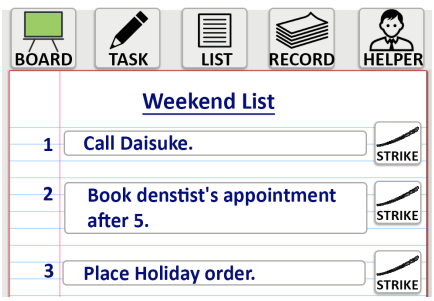
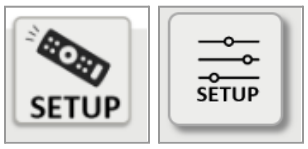


FIgure 5: Strike text animation

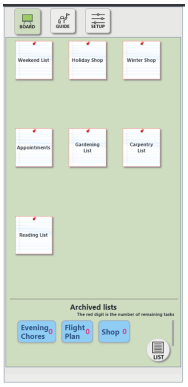
Functional prototype 2:

The first iteration of our functional prototype was tested on a panel of users aged (>65) and of characters most similar to our personas detailed in (Appendix A: Persona Design). Further details about this testing are included under the subheading “Testing” in this report. The following changes were made after integrating feedback from the initial testing:

<table>
<tr><td colspan="2">Integration of user testing feedback in Functional prototype 2:</td></tr>
<tr><td>Change 1: Setup icon changed to a more intuitive symbol.

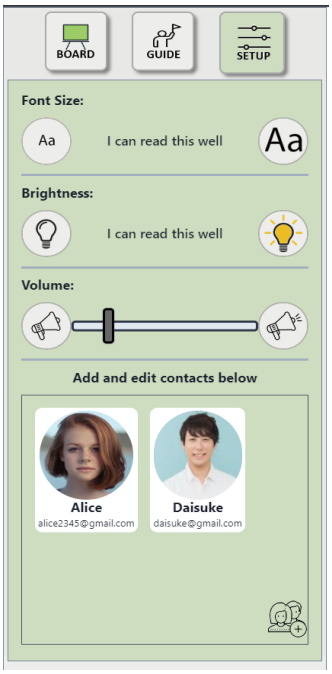
Justification:
Users couldn't relate the former icon to mean “setup”. So we carried out another survey to select an icon for the setup page. Hence, the icon was changed to this user selected choice.</td><td> 

Setup-button before and after changing icons.</td></tr>
<tr><td>Change 2: Integration of “Archive page” into the “Board” page.

Justification:
Users were left confused by the “Archive page”. They also could not utilize its functionality as it was on its own separate page. Hence, we integrated the archive page at the button of the board page, so that when users archived their pinned lists, they would notice it at the bottom.</td><td> 
</td></tr>
</table>

Change 3: Integration of “Helper page” into the “Setup” page.

Justification:
We observed in our testing that users seemed more likely to add contacts when it was part of the setup-page itself. Hence, we integrated the add-contact functionality into the setup-page itself.



Change 4: Terminology change from “**Helper**” to “**Contacts**”

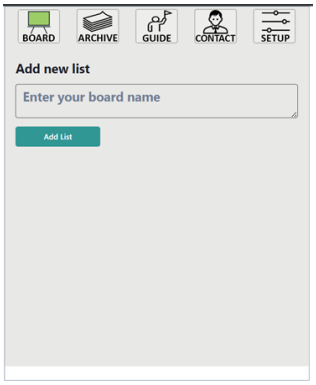
Justification:
Our users couldn't relate “Helpers” to mean their relatives or caretakers, which is what we intended it to mean. So, we changed the terminology to “Contacts”, after which we found no confusion among our users about this.

Change 5: Blue ink changed to black

Justification:
Initially we wanted to differentiate blue text from black to mean that blue was editable by the users and that black wasn’t. But, we found in our testing that this differentiation wasn’t perceived by our users and that there was no value in altering black texts to blue. So we chose to be uniform and set all texts to black.

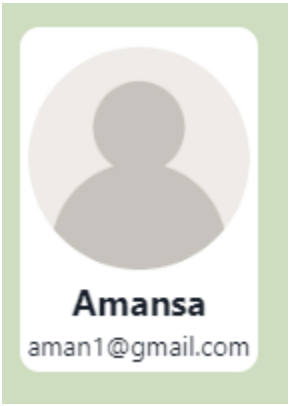
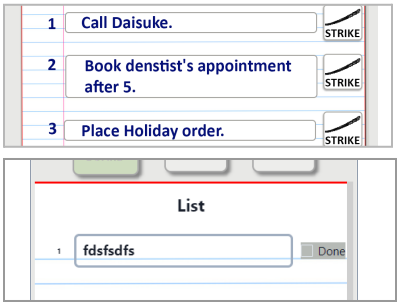
Change 6: Modal removed: In-place writing and listing.

Justification:
We noticed in our testing that the pop-up-modal to insert texts for tasks and list-titles was decreasing the usability of our application. So we changed the design in favor of In-place typing.



Functional prototype 3:

The following changes were made after integrating feedback from another round of testing.

Integration of user testing feedback in Functional prototype 3:	
Change 1: Users are no longer required to add profile pictures when adding contacts. **Justification:** Our app wouldn't let users add contacts without pictures and we found from the testing that users were discouraged from using this functionality as they always wouldn't have a picture on hand. As seen on the picture to the right, when users are unable to find a profile picture of their contacts, the app itself provides a placeholder.	
Change 2: Borders removed to maximize writing areas **Justification:** We wanted to make maximum use of the display area.	

List of bugs fixed leading up to the final product:

Component	Item	Type	Priority
Archive	Combine archive with Board (display at bottom, grayed-out)	Structure	1
Board	Background should be uniform	Style	1
Board	When adding a list, an "untitled list" appears on the board, instead of a "add new list" modal	Functional	1
Board	"Add new board" should be "add new list"	Style	1
Board	Share button functionality	Functional	1
Contact	Move contacts to settings	Structure	1
Contact	Change button from "list" to "add contact"	Style	1
General	Need to keep with terminology: list (not board), contact (not user), etc.	Code style	1
General	Entities need to implement complete data model and actions (see Data model and actions)	Functional	1
General	Code needs to be thoroughly documented and commented	Code style	1
General	White background instead of gray	Style	1
General	Go back, when you click anywhere on the faded background instead of clicking cross	Functional	1
List	Editable title in-place	Functional	1
List	Tasks written and edited in-place, instead of in modal	Functional	1
List	Footer buttons for the whole list (like in prototype file)	Functional	1
Navigation	Tabs should connect with panes, and panes should be bordered	Style	1
Settings	Auto-save on change, without a "save" button	Functional	1
Tasks	Auto-save on change, without a "save" button	Functional	1
Tasks	Strike gesture	Functional	1
Board	The title of a list should be editable in-place	Functional	2
Board	Click on empty space to add list	Functional	2
Buttons	Buttons should have consistent shadowing	Style	2
Buttons	Buttons should have consistent capitalization	Style	2
Contact	Limit file types that can be used as portraits to image files	Functional	2
Contact	Click on empty space to add contact	Functional	2
General	Need to clean up assets/images/	Other	2
General	Need to clean up .eslintrc.json	Code style	2
General	Define minimum width and redesign responsive behavior	Style	2
List	Click on empty space to add item	Functional	2
Settings	Add "vibration" slider	Functional	2
Board	"Mark complete" button for lists	Functional	3
Contact	If no image was chosen for a contact, displays a line drawing of a head instead	Style	3
General	Do we need a Node ESLint environment?	Code style	3
General	Switch from LocalStorage to LocalForage (size limitations) / another DB library	Functional	3
Buttons	Add click/mousedown styling	Style	3
Buttons	Add auditory/haptic feedback (see Vibration API)	Functional	3
General	Clickable elements (eg. lists on boards) should have consistent shadowing	Style	3
Navigation	Highlight tab based on the page you are on	Style	3

Figure 6: List of bugs and errors fixed before the final product.

Evolution of the “Guide page”:

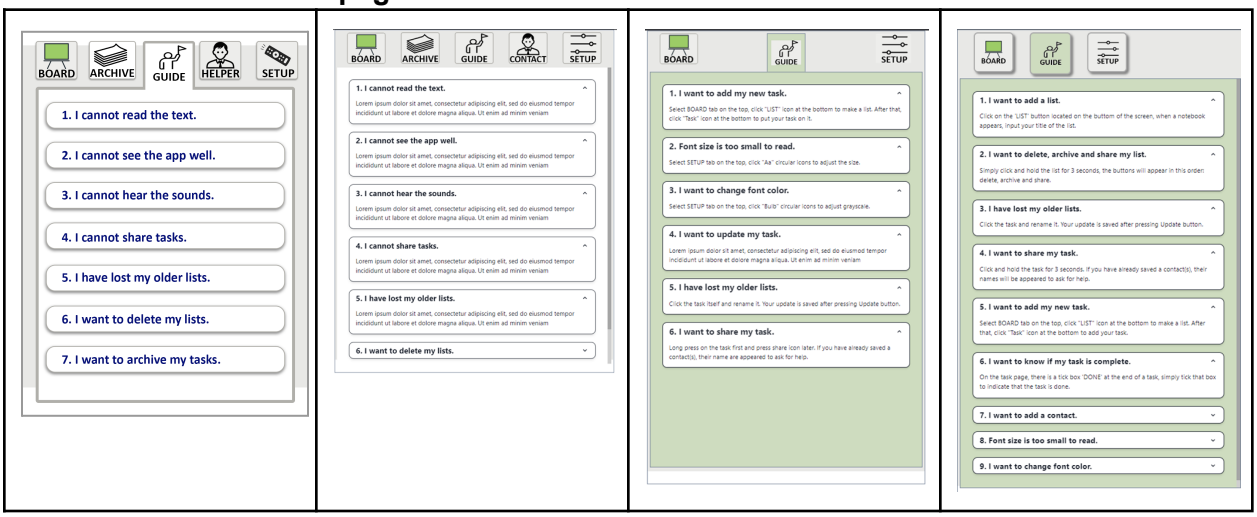
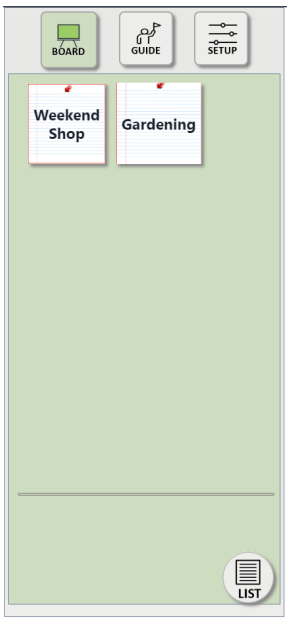
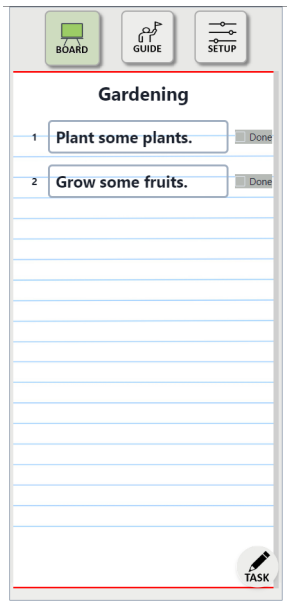
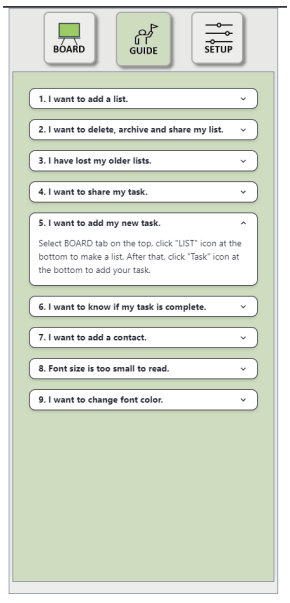
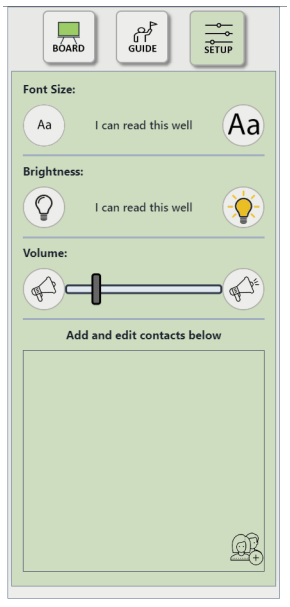


Figure 7: Evolution of the “Guide” page.

The Guide page was designed for users to look up problems they were having with the app. We wanted to study what the question on this page would be. We wanted these questions to reflect the most common problems that the users were having, and the most common type of help they needed to use this app. After multiple rounds of testing, we finalized the top 9 questions to put on this page.

Final product / Functional Prototype 4:

Board Page	Inside a list	Guide Page	Setup page
			

Final round of summative User testing was conducted. The following are some short conclusions from the testing (Further details under the heading “Testing”):

- Users were able to use all the functionalities with ease.
- Users were likely to recommend this app to their contacts.
- Users were likely to use this application in their daily life.
- All parts of the system worked as intended.

Technical design and System Development

Technical output

- A **web based application** compatible with most common **browsers** [Chrome, Safari, Edge, Firefox, Samsung Internet, Opera] (*Most Popular Web Browsers in 2022 [Jun '22 Update]* | *Oberlo*, 2022) with responsive displays to **devices of most common dimensions** [Smartphone, Laptop or desktop computer, Tablet Device, Smart Watch] (*Most Popular Electronics Worldwide [July 2022 Update]*, 2022)

- An application that uses three interactions - **Clicking**, **Long-press**, and **Typing**.

- An application that stores **data** related to 4 things: User's
    - contacts
    - lists
    - tasks
    - setting preferences

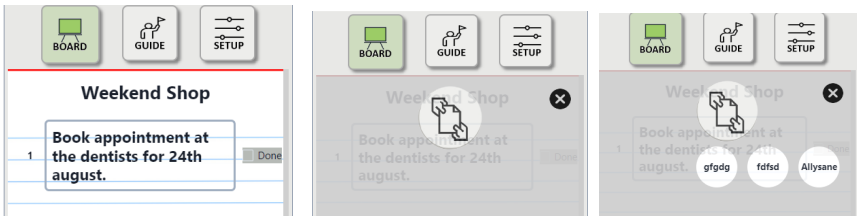
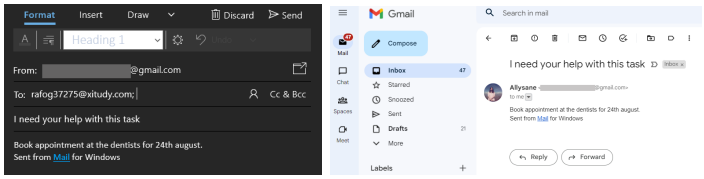
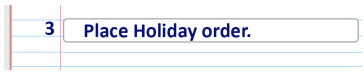
- An application with **3 navigable pages.**

- An application with **interactive** and **dynamic interface components**.

- A system that broadly allows users to:
    - Create interactive to-do lists.
    - Share these lists or tasks with their contacts.
    - Customize their interface [Brightness, volume, font sizes].

- **Security** and **privacy** of user data.

Some features we are proud of:

<table>
<tr><td colspan="2">1. Sharing tasks with contacts to ask for help</td></tr>
<tr><td>Long press on the task you want to share. Share-icon pops up and you click it, your contacts pop-up, you choose “Allysane”.</td><td> 
</td></tr>
<tr><td>Allysane receives a mail saying you require help with that task.</td><td> 
</td></tr>
<tr><td colspan="2">2. Security and privacy of user data: Data stays on the client side.</td></tr>
<tr><td colspan="2">3. Responsive to all dimensions of screen sizes and devices.</td></tr>
<tr><td colspan="2">4. Click anywhere in the faded background to go back.</td></tr>
<tr><td>5. Feedback Animation</td><td> 
</td></tr>
</table>

System description

As stated in our ‘technical output’, our product is a web-based, single page application(SPA), that dynamically interacts with the users rewriting the webpage from the web server, based on clicks and interaction from the users.
Changes in our approach to the system:

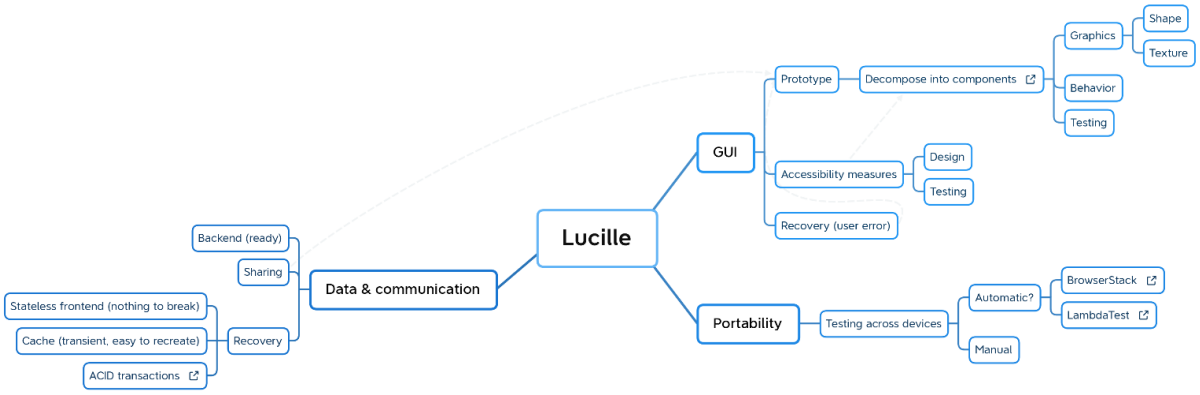


Figure 8 : Work breakdown on 07/27/2022

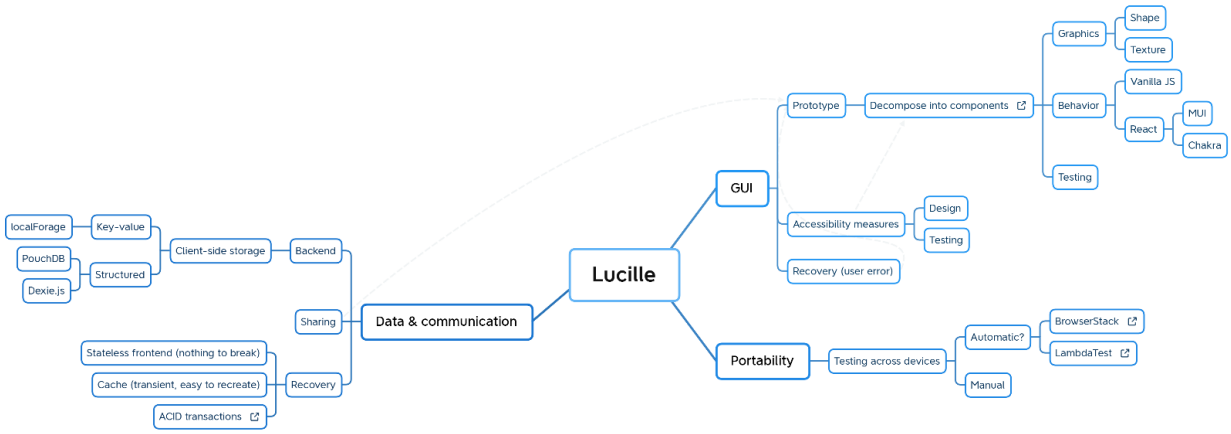


Figure 9 : Work breakdown on 08/10/2022

Features:

Features in broad terms and their history:

The following functionalities were selected after our preliminary study and survey.

1. Add list
2. Delete list
3. Add tasks to the list
4. Delete and edit tasks

The following 2 were added to incorporate the fact that Seniors take help from their contacts to manage their technology needs. Furthermore, we wanted to increase human connectivity.

5. Add contacts
6. Share list/task with contacts

The following was added after our survey showed users needed to change font sizes, screen brightness and volume levels:

7. Change brightness, font size, volume

Architecture:

Single Page Application (SPA)

After evaluating different options for the tech stack and our decision to select the React and IndexedDB stack, it was apparent that we wanted to develop a Single-Page App which served some of our other goals, such as:

- Faster load time
- No need for server-side operations
- Better user experience
- Less complex development and implementation process

Single-page apps are one of the most used ways to develop applications that perform faster than multi-page apps due to the removal of server-side operations needs. The client (the user's browser) downloads the entire application once and only changes the required components

later when the user requests them. Another benefit of using this design pattern is debugging the application directly in the browser while testing the different functions. This has helped us to have the first MVP fast to evaluate the use cases and the idea as a whole.

UML Diagram

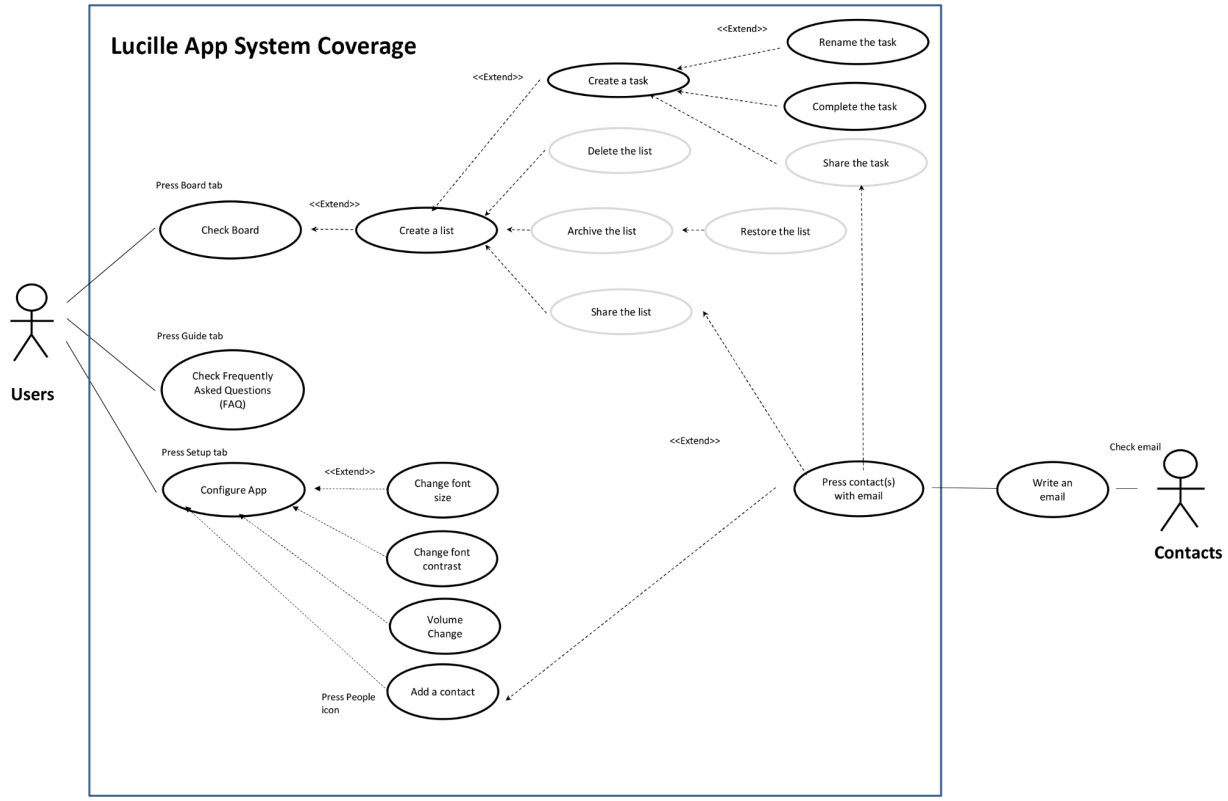


Figure 10: UML diagram

Programming design

OOP and Decorator design pattern and functional programming

We have used object oriented programming (OOP) and to some extent, “Decorator design pattern” to program our application. We chose OOP to facilitate our group’s programming experience and increase the comprehensibility of code and ease debugging through abstraction and modularity. We have also used functional programming throughout the program. Breaking down objects into reusable and smaller modules and use of inheritance and polymorphism has helped us make our program easy to debug and easy to comprehend.

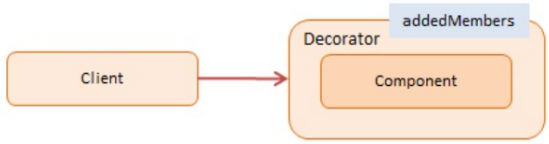


Figure 11: UML diagram Decorator design pattern in javascript (JavaScript Decorator Design Pattern - Dofactory, 2022)

```
// Convert the uploaded images to base64 to use them as the profile picture
const imageToBase64 = (file) => {
  // Create a new instance of the Filereader class to read the uploaded file via the user
  const reader = new FileReader();
  reader.readAsDataURL(file);
  reader.onload = () => {
    onLoad(reader.result);
  };
};
```

Figure 12: Use of decorator design pattern

```
{navData.map((navItem) => // Go thr
  path === navItem.to ? ( // This t
    <NavLink
      to={navItem.to}
      key={navItem.title}
      style={{ margin: "0 10px" }}
    >
```

Figure 13: Functional programming: Use of map function, use of “()=>{}” function

```
const bind = useLongPress(() => {
  boardList.map((board) => {
    if (board.id === selectedId) {
      setButtonPopup(true);
    }
    return board;
  });
});
```

Figure 13.1: Object oriented programming: Bind inheriting from useLongPress.

Why the decorator design pattern?
This design pattern was the most suitable for us as, in our program, we had many objects that belonged to the same class, but would also have their own individually varying characters. Decorator design pattern allows varying behaviors to be added dynamically to these objects without changing the class that the object belongs to(Dung Nguyen, Stephen Wong, and Mark Husband, 2008). We also wanted to use OOP for its abstraction and modularity, as we wanted to increase the comprehensibility of our code and decrease time requirements.

Evaluation of using design patterns:
While React comes with its own patterns, and there wasn't a strict necessity to use a design pattern, using the decorator design pattern has given us some more knowledge about program structures. It has also helped us understand object oriented programming more. Finding the correct design pattern fit for our app was difficult. We debated between “Decorator” and “Mixin”, the former was chosen, but if we had time, we would have liked to explore the latter as well.

Coding process

Our agile process:

We followed an agile methodology, with weekly sprints. Further details and description about our process is present in the “Planning management and execution” section of this report.

CI/CD Pipeline:

Using Continuous integration/ Continuous delivery (CI/CD) in our process significantly helped with building the app simultaneously by 5 developers. Every Time changes were pushed to github, we had an automation system in place that would check for security, stage and deploy our code. Staging of the program made instant accessibility of the product updates without altering the production environment.

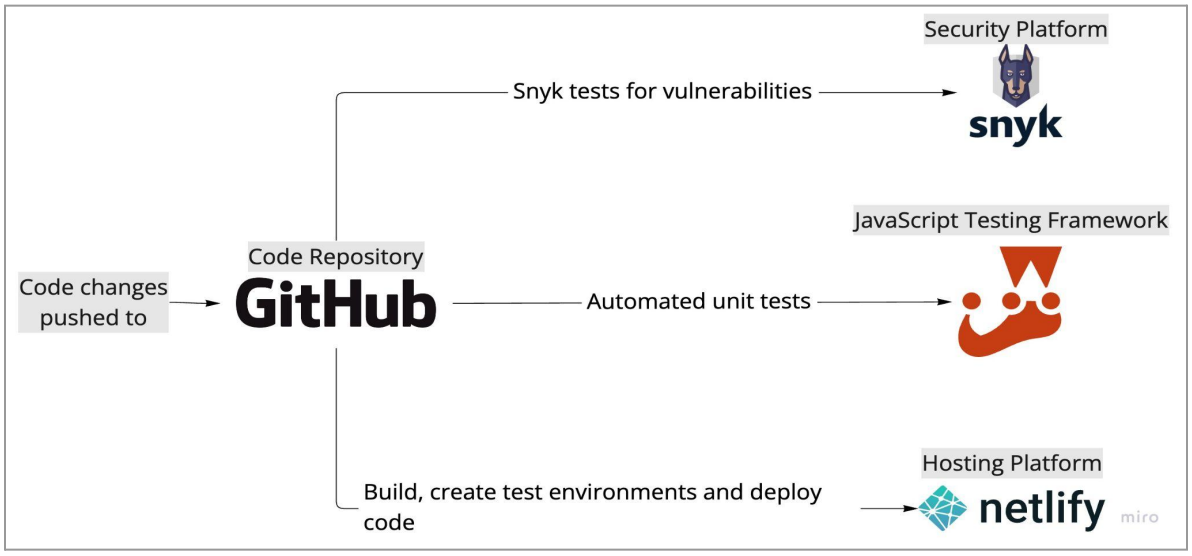


Figure 14: Production, staging and deployment.

Refactoring story:

While we did have to start from scratch multiple times, once we were certain about our stack choices, we made significant code development progress. But, we reached a point, when the code was not easy to comprehend and debugging was impossible. At this point, we started to refactor our code, and systematically divided every component into modules. Once refactored, our speed increased and the program was debuggable and understandable again.

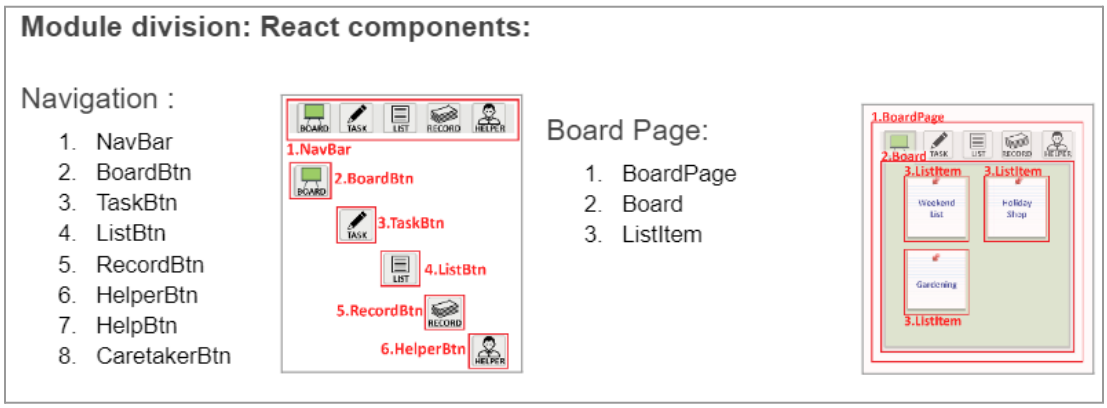


Figure 15: A glimpse into our module division process

Performance testing:

We used “Lighthouse” for general performance testing. One instance of us fixing the app performance was when our assets were slowing down the load time. We discovered this issue after testing and we promptly changed all our assets to web supported .svg formats. and this increased our performance and improved the loading times instantly.

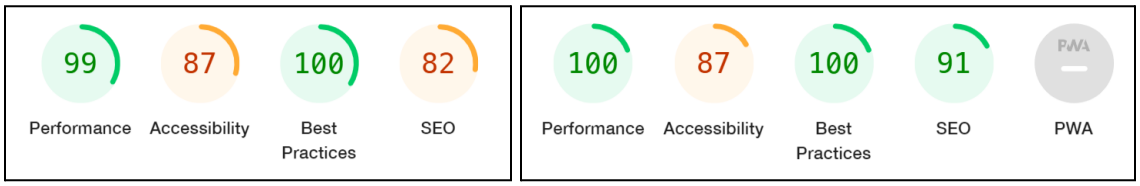


Figure 16: Performance measurement before and after converting assets to .svg format.

Technology stack and dependencies:

4 sets of complete tech stacks were considered in our exploratory study where we tried to answer 4 questions:

1. Establish the time resource that each stack would take to complete the application.
2. The level of ease
3. Accessibility to each member
4. Ability to produce desired functionality OR ability to meet our goals/objectives

In this study, we researched documentation and tutorials of each element of the stack and then went on to create one single page of our application with it. We recorded the time taken to build the first page with each of these stacks. We also noted down if we were able to produce all the functionalities of this page. We also noted down the level of ease for each applicant.

Stack **1**	
Language	• **TypeScript** 
Front-end	• **React**  • **Redux**  • **Material UI** 
Back-end	• **AWS Amplify** 
Advantages: • A complete software solution. • Availability of tutorial and starter templates.	
Disadvantages: • Steep learning curve. • Unfamiliar technology and new programming language to most members of the group. • AWS was only partially free. • Not enough time to grasp all the new technologies.	
Time taken to build the “Board” page: **10 hour**	
Level of difficulty: **High**	
Figured out a clear pathway to build what percent of the aimed features? **40%**	
Additional resources/dependencies required: • aws-amplify/cache 4.0.48 • aws-amplify/ui-react 1.0.1 • material-ui/core 4.11.3 • material-ui/icons 4.11.2 • reduxjs/toolkit 1.5.0 • testing-library/jest-dom 5.11.4 • testing-library/user-event 12.1.10 • types/jest 26.0.15 • types/node 12.0.0 • types/react 17.0.0 • types/react-dom 17.0.0 • types/react-redux 7.1.16 • types/styled-components 5.1.7 • typescript-eslint/eslint-plugin 4.15.1 • typescript-eslint/parser 4.15.1 • aws-amplify 3.3.19 • aws-amplify-react	

Stack **2**	
Language & Front-end	• **HTML, JS, CSS** 
Back-end	• **Node**  • **Mysql**  • **Express** 
Advantages: • The whole team is proficient in this stack. • Experiences from other modules (DNW).	
Disadvantages: • Time consuming. • Required repetitive and manual work. • Difficult to make our application dynamic and interactive with this stack.	
Time taken to build the “Board” page: **15 hour**	
Level of difficulty: **Low**	
Figured out a clear pathway to build what percent of the aimed features? **100%**	
Additional resources/dependencies required: • body-parser 1.20.0 • ejs 3.1.8 • express 4.18.1 • jquery 3.6.0 • jsdom 20.0.0 • mysql 2.18.1	

Stack **3**	
Language	• **Javascript** 
Interface	• **React**  • **CSS** 
Router	• **React Router** 
Data	• **LocalStorage** 

Advantages:

- New technologies included were moderately easy to grasp.

Disadvantages:

- LocalStorage couldn't provide a complete data solution.
- CSS was time consuming as a styling solution.

Time taken to build the “Board” page: **15 hour**

Level of difficulty: **Medium**

Figured out a clear pathway to build what percent of the aimed features? **70%**

Additional resources/dependencies required:

- testing-library/jest-dom 5.16.4
- testing-library/react 13.3.0
- testing-library/user-event 13.5.0
- dexie 3.2.2
- dexie-react-hooks 1.1.1
- expo 46.0.10
- gh-pages 4.0.0
- react 18.2.0
- react-dom 18.2.0
- react-router-dom 6.3.0
- react-scripts 5.0.1
- use-long-press 2.0.2
- uuid 8.3.2
- web-vitals 2.1.4

Stack **4**	
Language	• **Javascript** 
Interface	• **React**  • **Chakra UI** 
Router	• **React Router** 
Data	• **Dexie.js** 

Advantages:

- Reduced manual and repetitive tasks.
- Easy to learn.
- Plentiful online support and resources.
- Efficient documentation.
- Data design was fast and efficient.
- A complete software solution for our scope.

Disadvantages:

- Some new technologies to grasp.

Time taken to build the “Board” page: **5 hour**

Level of difficulty: **Medium**

Figured out a clear pathway to build what percent of the aimed features? **100%**

Additional resources/dependencies required:

- chakra-ui/react 2.2.4
- emotion/react 11.9.3
- emotion/styled 11.9.3
- testing-library/jest-dom 5.16.4
- testing-library/react 13.3.0
- testing-library/user-event 13.5.0
- dexie 3.2.2
- dexie-react-hooks 1.1.1
- expo 46.0.10
- framer-motion 6.5.1
- gh-pages 4.0.0
- react 18.2.0
- react-dom 18.2.0
- react-router-dom 6.3.0
- react-scripts 5.0.1
- use-long-press 2.0.2
- uuid 8.3.2

Conclusion:
We chose **'Stack 4'** to develop our final product because compared to the 4 stacks we explored, this stack was the most efficient in terms of time-consumption, the new technologies in it were reasonably easy to grasp and it was also a complete solution to our project aims and goals.

Discussion of the technologies and dependencies used for the final product:

1. **Javascript:**

   Using javascript as the main programming language was a straightforward choice as it was the only language in which all 5 group members were proficient in. Furthermore, javascript was a fitting language to attain all our technical aims and objectives.

2. **React:**

   Advantages:
   - React comes with built in states and render-update functionality so it reduces the need of manual work and having to build these functionalities from scratch.
   - Plentiful online support and efficient documentation.
   - Reuse of react components.
   - Our list of necessary libraries and dependencies work with React seamlessly.
   - Easy Testing procedure with React.
   - Top ranking among its competitors.

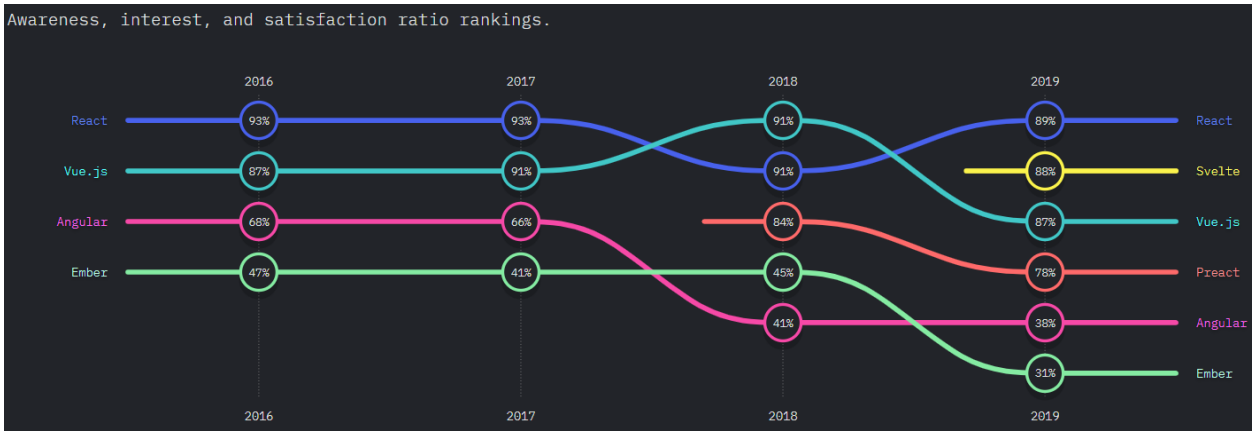


Figure 17: React's ranking compared to its competitors. (*The State of JavaScript 2019: Front End Frameworks*, 2022)

Issues faced:

- It was a new framework that most of the group had never worked with before, so we had to spend a lot of time researching and getting started. In retrospect, our group should have included a much longer sprint dedicated just to learning React.

3. **Chakra UI:**

Advantages:

- It reduced our need for manual styling.
- Other than the styling, Chakra UI also comes with tab, navigation and button functionalities which we were able to make use of.
- Chakra UI elements are accessible and there are some prebuilt security measures in place.
- We were also able to borrow some responsive nature of Chakra elements into our program.

Alternatives explored:

After exploring the most common css frameworks, our group was able to pinpoint “Material UI” and “Chakra UI” as the most beginner-friendly framework. We initially started our work with “Material UI” but the components that it provided were just not suitable for our interface and so we quickly changed to “Chakra UI”.

4. Node Package Manager (npm):

Advantages:

- With npm, we were able to systematically install, manage and update all our dependencies in a much shorter time.
- From a security perspective, npm ensured our dependencies were being installed from trusted sources.

Security

Snyk:

Synk is a cybersecurity tool that we are using in our project to analyze and fix vulnerabilities and bugs in our code and dependencies(*Snyk | Developer security | Develop fast. Stay secure.*, 2020). We chose Snyk because of the following reasons:

- Easy configuration
- Free of charge
- Top position in the market and good ratings among its competitors

Pre-detection of errors

ESLint

ESLint is an error detection and correction tool that analyzes your javascript code and detects errors or problematic patterns. (*Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter*, 2022). We are using ESlint for the following reasons:

- ESLint is customisable in the errors it detects.
- ESLint helped us maintain a uniform coding style in a group setting.
- We were able to incorporate coding style guides.
- The configuration of ESLint was simple enough after some initial preparation and research.
- We supplemented ESLint with “Prettier” which is a code formatter but doesn't handle errors.

Data layer

We created a data model for our application. The following tables contain all the data entities and the interaction of our app with the database layer.

Data model:

Data entities and properties:	
List 1. Title (string) 2. Tasks (list of tasks) 3. State (enum; active/archived/deleted) 4. Shared with (list of helpers)	**Contact** 1. Name (text) 2. Email (email address) 3. Shared active tasks (list of tasks) 4. Shared active lists (list of lists)
Board 1. Active lists (list of lists)	**Settings** 1. Font size (number) 2. Brightness (number) 3. Volume (number) 4. Vibration power (number) 5. UUID (string)
Task 1. Content (text) 2. State (enum; active/done) 3. Shared with (list of helpers)	

Data actions:	
From lists 1. Re/name list 2. Add task 3. Edit task 4. Change task status (complete/incomplete) 5. Share task 6. Delete list 7. Archive list 8. Share list	**From the board** 1. Create list 2. Delete list 3. Archive list 4. Share list
From settings 1. Adjust parameter	**From helper view** 1. Add helper 2. Edit helper 3. Delete helper

After exploring multiple data solutions such as Amplify, mySql, LocalStorage, and indexedDb(dexie.js, pouchDB, LocalForage), we settled on Dexie.js which is a wrapper for

indexedDB. We chose this solution for its easy interface, beginner-friendly usability, and it was most suited to attain our data goals.

Dexie.js

Dexie.js

Advantages:

- Security:
  One of our biggest reasons for choosing Dexie.js was that it is an indexedDb solution which means client side data remains on the client side and nobody but themselves have access to it. Also, in indexedDB, cryptography is done outside the browser execution environment, so it's additionally safer.
- Easy interface: Querying the database and updating it was simple.

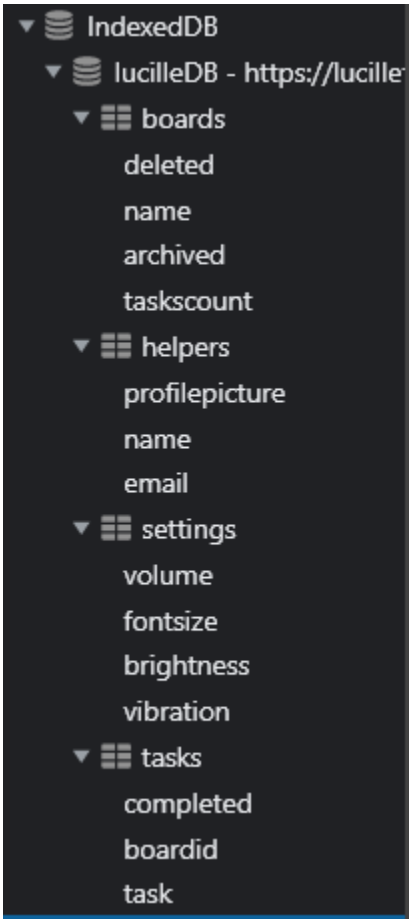
Issues and challenges:

- Asynchronous functions: We had to do extensive research and homework into javascript async functions, callbacks and promises before we could work with async functions in dexie.
- Migration from localStorage to indexedDb: Since dexie.js was not our first data solution, we were left with the issue of migrating all our data from localStoragestackoverflow, documentation for dexie.js

```
// Aync function to fetch settings table from the db and use the callback functions to store the data in variables
const allSettings = db.settings.toArray().then((setting) => {
  setMdFont(setting[0].fontsize);
  setSmFont(setting[0].fontsize - 10);
  setFontBright(setting[0].brightness);
});
```

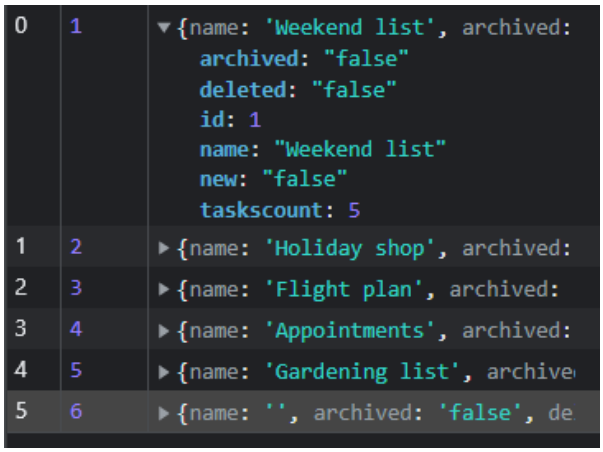
Figure : Use of asynchronous function to query the database.

Data Implementation:

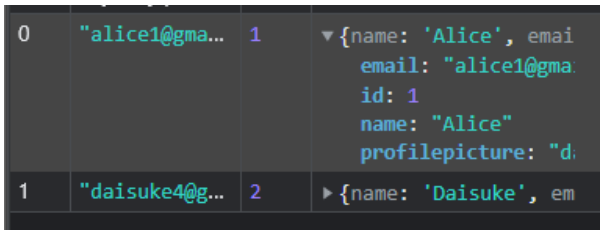


Our data structure in indexedDB-
The data structure it divided into 4 parts:

1. Board: all information related to the board and the lists on it.
2. Helpers: Information about the Contacts/Helpers
3. Settings: Information about app settings such as font brightness.
4. Tasks: The actual task's text content and its state.

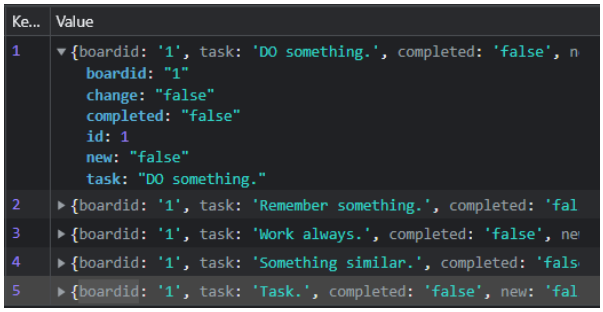


1. **Board:** This is where all the lists are stored. Different properties of the list such as its name, taskcount and its states are also stored here.



2. **Helpers/Contacts:** Contacts have a name, a photo and an email id assigned to them and this information is stored here.

3. **Settings:** This part of data store settings value of the app such as brightness and volume.

	 4. **Task**: This is where individual tasks and any information about them is stored. Each task comes with the board id of the list they belong to and other properties such as complete/incomplete state.

Program structure

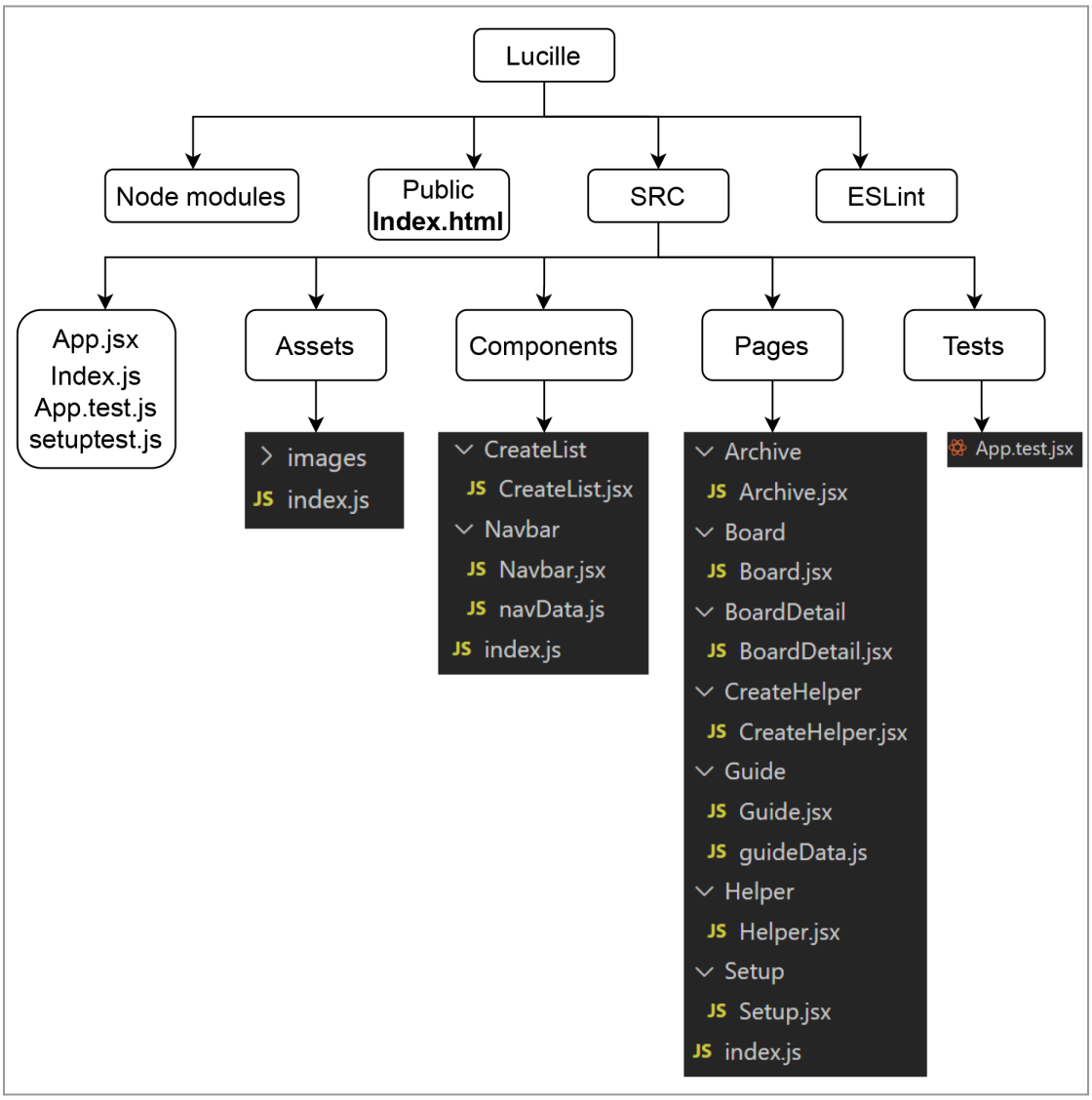


Figure 18 : File structure of Lucille application

Code Story: Building the app

Here, we write about the code that brought about this feature. Any issue or challenge faced when trying to program it. Any solution used and their sources (like stackoverflow). Any interesting bit of code's snippet. Any library or technology used to achieve this feature. Any description that sells the idea that it was technically challenging and sort of new in the way we did it.

Code story: feature development, challenges and evaluation.

1.	**"Share" functionality**

Challenges:

1. Share lists/ tasks with contacts.
2. Send email to contacts with the data about the lists/ tasks being shared.

Solution:

1. **mailto: method**
   We used the "mailto:" method in javascript to send emails to the contacts. This method was the most suitable for our goal, as it was the simplest method available that also allowed sharing of data through email.

```
<Box
  onClick={
    () => window.open(
      `mailto:${  helper.email  }?subject=I need your help with this task&body=
      ${  boardData.filter((task) => task.id === taskId)[0].task}`
    )
  }
```

2. **Uniform Resource Identifier (URI):**
   In our program, we create an URI instance using the email address of the contact. This email address is now an URI object and it is accessed through the Simple Mail Transport Protocol.

3. **Nested in-line functions:**
   We made use of nested in-line functions to pass down the sharing-function along a chain of react components. This process is triggered through event handlers in the interface elements.

Evaluation:
In our research for various ways to send emails through the program, we came across many libraries such as SmtpJS, but because of time constraints and in favor of simplicity, we went with the 'mailto:' method. In our future development, we plan on using the SmptJs library.

2.	“Long press” functionality

We try to differentiate between single click and long press and produce 2 different sets of functionalities for these two interactions.

Challenges:

1. Accessing child functions of a different object, from inside another object.
2. Selecting a library that can be used to detect long-press interaction.
3. Unfamiliarity with using React hooks.

Solutions:

1. **JavaScript function bind():**

```
const bind = useLongPress(() => {
  setButtonPopup(true);
});
```

```
<Image
  w="80px"
  ml="20px"
  {...bind()}
  src={StrikeIcon}
  onMouseEnter={() => setTaskId(task.id)}
  style={{ cursor: "pointer" }}
/>
```

   By using the bind method, we solved our issue of not being able to access the method of a different object from inside some other object. We stumbled upon this solution after extensive research into documentation and tutorials. The E4X double double dot operator “..” along with bind(), connects all the descendents of the component to the outer method, thus enabling us to use the setButtonPopup() method of an external object. This further enabled us to implement the long press functionality on our interface buttons and elements.

2. **‘use-long-press’ library:**
   use-long-press is a js library that is used to detect click events. This library was ideal for our purposes as it supports both mobile and desktop devices and it is very simple to use alongside react. It is also very easy to specify the exact hold period in seconds using the “threshold” property in event handlers.

```
import { useLongPress } from "use-long-press";
```

.

3. **React hook**
   Hooks are a new tool that appeared in React 16.8. We made use of “React Long Press Hook” which supplied us with long-press detecting objects. We were able to use react state and lifecycle without defining new classes or switching between classes. This was a new technology we were not familiar with, and we spent a long time trying it out.

Evaluation:
Our use of “React Long Press Hook” was limited and in future works, we plan on making full use of all the varied methods that it comes with. The error debugging and learning of new techniques took a significant amount of time but as a result the Code we have put in place is modular and flexible and can be easily extended to handle multiple interactions(hover, double-click) not limited to long-press.

3.	Voice to text

Challenges:

1. Identifying the most suitable voice to text conversion library.

Solutions:

1. **Alan AI**

   We explored multiple solutions to this challenge, such as the “react-speech-recognition” React hook, “SpeechRecognition JavaScript interface”, “Rev.ai speech-to-text API”, and “Alan AI”. We chose Alan AI for the abundance of documentation and tutorial resources on it. Alan AI is a voice interface that works seamlessly with React apps.While it may be a complete voice assistant tool, we are using it as a solution for voice to text conversion.

Evaluation:
Alan AI comes with advanced analytics that can be used for user experience and user behavior analysis. In future extensions, We aim to integrate these analytics into our testing regime. We also acquired the skill to integrate AI platforms into React apps while trying to implement this feature.

Final state of the feature: In the final product, we had to remove this feature as we couldn't successfully and completely implement it. We were also concerned about the ethics of storing voice data and ensuring the privacy and storage of this data, which was beyond the scope of our project.

4.	Responsive display

We wanted our application to be responsive to all screen sizes and devices. We were successful at our attempt to do this, but we did run into some issues and challenges.

Challenges:

1. We had to use multiple layers in our program and in an attempt to make the interface responsive , we ran into the issue of bottom layers being hidden and unresponsive..

Solutions:

1. Media queries and Breakpoints

2. Restructuring and arrangement of several layers of react components.

5. Animation

Challenges:

1. Create custom animation.
2. Trigger animation based on the database.

Solutions:

1. **Keyframe Animation**

```
# completedTaskAnimation.css ●
lucille > src > assets > # completedTaskAnimation.css > ...
1   @keyframes completedTask{
2     0%   { width : 0; }
3     100% { width: 100%; }
4   }
5   .completedTask {
6     position: relative;
7   }
8   .completedTask::after {
9     content: ' ';
10    position: absolute;
11    top: 50%;
12    left: 5px;
13    width: 100%;
14    height: 4px;
15    background: red;
16    animation-name: completedTask;
17    animation-duration: 0.1s;
18    animation-timing-function: linear;
19    animation-iteration-count: 1;
20    animation-fill-mode: forwards;
21  }
```

We used Keyframe animation technique along with self-made css components to create extremely flexible and customizable animations.

2. **Inquiring database to change class name:**

```
className={taskCompletedStatus(task.id)
  ? "completedTask"
  : ""}
```

```
const getTasks = useLiveQuery(() =>
  db.tasks.where({ boardid: boardId }).toArray()
);
```

Animations are triggered by changing class names of the animated element. This change is triggered by querying our database by using “useLiveQuery React Hook”.

Evaluation:

We are using animations in our app to give visual feedback to our users. These animations needed to be customisable so we opted to make our own animations from scratch instead of using a library. KeyFrames were a fairly recent technology and creating css component from scratch involved a lot of manual work. If we had more time, we would have explored other options with less manual work.

5. Completed and uncompleted state of task.

Challenge: Give a completed/uncompleted state property to each task and make it changeable.

Issues faced:

1. The completed/ uncompleted state of the task would change in the display layer, but it wouldn't change in the data layer.
2. Changing the state of one task would apply to all the tasks.

Solutions:

1. **Creation of individual ids:**
   Our problem of all the tasks changing their change when one task changed its state was solved by giving each task its own unique individual id.
2. **Boolean state column in database:**
   The state was stored as a boolean data in a column in our database, This database was first queried and then changed whenever a task changed its state.

```
const completeTask = async (id) => {
  const getTaskData = boardData.filter((taskItem) => taskItem.id === id);
  const currentTaskCompletedValue = JSON.parse(getTaskData[0].completed);
  const reverseValue = !currentTaskCompletedValue;
  db.tasks.update(id, { completed: reverseValue.toString() });
  await db.boards.update(parseInt(boardId, 10), {
    taskscount: currentBoardTasksCount.taskscount - 1,
  });
  setButtonPopup(false);
};

const taskCompletedStatus = (id) => {
  const getTaskData = boardData.filter((taskItem) => taskItem.id === id);
  const currentTaskCompletedValue = JSON.parse(getTaskData[0].completed);
  return currentTaskCompletedValue;
};
```

Evaluation:
We had to explore multiple data solutions to implement this feature. We tried Amazon amplify which was a complete solution but the difficulty level was high , and then we tried localStorage which was simple to use but it supported only a very small kilobyte of data. So we ended up using an indexedDb solution with dexie.js library, which was both easy to use and supported the full data plan.

6.	**Layering(Background clicking to go back)**

Challenges:

1. Click anywhere on the faded background, not just the cross-button to go back.

Solutions:

1. Onclick Event listeners.

```
onClick={() => setButtonPopup(!buttonPopup)}
```

7.	**Archiving**

Challenges:

1. Querying and updating the database.

Solutions:

1. **useLiveQuery** :
   We used the useLiveQuery technique that is available through dexie.js library. This doesn't just update the database but also observes any change to the database elements.

```
const boardList = useLiveQuery(
  () => db.boards
  .where({archived: "false", deleted: "false"})
  .toArray()
);
```

Evaluation:
We had to change our database solution multiple times when trying to implement the archiving functionality. If we were to do this process again, we would choose a complete database solution such as “Amplify” and set aside more time to learn it.

8.	**Save and change app settings (Brightness and font size)**

Challenges:

1. Apply display settings throughout the whole app.
2. Save the app setting once changed by the user.

Solutions:

1. **React state**
   We made the use of React state (useState()) to apply values for font-size and app brightness throughout the whole app.

```
<Heading
  className={taskCompletedStatus(task.id)
    ? "completedTask"
    : ""}
  border="4px"
  p="10px"
  w="100%"
  borderColor="gray.400"
  rounded="lg"
  style={{
    fontSize: `${mdFont}px`,
    filter: `contrast(${fontBright}%)`,
  }}
```

```
function App() {
  const [mdFont, setMdFont] = useState(30);
  const [smFont, setSmFont] = useState(20);
  const [fontBright, setFontBright] = useState(100);
```

2. **Change display settings:**
   We are using functions that query the database and change the display settings like “brightness” in the database itself.

```
const increaseBright = () => {
  const newFontBright = fontBright + 20;
  setFontBright(newFontBright);

  db.settings.update(1, { brightness: newFontBright });
};
```

Evaluation:
While trying to implement this functionality we realized that changing the whole app brightness and dealing with the device permission to do so was very complex so we came up with an easy solution of increasing the contrast of the fonts.

Version Control

Code version control: Git, GitHub, Fork
Report version control: Version history in google docs

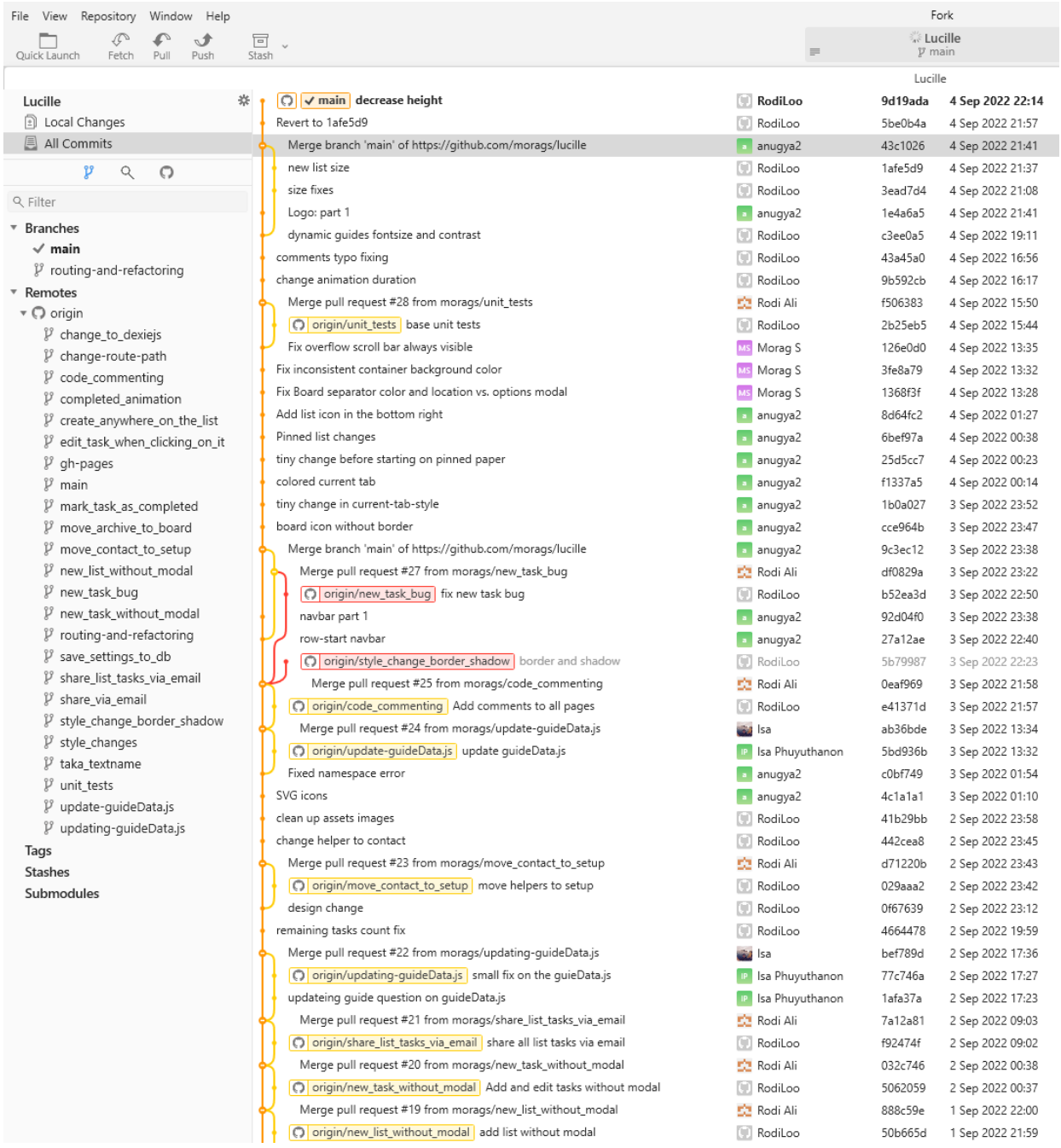


Figure: Changes to our repository and version control logs

Testing code and error handling

Error handling:

Because errors will happen in different stages of the application lifecycle, one of our focus areas when it comes to error handling was to wrap all the promise functions inside a try…catch statement to ensure the stability of our application, although we are not handling all the errors in a meaningful way, we can at least get information and logs to investigate the incidents more smoothly. Most of our try…catch blocks were wrapping the API calls to the IndexedDB client-side storage via the Dexie library, as all these calls are promises and don't always have values or connections.

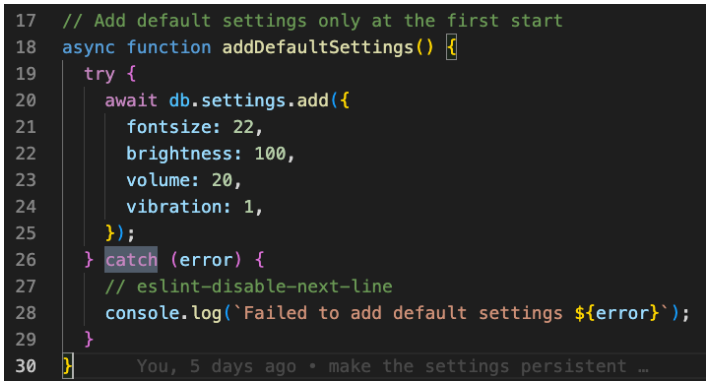


Figure 19: Try-catch code blocks in our program

Figure 20: Try-catch code blocks in our program

Overall testing regime:

As our application is highly dependent on the end-user's browser, on both mobile phone and computers, we had to always perform cross-browser visual testing which helped us to capture some of the bugs and unplanned behaviors on different versions of main browsers. The tool we used to perform this type of testing is www.browserstack.com which gave us instant access to the main operating systems such as Windows (with different versions) and MacOs (with different versions) and the main browsers stack on these operation systems such as Chrome, Firefox, Safari and Edge. One of our checks before deploying the latest changes to production was to test the staging environment on these different browsers and perform the application main functionalities.

We were able to have this level of control on our tests and application stability due to the implementation of continuous integration, continuous delivery practices which was provided by Netlify, our hosting platform.

Unit testing:

We have used both Jest and react-testing-library to unit test our code. We faced a lot of issues while preparing our code to be tested due to the use of the JSX syntax extension and the IndexedDB APIs. The team members responsible for the unit tests had to learn about different elements in order to be able to write the starter unit tests we implemented in our application. The total amount of hours we spent on the configuration struggles was approximately 15 hours. After dealing with all the difficulties, we managed to have 9 passing unit tests for the main components of the application to ensure they are rendering correctly on application startup.

Figure 21: Running tests

```
12  describe('CreateList', () => {
13    it('renders without crashing', () => {
14      const root = ReactDOM.createRoot(document.createElement('div'));
15      root.render(<CreateList />);      You, 7 hours ago • base unit tests …
16    });
17  });
```

Figure 22: Example of a testing code in our program

Deployment

Due to our early adoption of CI/CD (Continuous Integration, Continuous Delivery) practices in our development processes, such as storing the code-base in Github and using it for version control, and integrating our hosting platform Netlify with the repository, this helped us focus on adding new features, fixing bugs, improving the stability and faster design iteration rather than focusing on setting up the infrastructure. This has even reduced the lead time to production due to all the automation that has been configured in between Github and Netlify, the hosting platform to test, build and deploy our application on every new change. A new feature development journey looked like this:

- The developer implemented the feature locally on their machine
- Pushed the branch to Github
- Opened the pull request to merge it into the main branch
- A team member reviewed the changes
- The change was getting merged after approval from reviewers

- A webhook call invokes the Netlify function to fetch the new change
- Netlify build and deploy the change to a new dedicated staging environment
- If the new feature/change/bug fix is approved visually by the developer in the staging environment
- The staging environment is then pushed to production.

These practices helped us in some situations to revert back some undiscovered buggy changes during the tests via either Github or Netlify. This has been very beneficial to reduce the length of the impact on our end-users when having bugs in the production environment.

These processes were completely new to some of the team members, which created some confusion at the early stages of application development. After testing it and changing some of the rules at later stages, such as reducing the number of reviewers to only one and after experiencing its benefit to reduce the stress on team members pushing bugs into production, the whole team started appreciating it.

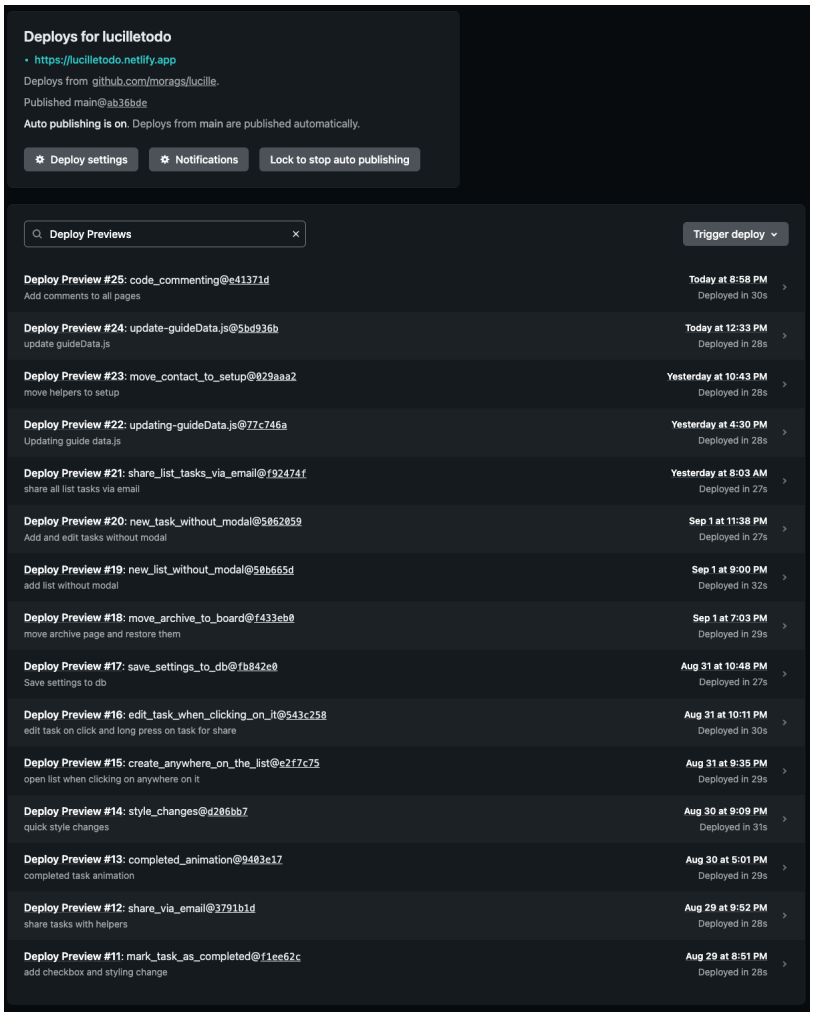


Fig 23 : Deployment History of our application

User Survey and Testing

Usability Test

We decided to use the Usability Test as our method of assessment. To gain as much knowledge from the user as possible, our planning, including selecting the best applicant to be invited to our interview, developing the questions to be asked, and altering our interview flow (see Appendix C), was rigorously tested internally. Companies routinely do usability testing as part of the creation of standardized products that endure multiple generations, including word processing programs, databases, and spreadsheets (Sharp, Rogers and Preece, 2019). A usability specification, which enables developers to test upcoming prototypes or versions of the product against it, frequently compiles the results of usability testing. Current levels are stated, along with optimal performance levels, minimal levels of acceptance, and other basic specifications. Following that, modifications to the design can be made to the navigation structure, terminology, and user interface. Then, these modifications can be observed (Sharp, Rogers and Preece, 2019). Usability testing is well-established in UX design, but it has also begun to acquire more traction in other industries like healthcare, particularly as mobile devices play an increasingly important role in hospitals and for tracking one's own health (for instance, Fitbit and the Polar series, Apple Watch, and so forth) (Sharp, Rogers and Preece, 2019). Usability testing was thus selected as our primary method of evaluation since it is a norm, is trackable, and produces positive results in healthcare products, as we have all seen in the market with devices like Apple Watch and Fitbit that are effectively implemented.

Formative Evaluation

Between our initial prototype design for the Notebook and a second prototype, the formative evaluation took place, and several applicants were selected to validate our concept with signed consent to record the session (see Appendix F). We were able to create plans in a one page dashboard thanks to David Travis' 1-page usability test plan from Userfocus (Travis, 2016). David Travis has set up a single-page testing dashboard because “some start-up users of our toolkit have complained that it's a little too documentation-heavy for their purposes. They understand that planning is necessary for usability tests, but they don't require the level of detail we do in our test plans because they are essentially executing everything themselves (where there are often multiple stakeholders)” (Travis, 2016). Travis's layout has provided us with an overview of what, why, where, and how due to the time constraints imposed by the fact that the majority of our applicants are older. What we discovered from each interview throughout the formative evaluation is listed below. For more details and verbatim documents (see Appendix E)

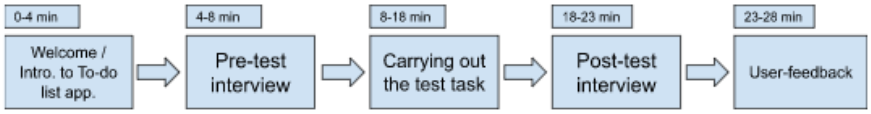
<table>
<tr><td colspan="2">Author:
Isa Phuyuthanon</td><td colspan="2">Contact:</td><td>Final date for comments:
August 3rd, 2022</td></tr>
<tr><td>Product under test:
A to-do list application prototype.</td><td rowspan="2">Test objectives:

See below</td><td>Participant:
Male age 65.</td><td rowspan="2">Test tasks:

Find out about the design layout

Find out about word choices

Find out about color choice

Find out about favorite/least favorite function</td><td>Responsibility:
Isa Phuyuthanon (Moderator)</td></tr>
<tr><td>Business case:
The test will address key questions/validate regarding the third prototype: Notebook for the next iteration. Failing to answer might increase the risk of developing a wrong product.</td><td>Equipment:
A laptop, the session will be recorded by zoom. The interview will be accompanied by a PowerPoint interview deck.</td><td>Location and date:
August 4th, 2022 via zoom.</td></tr>
<tr><td colspan="5">Procedure:
 
</td></tr>
</table>

.

Table 3: Usability Test Plan, Testing panel 1, Male age 65

<table>
<tr><td colspan="2">Author: takahisa Hashimoto</td><td colspan="2">Contact:</td><td>Final date for comments:</td></tr>
<tr><td>Product under test:
A to-do list application prototype.</td><td rowspan="2">Test objectives:

See below</td><td>Participant:
66, female, Japanese.
Part-time Caretaker</td><td rowspan="2">Test tasks:

Find out about the design layout

Find out about word choices

Find out about color choice

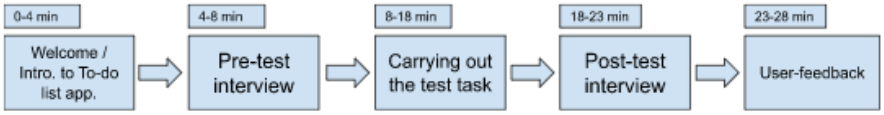
Find out about favorite/least favorite function</td><td>Responsibility:
Takahisa Hashimoto (Moderator)</td></tr>
<tr><td>Business case:</td><td>Equipment:
Only localized landing pages in Japanese and use figma to show sequence flows. Interview was recorded as a voice memo.</td><td>Location and date:
2022/Aug/16 18:00 - 18:45, her home.</td></tr>
<tr><td colspan="5">Procedure:
 
</td></tr>
</table>

Table 4: Usability Test Plan, Testing panel 2, Male age 66

<table>
<tr><td colspan="2">Author: takahisa Hashimoto</td><td colspan="2">Contact:</td><td>Final date for comments:</td></tr>
<tr><td>Product under test:
A to-do list application prototype.</td><td rowspan="2">Test objectives:

See below</td><td>Participant:
71, female, Japanese. Accounting for his husband business</td><td rowspan="2">Test tasks:

Find out about the design layout

Find out about word choices

Find out about color choice

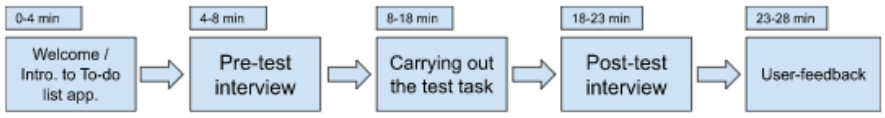
Find out about favorite/least favorite function</td><td>Responsibility:
Takahisa Hashimoto (Moderator)</td></tr>
<tr><td>Business case:</td><td>Equipment:
Only localized landing pages in Japanese and use figma to show sequence flows. Interview was recorded as a voice memo.</td><td>Location and date:
2022/Aug/16 19:00 - 19:35, her home.</td></tr>
<tr><td colspan="5">Procedure:
 
</td></tr>
</table>

Table 5: Usability Test Plan, Testing panel 3, Female age 71

Flow	Test objectives
Pre-test	
	To get started, can you please state your age and nationality ?
	Are you experiencing any difficulty with eyesight, hearing, memory or movement (incl. eyeglasses, hearing aid, joint pains, etc.) ?
	On a scale of 1 to 5 (1=not at all confident, 5=very confident), how would you rate your level of confidence in using computers and technology ?
	On average how many hours do you spend online every day ?
	What do you usually do online ?
	How do you keep track of your daily tasks ? What tool do you use ?
	What devices do you use (mobile/tablet, Android/Apple) ?
During test	
FLOW 1 (LIST)	Hover your mouse to the Weekend list, do you think this should be text or an icon ?
	If you wanted to create a list, how would you go about doing that ?
	How would you edit the title of the list?
	How would you delete a list, (Hover the mouse over Gardening)
	What motivated you to hover over the list and not click?
FLOW 2 (TASK)	If you click into the list, you will see a task, how would you create a new task to the list ?
	How would you edit the task ?
	If you have completed the task, how would you do it ?
FLOW 3(ARCHIVE)	What are you thinking as you view this Archive page ?
	If user hover or click asks: Why did you navigate to the hover or click at the list ? Do you expect to see some information ?
FLOW 4 (GUIDE)	What do you think is the functionality on the page you are viewing ?
FLOW 5 (HELPER)	How do you view the complete information for the contact ?
	I noticed you click the back arrow. Can you tell me why?
	If you need to add more contact, how do you go about doing that?
	Can you click on the name: Jack and edit his information
FLOW 6 (SETTING)	Can you think of an appropriate setting other than what you see on the screen ?
	Should the page contain more information ?
	Let the user choose the setting icon
Post-interview	
	How do you like the overall lay-out ?
	Is the color appropriate ?

Table 6: Test objective used during the Usability Test

We organized a total of four usability tests by gathering panel users who matched our target audience and profile. The initial attempt was carried out while displaying to them FIGMA-drawn step-by-step screen flows (see Appendix D). In order to understand their challenges and

confusion over our design and concept, we posed a series of questions to four seniors. The remaining usability tests were carried out on a laptop while using our browser-based application. In addition, we received input from four separate elders about the program that was operating on the device.

Details about our test methodology, their feedback, and our improvements are shown in the table below. We received negative comments on our initial designs, and the elders required our instruction for every single procedure. However, thanks to their candid criticism, our team was able to think about how to improve the following test cycles. Elders received less advice from our moderators during the most recent attempt than they had previously, and they expressed a desire to add their tasks to the list.

Attempts	Test Platform	Feedback from Elders and some major issues	Changes from previous Usability test
1	FIGMA	(Issue 1) Hard to understand how "ARCHIVE" works and what it's for. (Issue 2) Two Tabs "GUIDE" and "HELPER" look similar. Any difference? (Issues 3) The background color on the archive need to be contrast to that of the font	N/A (Not applicable)
2	Browser app	(Issue 4) Impossible to check my tasks after archiving the list	Switch from FIGMA to our 1st Browser app
3	Browser app	(Issue 5) I can't save my contact until all fields are filled out.	Moved ARCHIVE page to BOARD page to simplify the function (for issue 1 and issue 4) Added FAQ (Frequently Asked Questions) in GUIDE tab (for issue 2) Coloring is fixed for issues 3
4	Browser app	I can't probably find a long press function by myself. Add FAQ in GUIDE tab I want to keep entering new tasks after hitting the Enter button. However, the task button has to be pressed after moving the cursor. Further solution study required.	MOVED Helper page to SETUP so that users can configure all at the single page (for issue 2) Contact picture is set as optional (for issue 5)

Table 7:Result of the iterative usability tests and feedback and updated from elders

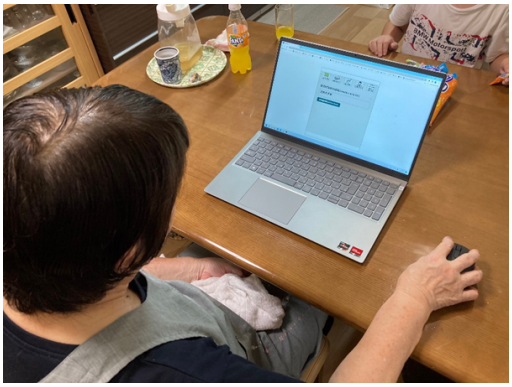
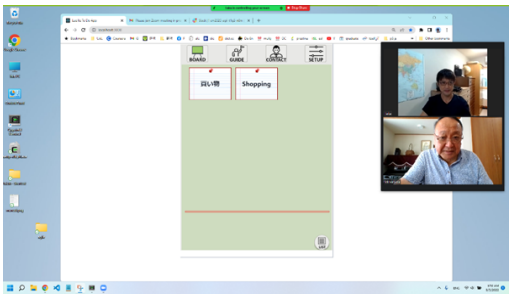
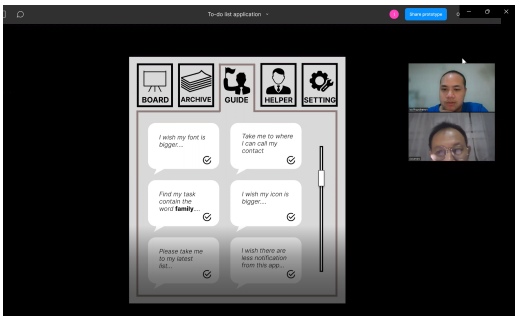
Pic 1: Our moderator visited elders and asked them how easily they could create their tasks on our app.	
Pic 2: Some elders participated in our usability test via an online zoom meeting.	 

Table 8: Pictures shows how usability tests were conducted

Summative Evaluation

We extensively tested our prototypes with actual users before sending the same group a survey asking them to review the finished product. They were asked to test our finished product and provide a score ranging from 1 (least satisfied) to 5 (most satisfied).

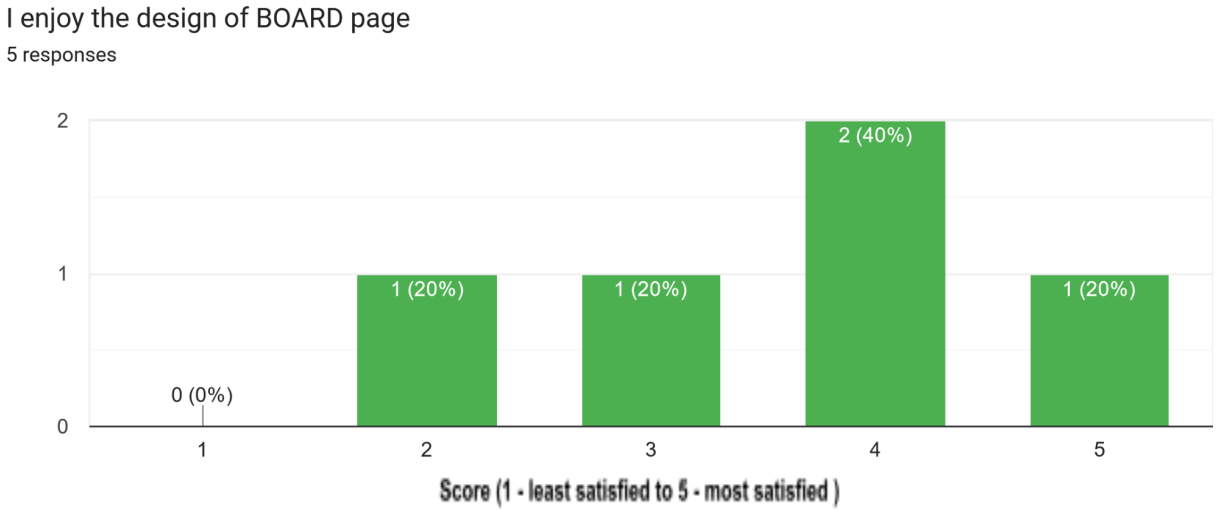


Figure 24 : Summative Evaluation “I enjoy the design of BOARD page”

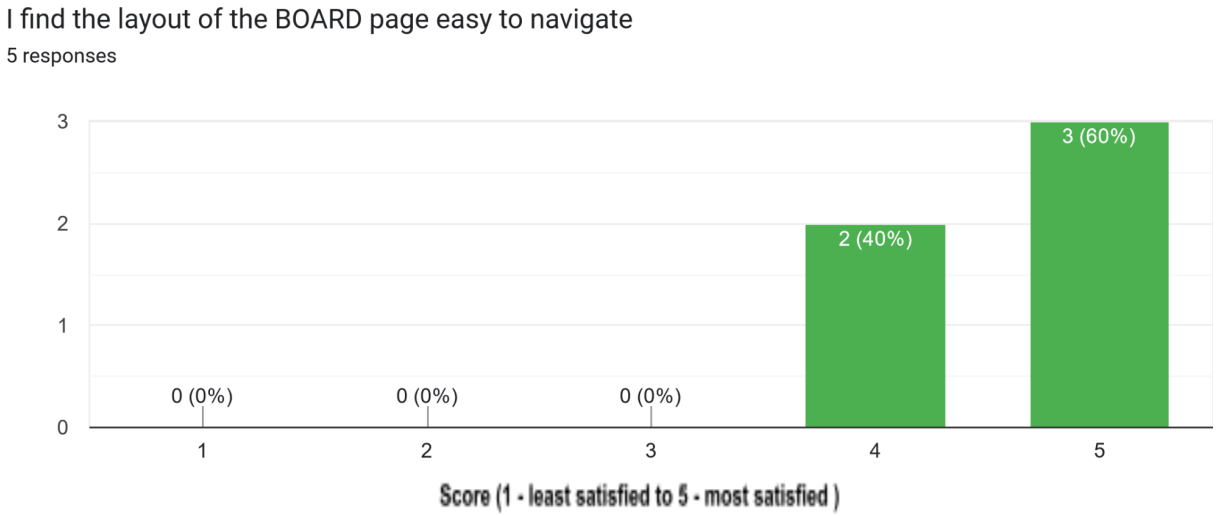


Figure 25 : Summative Evaluation “I find the layout of the BOARD page easy to navigate”

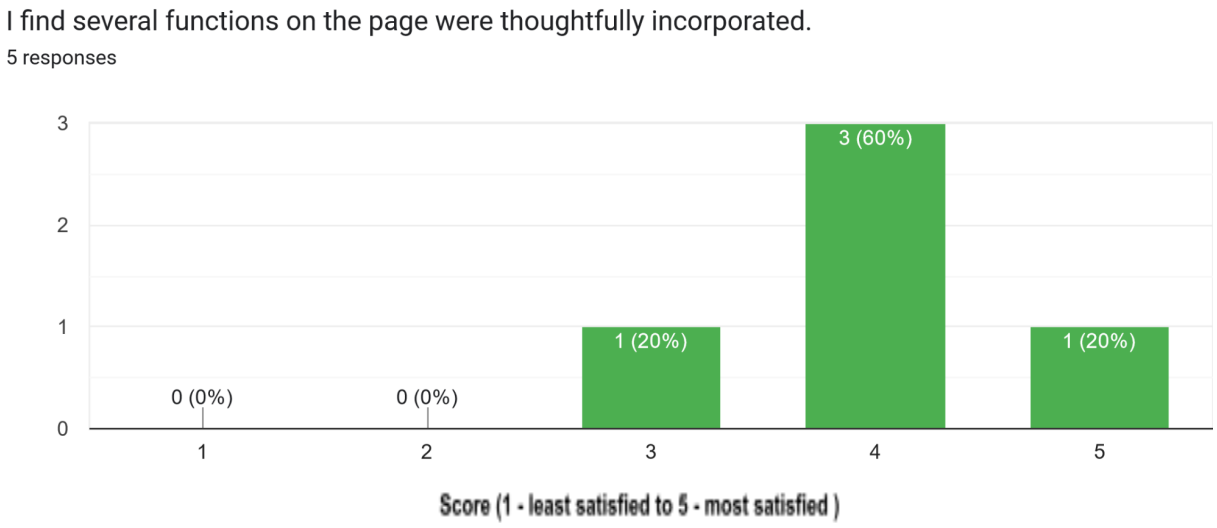


Figure 26 : Summative Evaluation “I find several functions on the page were thoughtfully incorporated”

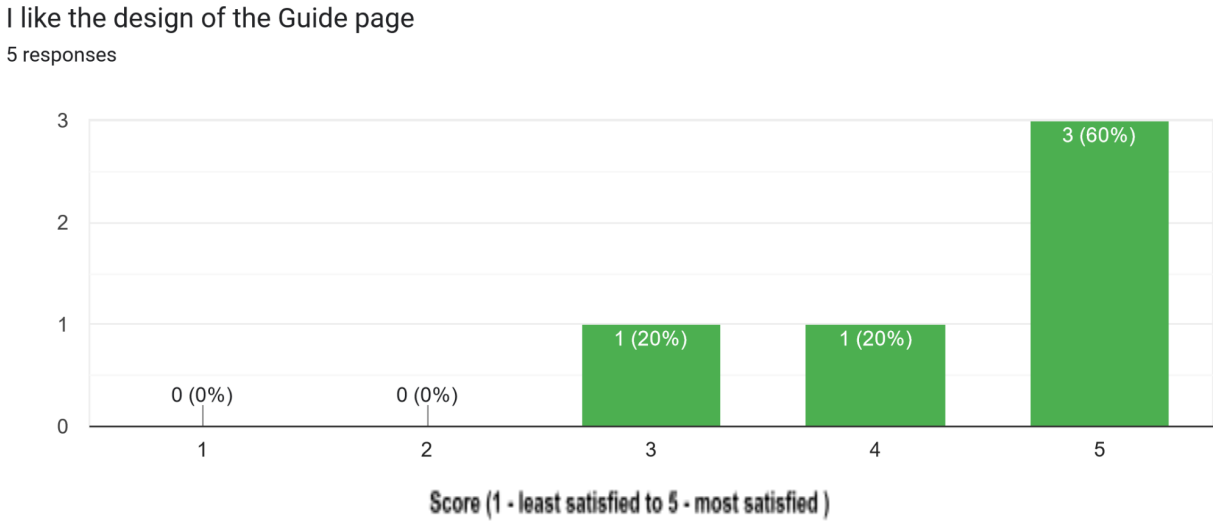


Figure 27 : Summative Evaluation “I like the design of the guide page”

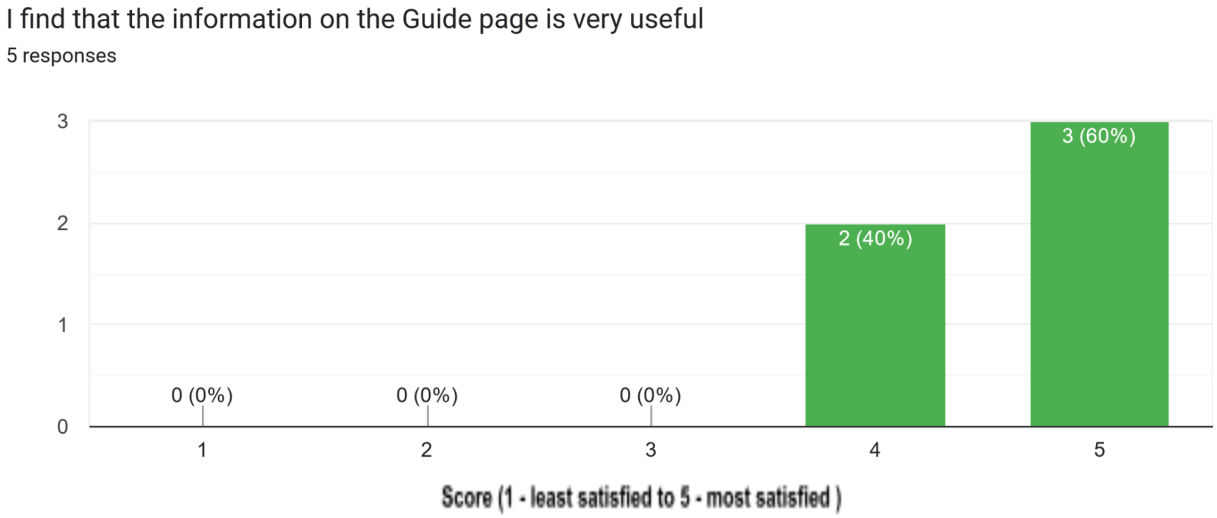


Figure 28 : Summative Evaluation “I find several adjustment (Font size, Brightness and Volume) to be very useful for this application”

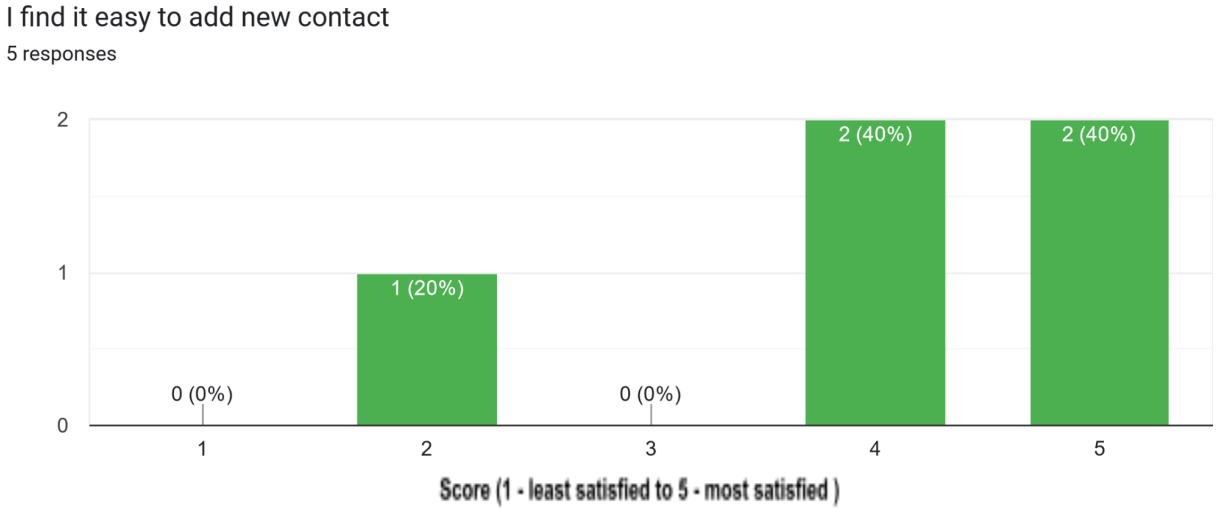


Figure 29 : Summative Evaluation “I find it easy to add new contact ”

With an average score of 3.6 when asked to rate the board page's design, most respondents said they had an "enjoyable" experience; but, when asked whether the board page was simple to navigate, the average score rose to 4.5. The data shows that the adjustments made between the initial prototype and the final product were carried out satisfactorily in design. The

respondents gave our features an average score of 4, thinking that they had been "thoughtfully incorporated." This demonstrates that the features we've created have undergone research-based testing and have a solid foundation.

Respondents gave the design of the Guide page an average score of 4, indicating that they 'liked' the page. The majority of users have praised the notion of guided questions throughout the development process, from the first prototype to the finished product. The high rating merely further demonstrates the value of this feature. Respondents gave the information on the guide page an average score of 4.6, classifying it as "helpful". This suggests that the inquiries were deliberately chosen to address and meet the concerns of an elderly users.

The respondents gave the setup page's design an average score of 4.2 and gave the experience of using the page, which includes the ability to quickly add new contacts, an average score of 4. This demonstrates that the improvements we made between the initial prototype and the end product were successful. With an average score of 4.6, the respondents considered changing the font size, brightness, and volume to be "helpful." This shows that the options we provided are sufficient for senior users.

The study's main drawback is the small sample size; however, with a larger sample size and more testers, we might increase the likelihood of product adjustments. One thing to note is that everyone who tests the product is between the ages of 60 and 73, so with this modest response from the correct market segment, we can really draw attention to the apparent features and design.

Conclusion and Evaluation Summary

Where are we now?

- We were able to build the application detailed in our specifications bar some advanced features which we plan on including in our future development endeavors.
- We were able to integrate findings from our literature review, market survey and user survey into our application design and development.
- The app as a product has been through a series of testing and changes and the final product we have has been tested for successful usability.

Place of our product?

- Lucille is a unique app in that it is one of the few apps specifically designed for elderly and most likely the only to-do app designed for elderly.
- Lucille was

Further development:
Below are some of the features we proposed, researched and tried to implement but could not successfully complete the implementation. These are also our potential areas of further development.

1. Sentence completion while filling in tasks.
2. Animation of pinned lists flying towards the archived area when archive-button is pressed.
3. Implementation of “cross” gesture to delete items.
4. Vibration and sounds when button pressed for haptic feedback.

User Manual

Instructions for creating a to do list on

Lucille Application

Getting Started

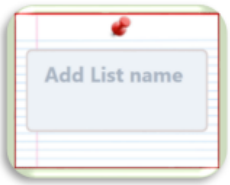
Access our to do list application at:

[Lucille Application]

Adding List

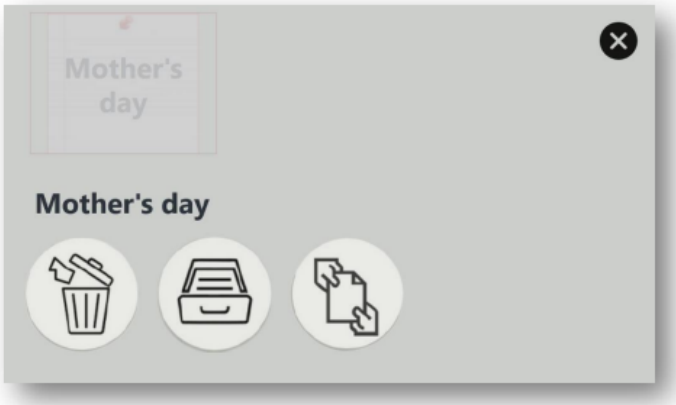
On the Board page, add your to do list by clicking 'LIST' located at the bottom of the screen.

When a notebook text box appears, put a list title in place of 'Add List Name'.



Deleting, Archiving and Sharing List

Click and press your list for 3 seconds to access delete, archive, and share buttons.



Click this bin icon to delete the list.

Click this drawer icon to archive your list.

Click this sharing icon to share your list to saved contact.

Click this icon to go back to the board.

Adding and Editing Task

Click your list to access your task. To add a new task, click a 'TASK' button on the bottom of the screen.

When a text box appears, simply type down your task.

To edit, double-click your task.

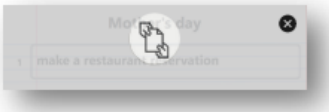
Completing Task

When you finish your task, tick on the 'DONE' box at the end of each task.

Unticking means you have not completed the task.

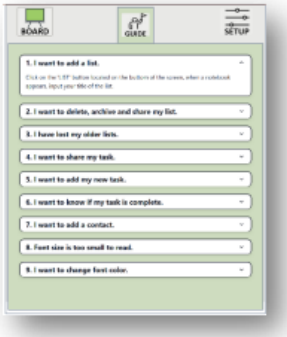
Sharing Task

Click and press your task for 3 seconds, share button will appear. Share your task with saved contact.



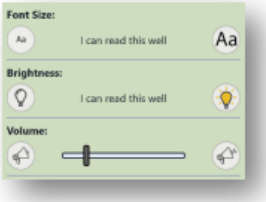
Guide

For help with 'How to..', access our Guide page.



Font size, Brightness and Volume adjustment.

Access SETUP page to adjust Font Size, Brightness and Volume.



Simply click on each icon to increase and decrease.

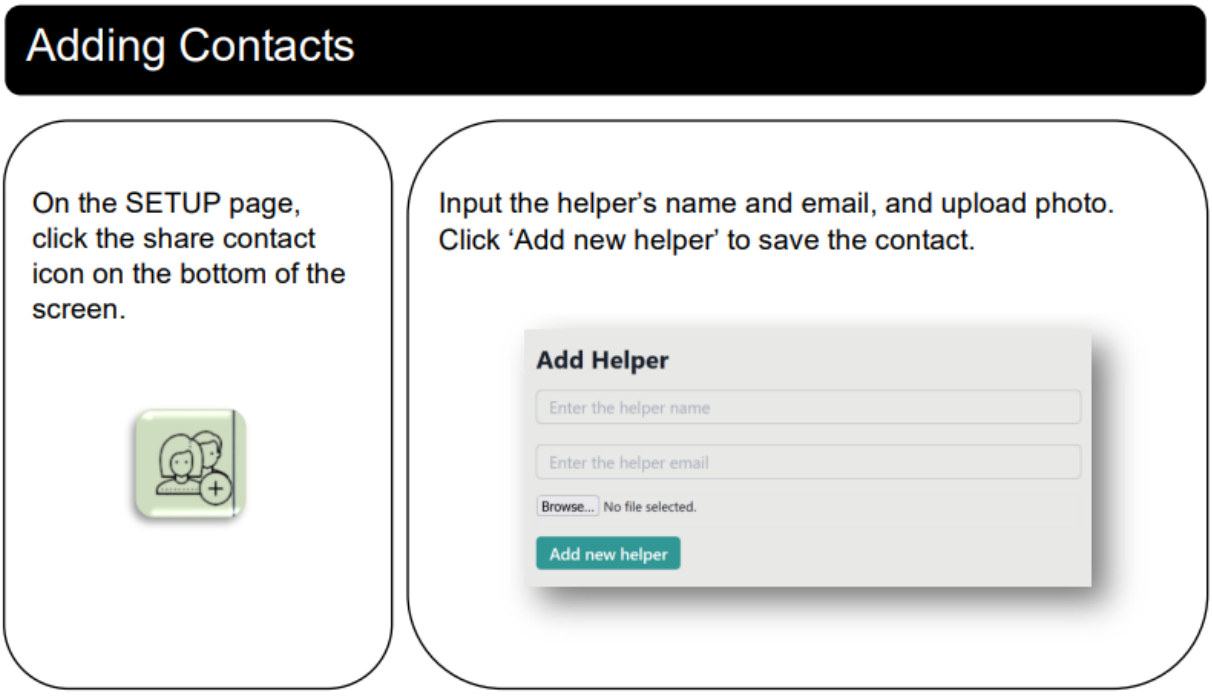


Figure 30: Instruction for creating a to-do list on Lucille

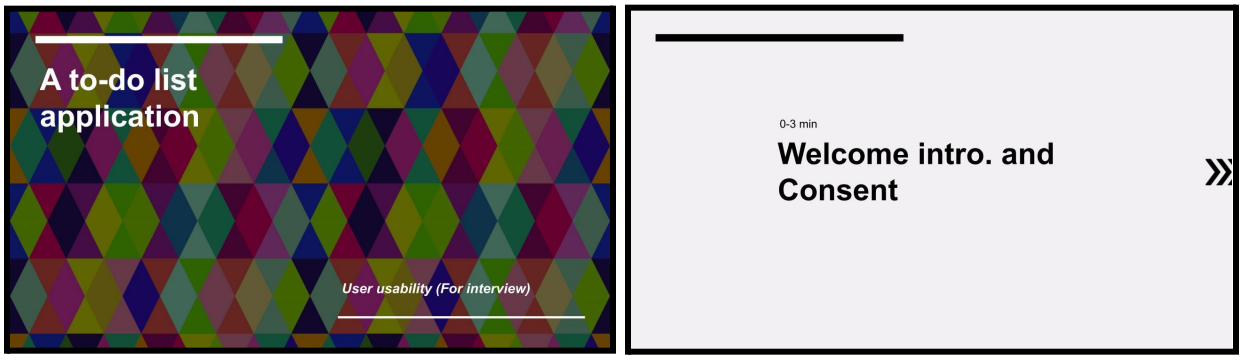
Appendix A: Persona design

Akiko 65 y/o Akiko lives in Miyama with her husband, Kazuo, where they own a farm and a small B&B. Their daughter, a marketing professional, works in nearby Kyoto. They have three grandkids, who occasionally visit them on weekends and holidays. They're both in good health, but the physical nature of the job means they now employ a handful of workers across their estate. They own a kei truck and a small family car, which they both drive. They socialize on a weekly basis in the local community center and in town events, and occasionally travel to nearby towns, as well as to Kyoto and Osaka. They've been abroad, but they're not fluent English speakers and have difficulty arranging and managing trips. They have a reliable internet connection at home and around town, but occasionally travel in areas with limited cellular coverage.
Johann 75 y/o Johann is a retired teacher from Cologne. Divorced for 20 years, he never remarried, but remains open to the notion. He has a son in Munich and a daughter who lives nearby, and four grandchildren. He likes to think of himself as curious and open to new experiences, and still enjoys everything the city has to offer. He doesn't drive but travels frequently, relying instead on the local public transport network and rail. He wears eyeglasses and takes medication to lower his blood cholesterol levels. He recently received an e-book reader as a gift; he finds it an interesting but imperfect experience, and still likes to pick up a book or a (printed) newspaper whenever he has the chance. He often uses public wi-fi networks in places like libraries and cafes.

(Sharma *et al.*, 2022, sec. Users and personas)

Appendix B: PowerPoint Presentation used during the Usability Test.

Figure 1 shows the PowerPoint that will be used during the usability test to get the test panel going and to explain what to expect during the interview.



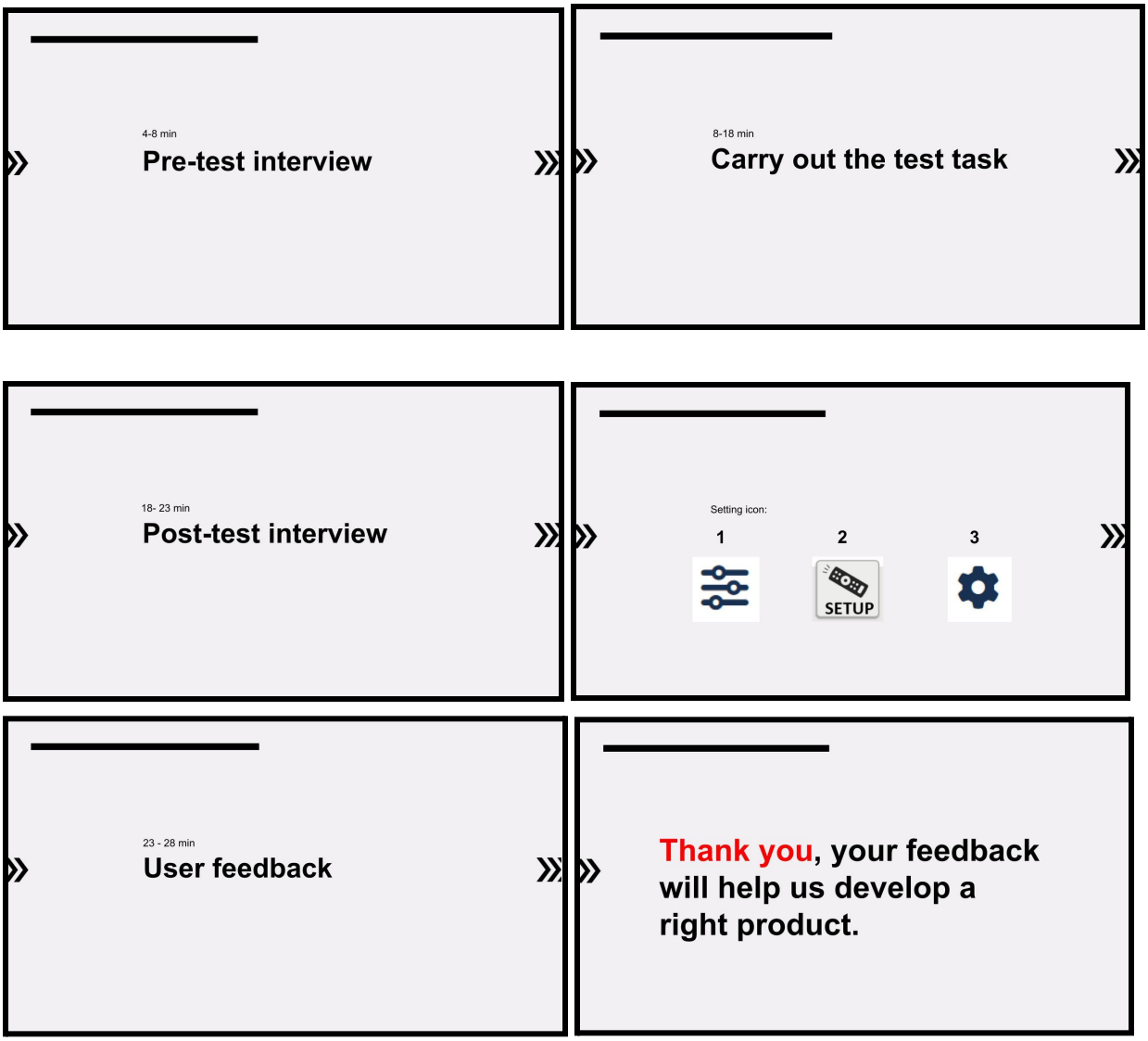
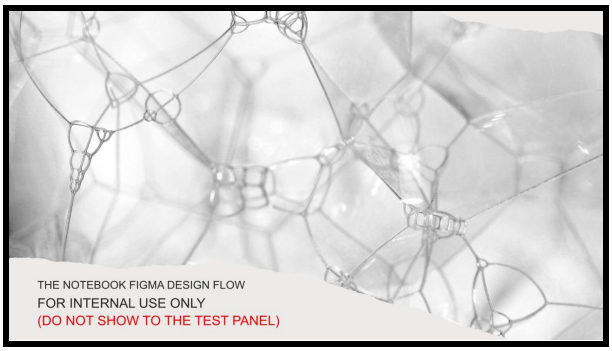


Figure 1: PowerPoint used during the usability test

Appendix C: Figma flow used by the testing team

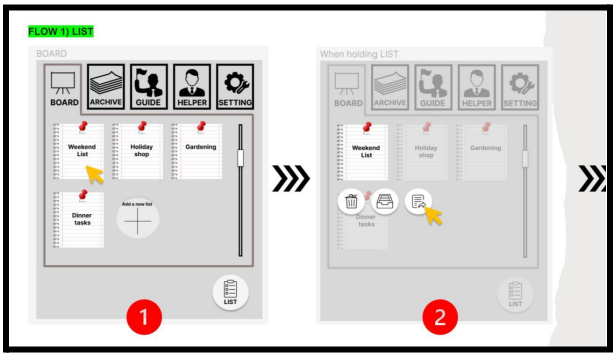
The PowerPoint below in figure 1 is for an internal testing team to use to evaluate how well they comprehend our application on Figma; it is for internal usage only and is not visible to the test panel.

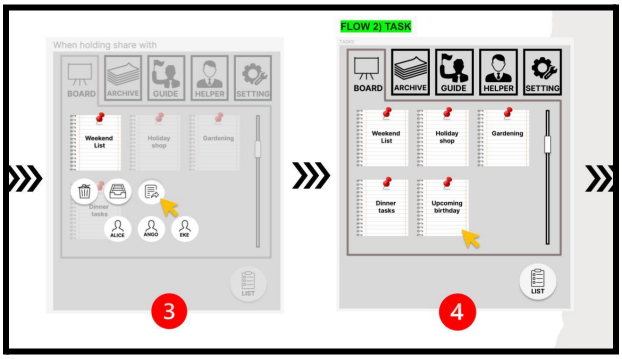


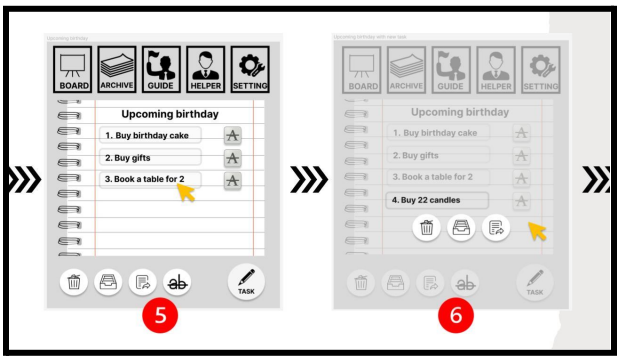
THE NOTEBOOK FIGMA DESIGN FLOW
FOR INTERNAL USE ONLY
(DO NOT SHOW TO THE TEST PANEL)

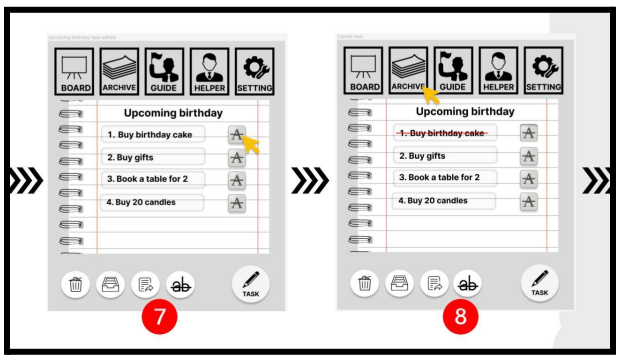
Link to access Figma design of the prototype: The Notebook

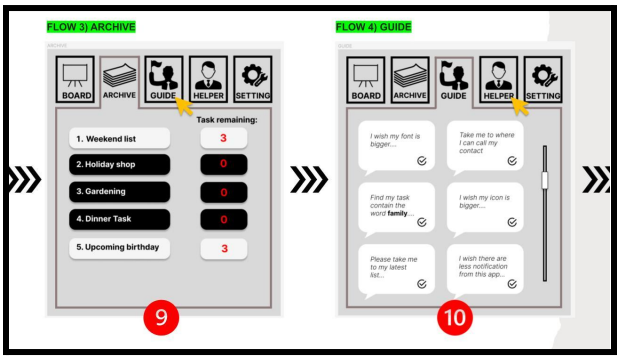
Please note that, currently the navbar for every page is clickable. This is meant for user to click so that we learn their behavior during the interview e.g., what motivate them to click when we did not guide them to click.

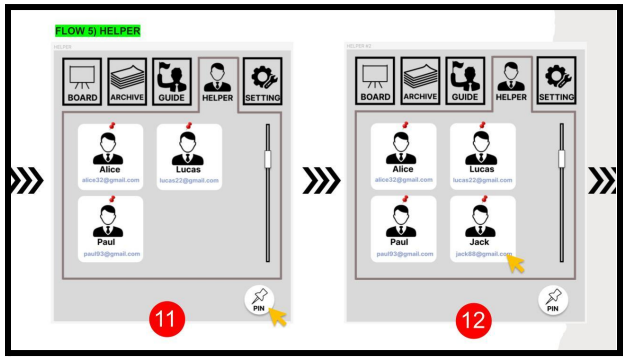












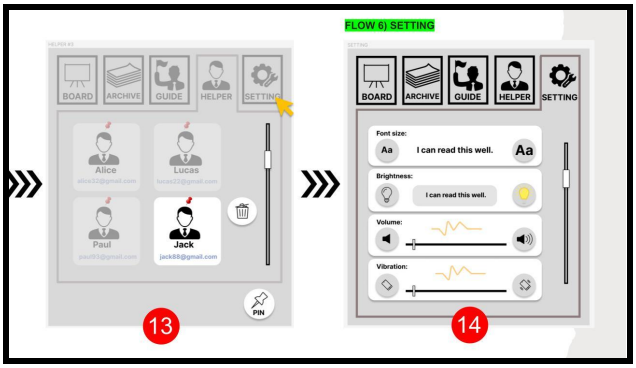


Figure 30: Figma flow for internal use

Appendix D: Figma design used on the first iteration.

Figure 1 shows Figma design of our first prototype, The Notebook. Access the Figma design here: [To-do list application prototype 1]

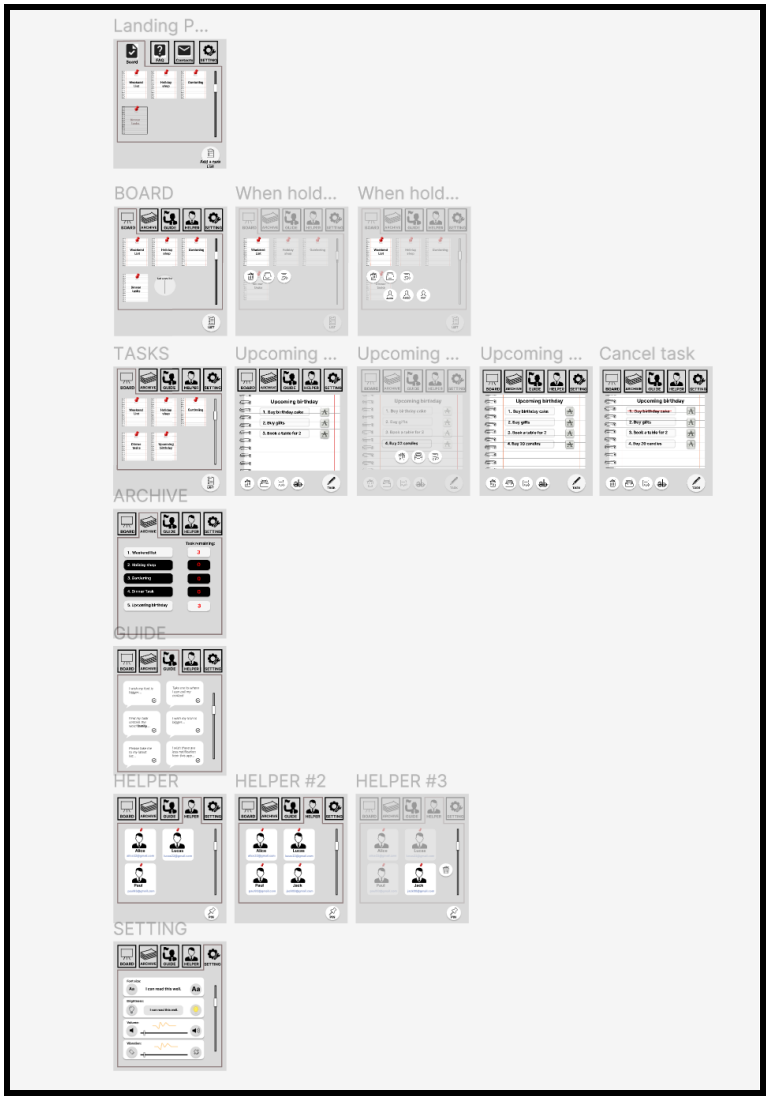


Figure 31: Figma design of the the first prototype of the Notebook

Appendix E: Usability Test Result (Verbatim)

We recorded the full session of the usability test, and the test panel's response is provided below in figure 1 - 3 (verbatim) including test panel 1 - 3.

Test Panel #1:

Flow	Questions	Answer	Observation
Pre-test			
	To get started, can you please state your age and nationality ?		

	Are you experiencing any difficulty with eyesight, hearing, memory or movement (incl. eyeglasses, hearing aid, joint pains, etc.) ?	Nothing, I am wearing glasses, farsighted. Sometime if the screen is too bright I can not focus very well.	
	On a scale of 1 to 5 (1=not at all confident, 5=very confident), how would you rate your level of confidence in using computers and technology ?	4	
	On average how many hours do you spend online every day ?	4 hours	
	What do you usually do online ?	reading online material and attend online conference	
	How do you keep track of your daily tasks ? What tool do you use ?	I use my notebook to keep track of my to do lists. Sometimes I also use reminder on my phone.	
	What devices do you use (mobile/tablet, Android/Apple) ?	When I am on the go, I use my mobile phone and tablet	
During test			
FLOW 1 (LIST)	Hover your mouse to the Weekend list, do you think this should be text or an icon ?	I think it should have both text and icon	
	If you wanted to create a list, how would you go about doing that ?	I clicked the list button.	User took about 50 secnds - 1 min to find the button and confuse about creating a new list with already made list but eveutually found the button
	How would you edit the title of the list?		User navigate himself to click the list. Moderator asked why, he replied, he is hoping there is an edit button inside the list. He continued to say there is no button visible

			in the list page so moderator asked him to long-press the button
	How would you delete a list, (Hover the mouse over Gardening)		After being introduce that the list is long-press enable, user long-pressed the list and a choice to delete appear.
	What motivated you to hover over the list and not click?	Edit and delete should be basic functionality there must be a way to make it easily visible to the user	
FLOW 2 (TASK)	If you click into the list, you will see a task, how would you create a new task to the list ?	I will click on the pencil icon	
	How would you edit the task ?		User keep clicking the task. When asking what motivate them to click the task, he said he was hoping to have the same function as when asking to click inside the list.
	If you have completed the task, how would you do it ?	This icon [strike icon]	User is able to navigate to the strike icon very fast. He added that he likes this idea as it is very intuitive.
FLOW 3 (ARCHIVE)	What are you thinking as you view this Archive page ?	I am not sure but, I think it's a summary of task remaining on the list.	The user has made some important comment here regarding the color: He suggest that the background colored green should contrast with the white font to see it clearly.
	If user hover or click asks: Why did you navigate to the hover or click at the list ? Do you expect to see some information ?		User click on the remaining task, when asking why he click the task remaining, he said he would expected to

			see the actual remaining tasks.
FLOW 4 (GUIDE)	What do you think is the functionality on the page you are viewing ?	Actually I'm not sure, looking from the title of the menu I think it is some kind of a guiding questions but not sure whats the main functionality is	The user has given his feedback about the title of the this functionality. He suggested to us to consider changing the title of this menu but he really likes this functionality, and it could be our USP for the app.
FLOW 5 (HELPER)	How do you view the complete information for the contact ?	Click on the contact where name is located	The test is being done on the previous version of the app.
	I noticed you click the back arrow. Can you tell me why?	I understand that black arrow means to go to the previous page.	
	If you need to add more contact, how do you go about doing that?	Just click "Add contact"	
	Can you click on the name: Jack and edit his information		User is able to edit the contact, he added that the button and the instruction is clearly stated.
FLOW 6 (SETTING)	Can you think of an appropriate setting other than what you see on the screen ?	I think this is all we need for setting	
	Should the page contain more information ?	No	
	Let the user choose the setting icon	3rd icon	His reason for choosing number 3 is that his background is in IT and most of the software he uses in the past uses this style of icon.
Post-interview			

	How do you like the overall lay-out ?	I think it is a good to do app however I would suggest to change the title of the guiding page. Also is there a way to group contact into Family, Colleague and Friends ? I like the functionality of the guiding page a lot it would be cool to see it being implemented.	
	Is the color appropriate ?	Color is appropriate except on the archive page where I suggest the background and font need to be contrast.	

Figure 32: Usability Test result for test panel #1

Test panel #2:

Flow	Questions	Answer	Observation
Pre-test			
	To get started, can you please state your age and nationality ?	66, Japan	
	Are you experiencing any difficulty with eyesight, hearing, memory or movement (incl. eyeglasses, hearing aid, joint pains, etc.) ?	Hard to read small letters on books these days.	
	On a scale of 1 to 5 (1=not at all confident, 5=very confident), how would you rate your level of confidence in using computers and technology ?	1	
	On average how many hours do you spend online every day ?	2 hours	
	What do you usually do online ?	email, SNS (Line) with her daughters, Watch youtube	
	How do you keep track of your daily tasks ? What tool do you use ?	Put schedules on paper calendar	
	What devices do you use (mobile/tablet, Android/Apple) ?	Her own smartphone	
During test			
	(show a panel our landing page and ask) what do you think this app is?	Support to make a reservation for something? Because "helper", "guide" encourage a booking service	
FLOW 1 (LIST)	Hover your mouse to the Weekend list, do you think this should be text or an icon ?	she thought it's a button after thinking a few minutes.	
	If you wanted to create a list, how would you go about doing that ?	she didn't understand a list itself without any explanation.	
	How would you edit the title of the list?		
	How would you delete a list, (Hover the mouse over Gardening)	she said "there is no delete button, so I can't".	
	What motivated you to hover over the list and not click?		

FLOW 2 (TASK)			
	If you click into the list, you will see a task, how would you create a new task to the list ?	she didn't understand my question. But she said "I want to make todo items for her job", not private stuff stored on my smartphone.	
	How would you edit the task ?	she said "no edit/delete required for me because I need to check same todo item every week"	
	If you have completed the task, how would you do it ?	Since I need to keep doing same todo next week or so, they would be never completed (She wants to keep showing the same todos as her routine work)	
FLOW 3 (ARCHIVE)			
	What are you thinking as you view this Archive page ?	Probably I can see my old todo??	
	If user hover or click asks: Why did you navigate to the hover or click at the list ? Do you expect to see some information ?	She didn't understand what the screens are for, even I explained "tasks remaining". "holiday shop 0" means "no shop infomration available??"	
FLOW 4 (GUIDE)			
	What do you think is the functionality on the page you are viewing ?	Probably learning my preference with AI technologies and	

		suggest something?	
FLOW 5 (HELPER)			
	How do you view the complete information for the contact ?	She said the icons were for AI agents. She expected the AI agents could answer her questions.	
	I noticed you click the back arrow. Can you tell me why?		
	If you need to add more contact, how do you go about doing that?		
	Can you click on the name: Jack and edit his information		
FLOW 6 (SETTING)			
	Can you think of an appropriate setting other than what you see on the screen ?	Why do we need the settings? Smartphone already have the funcitons.	
	Should the page contain more information ?	No	
	Let the user choose the setting icon	I don't think it's hard to read.	
Post-interview			
	How do you like the overall lay-out ?	Hard to use. too much functions.	
	Is the color appropriate ?		
	What do you think could be improved for this interview?	Can I directly make a todo first? Why do I need to make a list first?	

Figure 33: Usability Test result for test panel #2

Test panel #3:

Flow	Questions	Answer	Observation
Pre-test			
	To get started, can you please state your age and nationality ?	71, Japan	
	Are you experiencing any difficulty with eyesight, hearing, memory or movement (incl. eyeglasses, hearing aid, joint pains, etc.) ?	I took cataract surgery last year though, I have no problems to see things.	
	On a scale of 1 to 5 (1=not at all confident, 5=very confident), how would you rate your level of confidence in using computers and technology ?	3	
	On average how many hours do you spend online every day ?	2 hours	
	What do you usually do online ?	business and use excel, word etc	
	How do you keep track of your daily tasks ? What tool do you use ?	write todo at memo pad	
	What devices do you use (mobile/tablet, Android/Apple) ?	old fashioned phone, laptop and iPad	
During test			
	(show a panel our landing page and ask) what do you think this app is?	An app to make a video? (why?) archive videos, guides to show how to make a video etc.	
FLOW 1 (LIST)	Hover your mouse to the Weekend list, do you think this should be text or an icon ?	button	
	If you wanted to create a list, how would you go about doing that ?	touch LIST	
	How would you edit the title of the list?	touch "weekend list", probably there is an edit button.	

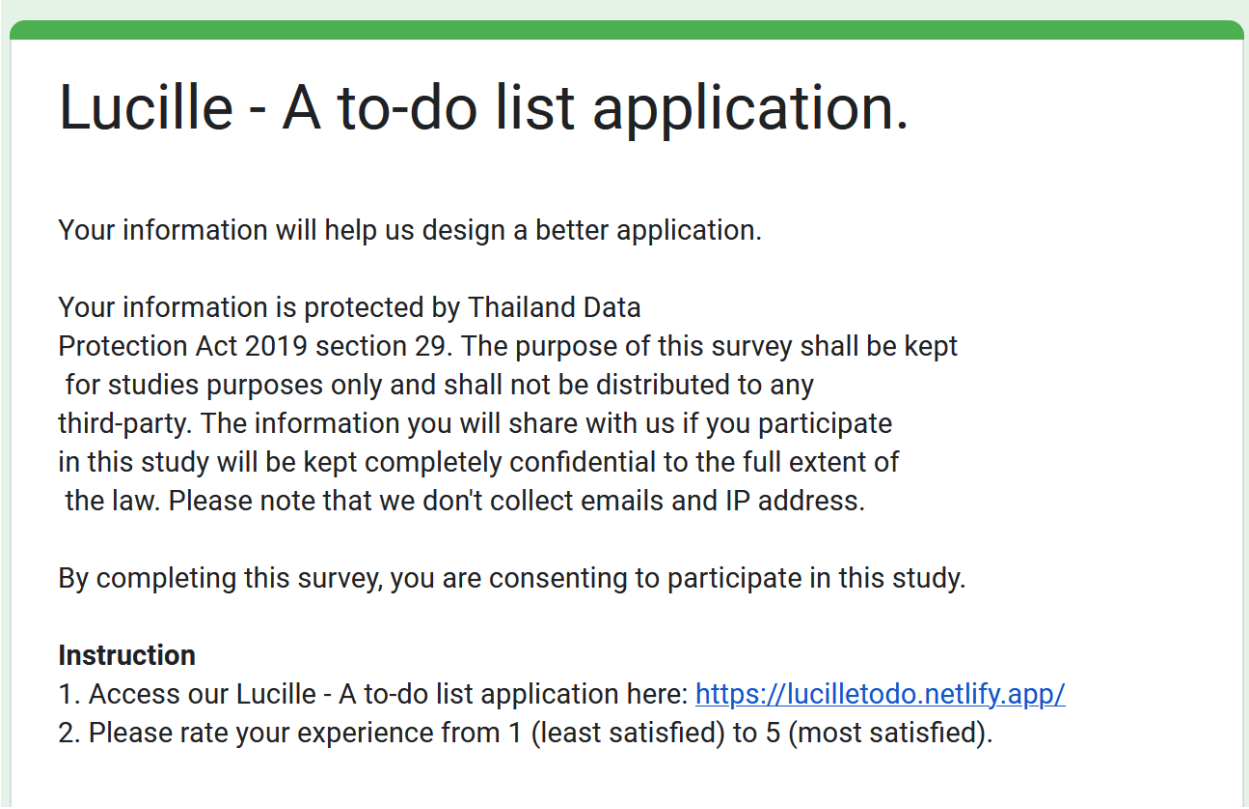
	How would you delete a list, (Hover the mouse over Gardening)	i touch *weekend list" and press "delete" or "back" buttons on keyboards	
	What motivated you to hover over the list and not click?		
FLOW 2 (TASK)			
	If you click into the list, you will see a task, how would you create a new task to the list ?	select a last item and press enter button. After the link break, add a new task	
	How would you edit the task ?	Book a table for 2 --> select the item and move the cursor at the end and change 2 to 3 for instance.	
	If you have completed the task, how would you do it ?	select item and press "delete" button on keyboard	
FLOW 3 (ARCHIVE)			
	What are you thinking as you view this Archive page ?	showing old records?	
	If user hover or click asks: Why did you navigate to the hover or click at the list ? Do you expect to see some information ?	"weekend list 3" means "three people have to do something on this weekend?"	
FLOW 4 (GUIDE)			
	What do you think is the functionality on the page you are viewing ?	If I lost my way, the app can help where to go. (why?) The icon show a tour guide with flag.	
FLOW 5 (HELPER)			
	How do you view the complete information for the contact ?	probably guide to a hotel? (why?) The icon shows "person with a tie", It looks like a "hotel concierge". No	

		idea why there are similar buttons like "guide" and "helper".	
	I noticed you click the back arrow. Can you tell me why?		
	If you need to add more contact, how do you go about doing that?		
	Can you click on the name: Jack and edit his information		
FLOW 6 (SETTING)			
	Can you think of an appropriate setting other than what you see on the screen ?	we can change the font size by browser or laptop. why do we need the settings?	
	Should the page contain more information ?	No, just need more bigger font size.	
	Let the user choose the setting icon		
Post-interview			
	How do you like the overall lay-out ?		
	Is the color appropriate ?	no issues.	
	What do you think could be improved for this interview?		

Figure 34: Usability Test result for test panel #3

Appendix F: User survey consent statement and Usability Test consent form.

Participation in the users survey was governed by Section 29 of the Thai Personal Data Protection Act B.E. 2562 (2019) (“PDPA”). Participants were notified of the purpose of the survey and the legal background, and were assured that their data will be kept confidential and not shared with third parties (Fig. 1). They were then asked to sign a consent form (Fig. 2).



Lucille - A to-do list application.

Your information will help us design a better application.

Your information is protected by Thailand Data Protection Act 2019 section 29. The purpose of this survey shall be kept for studies purposes only and shall not be distributed to any third-party. The information you will share with us if you participate in this study will be kept completely confidential to the full extent of the law. Please note that we don't collect emails and IP address.

By completing this survey, you are consenting to participate in this study.

Instruction

1. Access our Lucille - A to-do list application here: https://lucilletodo.netlify.app/
2. Please rate your experience from 1 (least satisfied) to 5 (most satisfied).

Figure 35: Finished product user survey consent statement

Consent to take part in a user's testing interview for To-do list application.

I..... voluntarily agree to participate in user's testing interview.

- I understand that even if I agree to participate now, I can withdraw at any time or refuse to answer any question without any consequences of any kind.
- I understand that I can withdraw permission to use data from my interview within two weeks after the interview, in which case the material will be deleted.
- I have had the purpose and nature of the study explained to me in writing and I have had the opportunity to ask questions about the study.
- I understand that participation involves answering questions for a To-do list application
- I understand that I will not benefit directly from participating in this user's testing interview.
- I agree to my interview being recorded.
- I understand that all information I provide for this study will be treated confidentially.
- I understand my name, age and nationality will be including in the project proposal.
- I understand that disguised extracts from my interview may be quoted in a project proposal for a To-do list application.
- I understand that signed consent forms and recordings will be retained in Google drive named *Agile_group* (https://drive.google.com/drive/folders/14ZJFB-Oj_Yt2DZB2rn1ylutsAm2qFrKm), YouTube for private viewing until the university have confirmed grades for CM2022.
- I understand that under Thailand Data Protection Act 2019, I am entitled to access the information I have provided at any time while it is in storage as specified above.

Signature of participant

---------- ----------
Signature of participant Date

Signature of researcher

I believe the participant is giving informed consent to participate in this study

---------- ----------
Signature of researcher Date

Figure 36: Interview consent form

Appendix G: Technology stack proposal

Background

Project priorities

Our priorities center on the production of a useful, usable and satisfying UX informed by up-to-date research (see project goals). This means that anything not directly relevant to UX is of secondary importance.

Problem statement

We are at the end of our second sprint, which was preceded by several weeks that were dedicated to preparatory work on React. By this point (both by the course and the project plan) we should've had a functional prototype and started user testing; instead we have medium and high-fidelity designs, a data model and a testing plan, but no prototype to test. Since we are less than a month away from the deadline, we have to quickly reconsider our options and adjust, so we don't find ourselves at the finish line with no product.

Since the rest of the project is on schedule, and judging by input from the team, I believe the problem has to do with our choice of technology stack.

Current technology stack

Our current technology stack comprises the following (as forked from AndyW22's source):

- TypeScript
- React
- Redux
- Material UI
- AWS Amplify

Of these, only React and Amplify were explicitly *chosen*; the rest were a byproduct of the source we forked, and neither was chosen as part of a thorough review of available solutions. Three of the technologies required learning from most of us (TS, React and MUI), and two are usable by only one team member (Redux and Amplify). This means we have issues with fluency and work distribution, which result in significant slowdowns.

Alternatives to the current technology stack

Our program has to perform in three domains across both the backend and frontend: data handling, design, and user-facing behaviors. The following alternatives try to address all three, on both "ends" of the app.

Backend[7]

1. Express + MySQL:
    a. Pros: familiar and straightforward (taught at DNW), especially when combined with EJS.
    b. Cons: requires hosting (may be readily available, but need to check) and schema design; some functionality (authentication) may have a learning curve; requires building from scratch?
2. Other backend frameworks:
    a. Pros: designed to integrate with React; easy deployment to Vercel, Fly.io, and others; templates available; plenty of frameworks to choose from.
    b. Cons: some learning curve compared to Express.
3. No backend (client-side storage):
    a. Pros: very simple to do using an IndexedDB wrapper.
    b. Cons: no cloud sync and backup (limited to one device, with sharing through email or some dedicated API, and limited resilience).

Frontend[8]

1. EJS + CSS framework (demo):
    a. Pros: familiar and straightforward (taught at DNW); plenty of frameworks to choose from (also here).
    b. Cons: essentially stateless - may require some programming tricks to appear dynamic (eg. routing), especially if not using a backend; SCSS (if needed) has a mild learning curve.
2. Frontend router / framework + CSS framework:
    a. Pros: suitable for dynamic applications; familiar - closer to plain JS/Express than React and JSX; plenty of frameworks to choose from.
    b. Cons: some learning curve, depending on the framework; SCSS (if needed) has a mild learning curve.
3. React + component library (demo):
    a. Pros: essentially dynamic; encompasses both behavior and design; plenty of frameworks to choose from.
    b. Cons: some learning curve, depending on the library; may require a router.
4. Vue / Svelte + component library:
    a. Pros: may be easier to work with than React.
    b. Cons: some learning curve; already invested in React; may require a router.

[7] Also see *State of JS 2021*.
[8] Also see *State of JS 2021* and *State of CSS 2021*.

Considerations

Of the above alternatives, I believe we should choose one that maximizes productivity by way of functionality and ease of use, with a view to having a minimal viable product (MVP) by the end of the month.

Bibliography

Adebayo, S. (2022) ‘Chakra UI’. Available at: https://chakra-ui.com (Accessed: 11 September 2022).

‘Arrested Development’ (2003). 20th Century Fox Television.

Beck, K. *et al.* (2001) ‘Agile Manifesto’. Agile Alliance. Available at: https://www.agilealliance.org/agile101/the-agile-manifesto/.

Brooks, F.P. (1995) *The Mythical Man-Month: Essays on Software Engineering*. Anniversary ed. Reading, Mass: Addison-Wesley Pub. Co.

Card, S.K., Robertson, G.G. and Mackinlay, J.D. (1991) ‘The information visualizer, an information workspace’, in *Proceedings of the SIGCHI Conference on Human factors in computing systems*, pp. 181–186.

Doran, G.T. (1981) ‘There’s a SMART way to write management’s goals and objectives’, *Management review*, 70(11), pp. 35–36.

Dung Nguyen, Stephen Wong, and Mark Husband (2008) *Principles of Object-Oriented Programming*. Connexions. Available at: http://archive.org/details/ost-computer-science-ooprogramming (Accessed: 2 September 2022).

Fahlander, D. (2022) ‘Dexie.js’. Available at: https://dexie.org/ (Accessed: 11 September 2022).

Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter (no date). Available at: https://eslint.org/ (Accessed: 4 September 2022).

‘Git’ (2022). Software Freedom Conservancy. Available at: https://git-scm.com/ (Accessed: 11 September 2022).

‘GitHub’ (2022). GitHub, Inc. Available at: https://github.com (Accessed: 11 September 2022).

‘Google Drive’ (2022). Google. Available at: https://drive.google.com/ (Accessed: 11 September 2022).

Gould, J.D. and Lewis, C. (1985) ‘Designing for usability: key principles and what designers think’, *Communications of the ACM*, 28(3), pp. 300–311. Available at: https://doi.org/10.1145/3166.3170.

JavaScript Decorator Design Pattern - Dofactory (2022). Available at: https://www.dofactory.com/javascript/design-patterns/decorator (Accessed: 2 September 2022).

‘Jira’ (2022). Atlassian. Available at: https://www.atlassian.com/software/jira (Accessed: 11 September 2022).

Leonardi, C. *et al.* (2008) ‘Designing a familiar technology for elderly people’, *Gerontechnology*, 7(2), p. 151. Available at: https://doi.org/10.4017/gt.2008.07.02.088.00.

Most Popular Electronics Worldwide [July 2022 Update] (2022). Available at: https://www.oberlo.com/statistics/most-popular-electronics (Accessed: 5 September 2022).

Most Popular Web Browsers in 2022 [Jun ’22 Update] | Oberlo (2022). Available at: https://www.oberlo.com/statistics/browser-market-share (Accessed: 5 September 2022).

Neves, B.B. and Amaro, F. (2012) ‘Too Old For Technology? How The Elderly Of Lisbon Use And Perceive ICT’, *The Journal of Community Informatics*, 8(1). Available at: https://doi.org/10.15353/joci.v8i1.3061.

Nunes, F., Silva, P.A. and Abrantes, F. (2010) ‘Human-computer interaction and the older adult: an example using user research and personas’, in *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments - PETRA ’10*. *the 3rd International Conference*, Samos, Greece: ACM Press, p. 1. Available at: https://doi.org/10.1145/1839294.1839353.

Pinheiro, C. and da Silva, F.M. (2012) ‘Colour, vision and ergonomics’, *Work*, 41, pp. 5590–5593. Available at: https://doi.org/10.3233/WOR-2012-0891-5590.

‘React’ (2022). Meta Platforms. Available at: https://reactjs.org/ (Accessed: 11 September 2022).

Schwaber, K. and Sutherland, J. (2020) *Scrum Guide*. Available at: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf (Accessed: 5

September 2022).
Sharma, A. *et al.* (2022) *Lucille: A To-Do App for the Elderly*. Agile Software Projects mid-term submission. Goldsmiths, University of London.
Sharp, H., Rogers, Y. and Preece, J. (2019) *Interaction Design: Beyond Human-Computer Interaction*. 5th edition. Indianapolis, IN: Wiley.
'Slack' (2022). Slack Technologies, LLC. Available at: https://slack.com/ (Accessed: 11 September 2022).
Snyk | Developer security | Develop fast. Stay secure. (2020). Available at: https://snyk.io/, https://snyk.io/ (Accessed: 4 September 2022).
The State of JavaScript 2019: Front End Frameworks (no date). Available at: https://2019.stateofjs.com/front-end-frameworks/ (Accessed: 3 September 2022).
Travis, D. (2016) 'The 1-page usability test plan', *Medium*, 15 January. Available at: https://medium.com/@userfocus/the-1-page-usability-test-plan-dbc8c3d7fb54 (Accessed: 9 September 2022).
Voytko, J. (2020) 'Why are we so bad at software engineering?', *www.bitlog.com*, 12 February. Available at: https://www.bitlog.com/2020/02/12/why-are-we-so-bad-at-software-engineering/ (Accessed: 12 August 2022).
Wells, D. (2009) *Dedicated Open Work Space*, *Extreme Programming*. Available at: http://www.extremeprogramming.org/rules/space.html (Accessed: 12 September 2022).
Williams, D. *et al.* (2013) 'Considerations in Designing Human-Computer Interfaces for Elderly People', in *2013 13th International Conference on Quality Software*. *2013 13th International Conference on Quality Software*, pp. 372–377. Available at: https://doi.org/10.1109/QSIC.2013.36.
'Zoom' (2022). Zoom Video Communications, Inc. Available at: https://zoom.us/ (Accessed: 11 September 2022).