Creating an App for Free Learning and Memorisation

Space Lutterodt-Clottey

Contents

Introduction

This report reflects the process of creating an app for the process of free learning and memorisation.

Free learning involves learning whatever you find most interesting at any specific moment, as well as not reading past the point that you begin to find a text uninteresting. This is opposed to learning things based on a curriculum.

The report reflects the process of creating a program from scratch when you already know the features, as the desired features were already largely present in the desktop program *SuperMemo.* However, I was unable to use SuperMemo while on the go as it does not have a corresponding mobile application.

Because I couldn't use SuperMemo on my phone, the reading I was doing on it felt transient and unproductive, therefore this report reflects the process of making a web app that can be used on a mobile phone that allows the user to read, remember and retain any article they wish to read while they're using their phone.

The report begins by analysing the theory behind SuperMemo, including what *spaced repetition* is and how it is conducive to learning. It then discusses a more niche implementation of the features of spaced repetition, *incremental reading*, which SuperMemo is built around. The report then looks at the pre-existing applications that complete similar but incomplete functions related to the full desired feature-set. The report then goes over the more technical side of app development, highlighting the choices made in the program architecture.

I then discuss the precise features required to create a minimally viable product and go over the process of creating two iterations of the program.

Finally, I discuss and evaluate the overall process of creation.

For the sake of simplicity, I will be referring to my program by the name *ReadAgain.*

Research Review / Background Information

This research review first covers the concepts of spaced repetition, active recall and incremental reading, three concepts that are central to learning. It then analyses the benefits and costs of three currently available programmes for this purpose.

Benefits of Spaced Repetition and Active Recall

Spaced repetition is the process of going over material repeatedly over time, spacing out reviews of information already covered. Research has shown that learning done in this manner is far more effective and the knowledge gained is far more stable in the subject's memory compared to when learning is done all at once (Dempster, 1989).

Spaced repetition was first tested in the 1970s (Oren, 2014) and first computerised in the 1980s (Wozniak, 2018), however, though it is growing in popularity it maintains being a rather niche concept *(Figure 1)*, despite its immense effectiveness.

Passive recall is when the subject is shown the information without any active effort on their part. For example, when a student studies by rereading a section in a textbook. *Active recall* however is when the subject is made to actively remember the information being learned, for example, a student spending their revision session answering flashcards they made previously. Research has also found that active recall is far more effective than passive recall in keeping information in long-term memory (Mark A. McDaniel, 2009).

Information learned via active recall is far more stable in long-term memory (Blunt, 2014) in part because it mimics the way the information will realistically be called upon when it is needed in a real world scenario.

Therefore, to create an application that brings maximum learning efficiency for the users it would be desirable to include spaced repetition and active recall as the learning mechanisms.

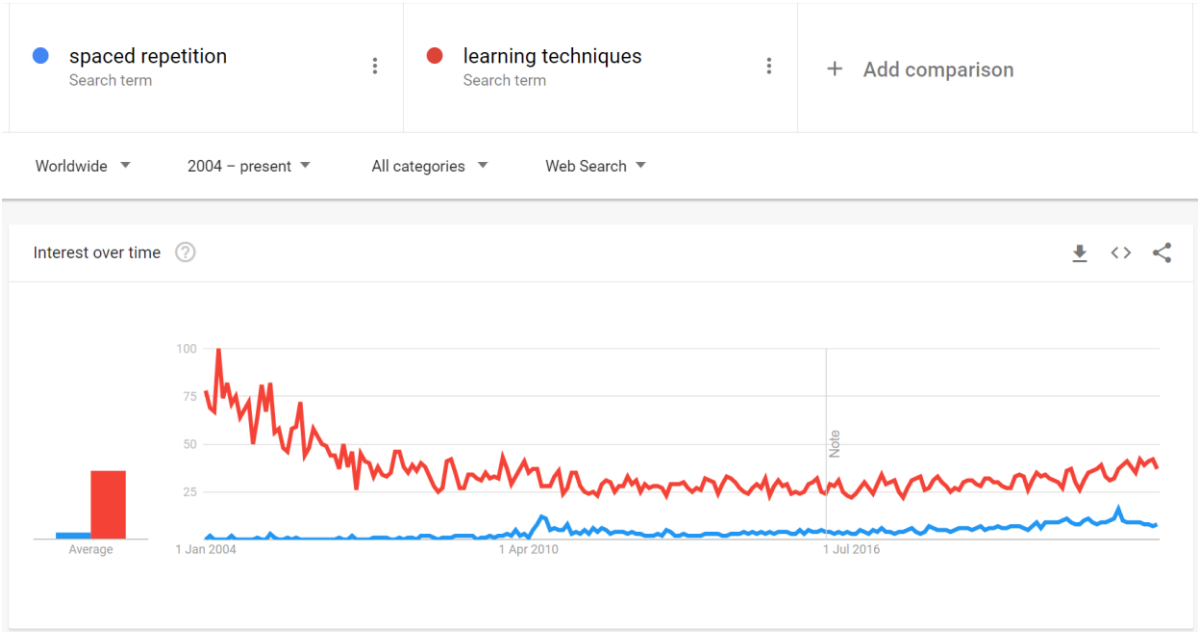

Figure 1 - Google Trends data on searches of "spaced repetition" vs "learning techniques"

Incremental Reading

Incremental reading is a method of knowledge acquisition in which many sources (such as articles) are read in parallel, with the reader able to make extracts of the reading material as they are reading, which will show up themselves later. In a good incremental reading software, the user is also able to set priorities for the materials they are reading, and it is the program's responsibility to show the reader the highest priority materials they have said they wish to read. (Wozniak, n.d)

Incremental reading is highly efficient and yet scarcely adopted, and even more scarcely implemented in software, with really the only fully-fledged implementation of it being in the *SuperMemo* software developed by Piotr Wozniak. However, as incremental reading is merely based on the collection of principles stated above, it is possible to implement in other programs.

To begin incremental reading, the user adds the materials they wish to learn from into the program (for example, the Wikipedia page on *Universal Basic Income, World War II, The Internet,* and as many others as the user wants, not confined to *Wikipedia*). The user adds a priority to each of the sources they import, depending on their importance.

The program will show the source back to the user, who will then read it to the extent that they want to. As they are reading, the user highlights any passages of text they find interesting. They then press a button, which turns the highlighted portion into an *extract*, meaning that this passage they have highlighted is now its own source, which will show up on its own at some point.

If the user finds something they particularly wish to remember, they can highlight that passage and make a *cloze deletion*, which creates a flashcard that the user will be asked to recall. (Wozniak, n.d)

At any point, users can dismiss any flashcards or sources that they no longer wish to come up in reviews, and the source will still be locatable by searching through the entire collection.

Once the user finishes reading a particular source, they can move on to the next source, and the program will schedule the article they just read to be shown at a point in the future that is calculated by a spaced repetition algorithm. (Wozniak, n.d)

This makes reading, learning and memorisation highly convenient as they are all combined into one unified workflow and program. Additionally, this method of learning also means that the user can maintain interest and focus on a learning session for longer than they might do while reading a book, as they can move on to other topics or sources whenever they incur boredom over their current source.

Incremental reading also integrates the method of interleaving, meaning different topics can be learned at the same time, of which there are studies which suggest it improves retention (Dempster, 1989). Anecdotally, I and others have found it to maintain interest longer (SuperMemo Community Wiki, 2021).

Additionally, SuperMemo combines passive review (reading sources) with active recall (answering flashcards), appropriately implementing the earlier discusses literature on efficient learning.

Unfortunately, due to its small userbase and even smaller product space, there is currently extremely limited research in the area, the overwhelming majority being conducted by Piotr Wozniak, the creator of SuperMemo. The limitations of this scenario are discussed further in the annotated bibliography.

Competitor Analysis

There are multiple products that achieve similar goals to the one intended by my project; however, each misses a crucial piece of functionality. Each of these programs is reviewed below.

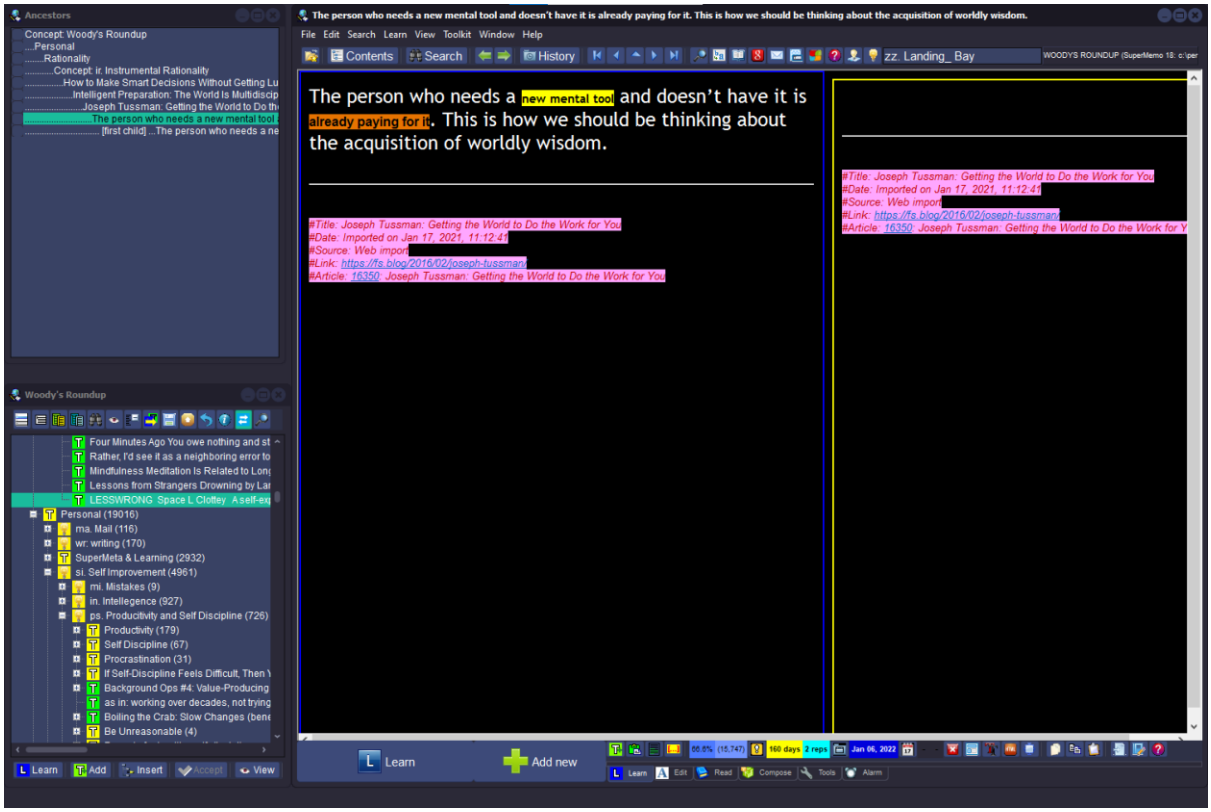

Figure 2 - SuperMemo in use (screenshot by me)

SuperMemo

The first is *SuperMemo,* a desktop application for windows created by Piotr Wozniak (Wozniak, SuperMemo.com, 1990). SuperMemo is incredibly feature-dense and has been in development for nearly three decades, however this comes at the expense of having a terrifically cluttered UI (*Figure 2)* that is excessively difficult to get used to. For example, as an experienced user for a year and a half, I still use no more than 20% of the buttons on the interface.

Additionally, it is not cross-platform, which is very limiting to users of the Mac and Linux operating systems, who are forced to either use parallels to simulate a windows machine on their own (which are often inconvenient, buggy and expensive), or buy a windows machine just to use SuperMemo (Incogito, 2021). This of course means that you are unable to go through your articles and flashcards while on your phone, which is part of my main motivation to create a phone-friendly program that achieves the same function.

In summary, SuperMemo has two main benefits of usage. The first being that it is one of the only fully-fledged pieces of software for incremental reading in existence, and the second being that it allows users to import many kinds of materials to learn from, including their own writings and notes.

However, as SuperMemo itself is built on top of the horrendously archaic web browser Internet Explorer (Wozniak, Less dependent on Internet Explorer, 2020), users are forced to either use copy-

pasting to enter their learning materials or to use Internet Explorer, the inconvenience of which likely filters out many potential users who would gain utility from the program.

Instapaper

Instapaper allows the user to import articles from any device to be read on any device. It is a cross-platform read-it-later application (Instant Paper, 2022).

It is far easier to import content into Instapaper compared to SuperMemo, as Instapaper has a bookmarklet option (a small button on your bookmark bar you can press on any page that imports to your collection) as well as a chrome extension.

This is an extreme improvement over ease of imports compared to SuperMemo.

Additionally, Instapaper's mobile function is highly appealing and simple to use, just like its web version. I would like to take Instapaper's design as an inspiration for my own project.

I currently use Instapaper for whenever I wish to read articles on my phone.

The downsides of Instapaper are that it does not have an incremental reading feature, nor does it have flashcard functionality (Instant Paper, 2022), meaning that it is not useful for long-term learning while on the go.

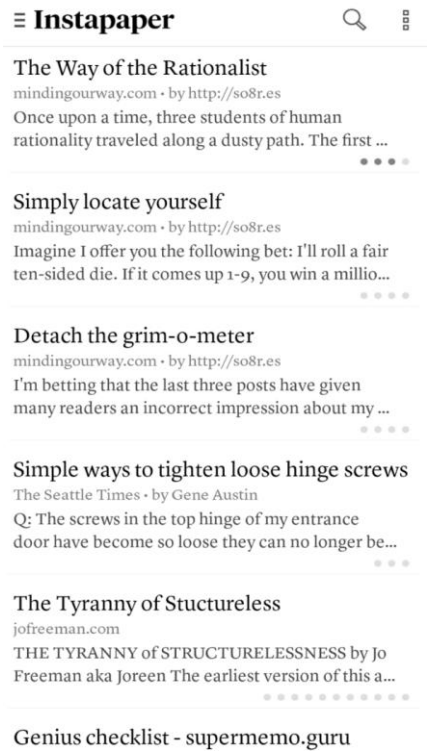

Figure 3 - Instapaper in use (screenshot by me)

Anki

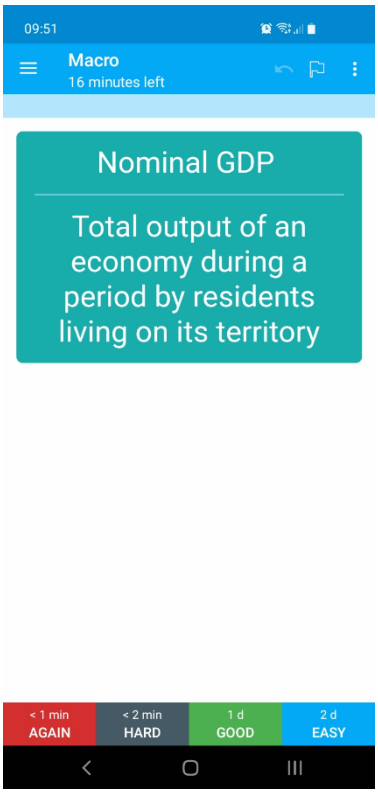

Figure 4 - Anki in use (screenshot by me)

Anki is a cross-platform spaced repetition flashcard program. It excels in card scheduling and spaced repetition, and as it is an open-source application, anyone can see its code and improve it. It is also free (Anki, Inc, 2022), which is highly appealing.

Anki's biggest flaw however is the inconvenience of creating new flashcards, as it lacks any in-built incremental reading feature (Rice, 2020). There is an incremental reading extension for the Anki desktop program, however that has been described as clunky and unpleasant to use (Anki Community, n.d.). Additionally, this does not allow for incremental reading while using your phone, which is one of my major goals.

While the lack of incremental reading does not inhibit using the program for things such as memorising formulas for mathematics, it makes it highly inconvenient for memorising lessons from books or articles, as the user cannot put the learning materials into Anki, meaning it requires both the learning recourse and Anki to be open at the same time, as opposed to just one unified program (like SuperMemo).

Additionally, Anki lacks prioritisation of material, meaning that when you have large amounts of flashcards you cannot be certain that you are going over the most important aspects first.

A visual comparison of the features available in each of these three programs is available below (*Table 1*).

Table 1 - Table of how features differ between different learning software

	Description	Read articles	Incremental reading	Flashcards	Usage on phone
SuperMemo	Desktop-only flashcards + reading	Yes	Yes	Yes	No
Anki	Cross-platform flashcards	Yes	No	Yes	Yes
Instapaper	Cross-platform reading	Yes	No	No	Yes

App Development

Backend

In creating a web application, it is necessary to include both a backend and a frontend. The frontend is the aspect of the application that is rendered to the user and that the user interacts with. It is important that this side of the application is simple and appealing to use and navigate. The backend of an application is the storage of the data needed for the application to work, for example usernames and passwords, history of articles added, videos watched, etc. (Wales, 2021).

It is important that sensitive data is separated from the main section of the application otherwise malicious users would be able to access the information of other users from the main section of the app, therefore the front end does not have all the data, but merely sends *requests* to external servers whenever data is required (Schwarzmüller, 2020).

For this program, I expect I will only really need the backend to create a database, which I have done before in previous projects I have created; therefore, it is not required for me to learn further backend strategies for this project.

Because the user needs to be able to access their data at any time, it is necessary to hire servers from other companies. Running a personal server would be too expensive and fragile, as whenever the host computer went down, users would be unable to access the site (Aravindhan, 2022).

Fortunately, many companies offer free server space such as Heroku. I also have previous experience with Heroku, making it an appropriate technology for this project.

Firebase by Google may also be an appropriate technology to use for this product. This is because it takes a comprehensive approach to backend management, integrating both authentication (ensuring that the user really is who they claim to be) as well as storage. Additionally, it is cross-platform and cross-framework (Java T Point, n.d.). However, having never used Firebase previously, it would be costly to learn how to use it for this project, therefore I will use Heroku for the database.

Frontend

The two major platforms for native app development are Android and IOS. Writing separate code for each of the platforms would be time consuming and unnecessarily difficult.

Another method for writing cross platform code which creates applications *native* to their platform (i.e., actual apps) is to use codebases that are cross platform, meaning you only write code in one language which automatically translates itself for both Android and IOS.

Currently, there are two major competitors for coding a native, cross platform application: *React Native* by Facebook and *Flutter* by Google. (Statista, 2021)

React is a framework for web development where the user writes HTML (a markup language, simply for describing *what* is on a webpage) and JavaScript (a programming language, for describing what the page *does* and *how* it functions) in the same code. (Krill, 2014)

React Native is based off React, however it allows the user to create cross-platform applications (What is React Native?, n.d.). As I am already familiar with React, React Native would be vastly more appropriate for me to use as opposed to learning a new programming language with little comparative advantage in this scenario.

Another method for writing only one set of code that can be run on both platforms is by creating a website that can be run in the browser and instructing the user to manually download it as an application. These are called “Progressive Web Apps” (Tran, 2022).

As I have already programmed websites before, this has the benefits of minimising the amount of new framework and code I will have to learn. Additionally, this method removes the need to learn how to distribute applications and/or put them on the App Store or the Google Play Store. As this method is far more time efficient, I have decided to create a website that can be run on mobile instead of a mobile-first app.

In this paper, I also use the terms client side and server side to refer to the frontend and backend respectively.

Development

During software development, it is important to focus on creating something that is valuable to users, which is called having a product-market fit. This is important as it is often too easy to assume that your project is of general interest but find out that it actually isn’t only after you have already dedicated a large amount of time working on it (Kromer, n.d.).

There are two main ways of ensuring you have a product-market fit: creating a “minimal viable product”, and “eating your own dog food”, which are discussed below.

An aspect that is integral to software development is the order of development. It is necessary to develop features in a logical progression in which the most necessary aspects are implemented first, ignoring important yet superfluous features along the way, but returning to add them once the more important aspects have been added. If the programmer is unable to finish their project, this will leave them with a barebones application however it will ensure that it is useable. This is called a “minimal viable product” (Becker, 2020).

This is important for my project as I will first be creating a minimally viable product, such that if there are unexpected delays or difficulties, it will still at minimum result in something that is functional, and that can provide utility to users. It also means that if I later decide to commercialise the website, then I can very cheaply test product-market fit. If few people want the minimum viable product, then it means there is no product-market fit (Kromer, n.d.) and I should move on to a different idea because it would not actually be that useful for many people.

Another way of ensuring product-market fit is to have yourself be the intended user, a euphemism called “eating your own dog food” in software development (ZD Net, 2003). Thus, I intend to use my own program for reading and learning while on my phone, to make sure I am creating something that is specifically useful to the target audience that includes me, and not merely an imagined populous.

Design Brief and Specification

It is very satisfying reading articles on SuperMemo because there is the assurance that anything I read, make extracts or flashcards out of will be shown to me again in the future, providing an assurance that it isn't transient, meaning that time spent reading in SuperMemo felt productive and meaningful.

It was upsetting to me that I could not replicate this on my phone, as I could not have any assurance that I would come across again any articles or interesting ideas from books I was reading on my phone, and SuperMemo does not have a mobile application. So, what I want to achieve with ReadAgain is to have a reading experience on my phone that feels productive, and to be able to know that I will see what I am reading again in the future.

The following page contains the full specification table (*Table 2*)

Specification

Below I explore the specification requirements for the program (*Table 2*).

Table 2 - Feature specification table for ReadAgain

Requirement	**Justification**
The user can enter a link to an article they wish to read, which is added for them into the database	The alternative to getting the articles for the reader based on a link they copy is for them to copy the entire article, which is more cumbersome and annoying. Therefore, this is for ease of user experience.
The article is saved into the database	This is so that the users articles can still be saved if the user closes the website, and so that it can be accessed from multiple devices.
Pressing the "**Next**" button causes the next element to appear	The user has a way to navigate through the sources they have imported.
When a new section of text is selected and the user presses "**Extract**", a new topic is created with the highlighted text	Users are able to narrow down their sources over time, keeping only the most vital pieces of information to be seen by their future selves.
When a section of text is highlighted and "**Create Flashcard**" is pressed, a new flashcard is created, using the selected text as the answer field.	Users can quickly switch from memorising content with passive recall to using the far more stable method of active recall.
When a new extract or flashcard is created, it is added into the queue	So that the user does not have to manually seek out the flashcards and extracts they have created, and only has to remember to press one button, keeping the process simple.
When a flashcard appears in the queue, the answer section is hidden until the user presses "**Show answer**"	If the user can see the answer before they try to recall it themselves, they won't be given an opportunity to actually recall it themselves, defeating the point of a flashcard.
Flashcards can be rated based on how easy they are to remember	So that the card can be scheduled such that harder-to-remember cards come up sooner.
The program works on a mobile phone	Users can read articles, learn, and recall flashcards while on the go, making use of otherwise dead time such as waiting in queues.
The interface is uncluttered and clear	It is not complicated for new users to get a hang of, decreasing the barrier to entry. Additionally, so that it is not annoying and aesthetically displeasing for even experienced users.
The program feels swift to use	If the program is slow and has a lot of friction, the user may get bored and switch to other applications on their phone, decreasing the amount that they spend learning. Additionally, to just have a more enjoyable experience of using the application.
The program gives visual cues of key functions (especially when an article is saved)	This allows the users peace of mind that their data is actually being stored so that they can actually put things into it and trust that they don't have to remember it.

Iteration 1

This section explains the development of the first iteration, exploring the challenges of implementing specific features, and the techniques that were used to overcome them.

First design

Inputting article URLs

It was simple to create a text input box that can receive article URLs. However, I was not able to fetch the text from the article for my own use from the client side so I had to create API call (Application Programming Interface — a method of communicating between computers and programs, [Mulesoft, n.d.]) to a remote server and fetch the article text from there.

Below is the code that runs on the server side, which fetches the requested article text. I have commented the code to describe what the lines beneath each of the comments do (*Figure 5*).

```
5   export default async function fetchText(req, res){
6       // Receive the link from the frontend
7       const link = JSON.parse(req.body).link
8
9       // fetch the text from the link
10      const hello = await fetch(link)
11      let text = await hello.text()
12
13  }
```

Figure 5 - fetchText() function, fetches text from an article, given a link

The link is passed into this code from the client side, stored in the variable “link”. Line 8 shows the text being fetched, and lines 11-15 show the text being added to a record including its user, its URL and its text, which is ready to be put into the database.

I experienced a lot of difficulty with correctly running the fetch request and had to look up documentation for the fetch command in order to use it correctly.

Eventually that worked, and the next issue was with passing the fetched data back to the client side. However, I found out it was possible to save the text straight to the database and then load it up on the client side by doing an entirely separate call to the database. I decided to do this as I already knew how to fetch from the client side to the database, and it seemed impossible to pass the data straight from the server side to the client side (even though it initially seemed very simple).

Queue of topics

I made it so that each element has a variable called *interval* inside the database. The interval of an element is incremented whenever "Next" is pressed while you are on it, which is supposed to represent once you have finished reading it. So, the program sorts what should by shown by ranking all the elements in the database by *interval* in ascending order.

I decided to use an "interval" counter to decide when it was seen again as that is how Anki implements it (*Figure 6*).

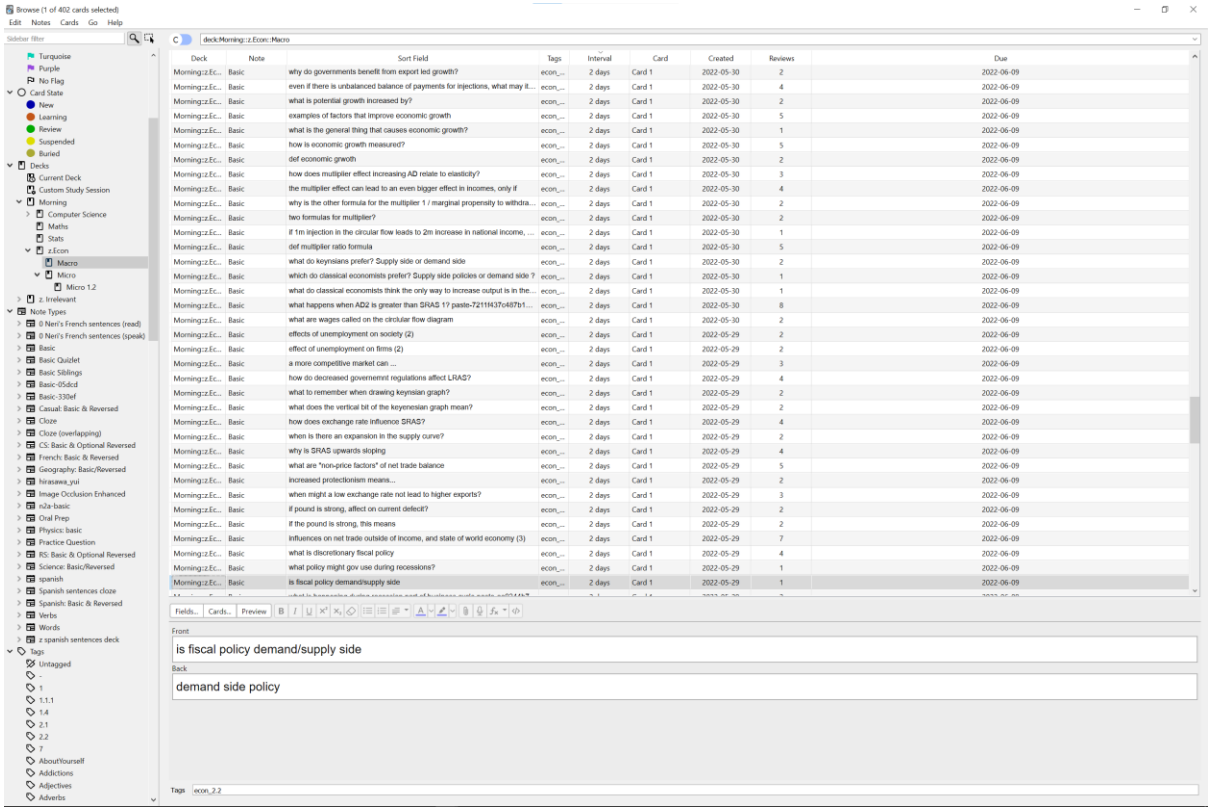

Figure 6 - Screenshot of the browser in Anki (by me)

Extracts

I experienced many difficulties while implementing the ability to create extracts.

The first was in figuring out a way to know what text was actually being highlighted while the button was pressed. I had to do a lot of research to figure out how to do this but ended up learning that this was possible with *window.getSelection()* command (MDN Web Docs, n.d.), which made it relatively simple to get it working.

After that, adding the text that was highlighted into the queue was simple enough, as all I had to do was add it to the pre-existing database, which was possible using the same API call that I used to be able to add fetched article texts into a topic in the database.

This was all completed with the "makeExtract()" function, which is shown below. I have commented the code to describe what the lines beneath each of the comments do (Figure 7).

```
12  const makeExtract = async () => {
13    // Get the selected text
14    let selectedText = window.getSelection().toString().replace(/(?:\r\n|\r|\n)/g, '<br />')
15
16    // Add the selected text to a record that is ready to be added to the database
17    let data = {
18        html: selectedText,
19        url: currentElement.link
20    }
21
22    // Pass the created record into an API call that communicates to the backend, where it is added to the database
23    const response = await fetch("/api/createElement", {
24      method: "POST",
25      body: JSON.stringify(data),
26    });
27
28    console.log(response)
29  }
```

Figure 7 - makeExtract() function, allows extracts to be created and makes a request to add them to the database

Screenshots

Below are screenshots of the first iteration.

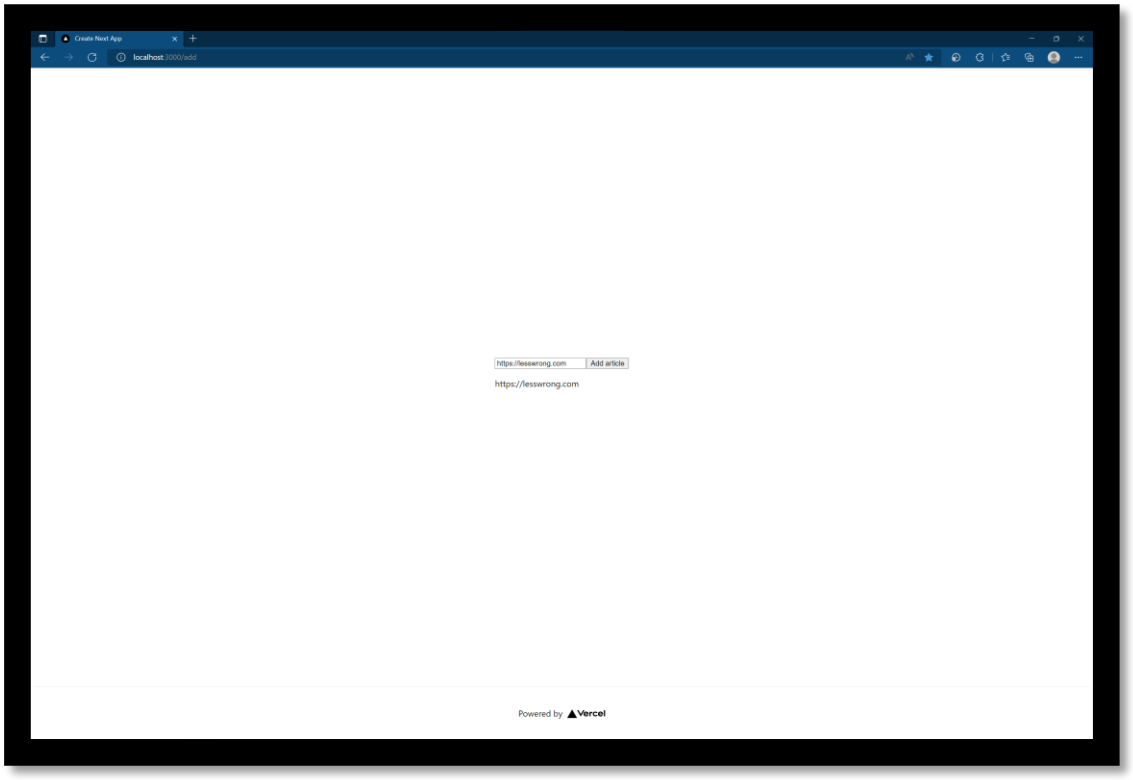

Figure 8 - the page for adding links to new articles

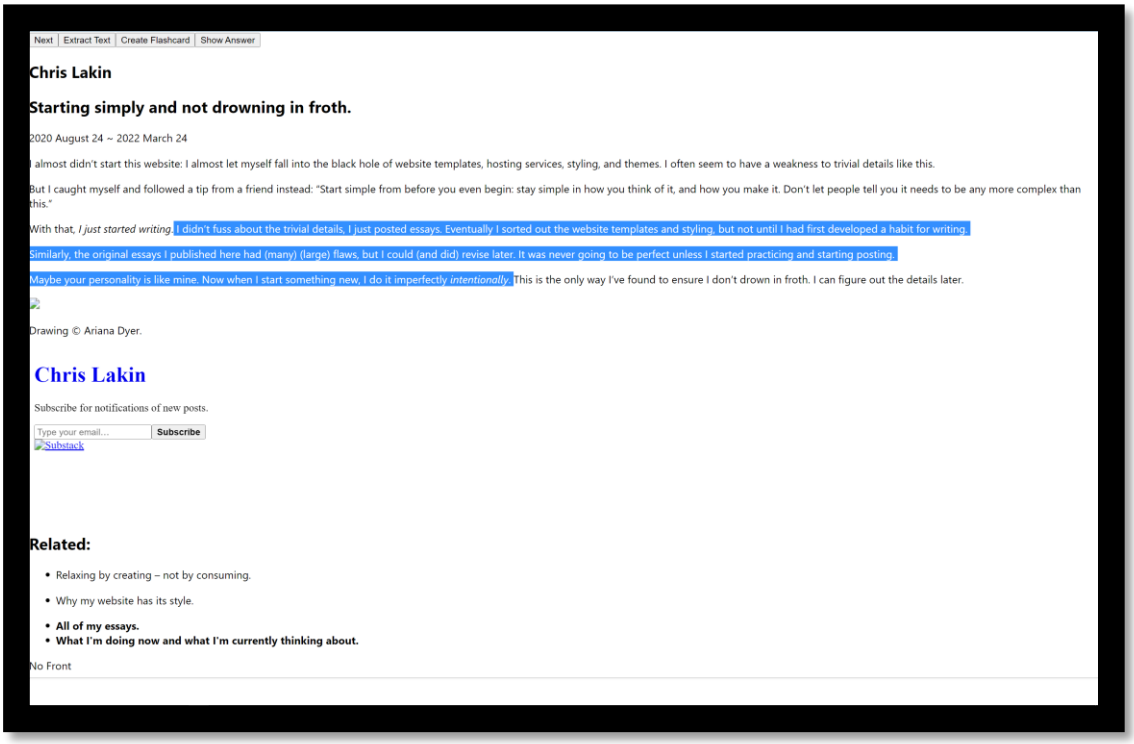

Figure 9 - an extract being made while reading an article

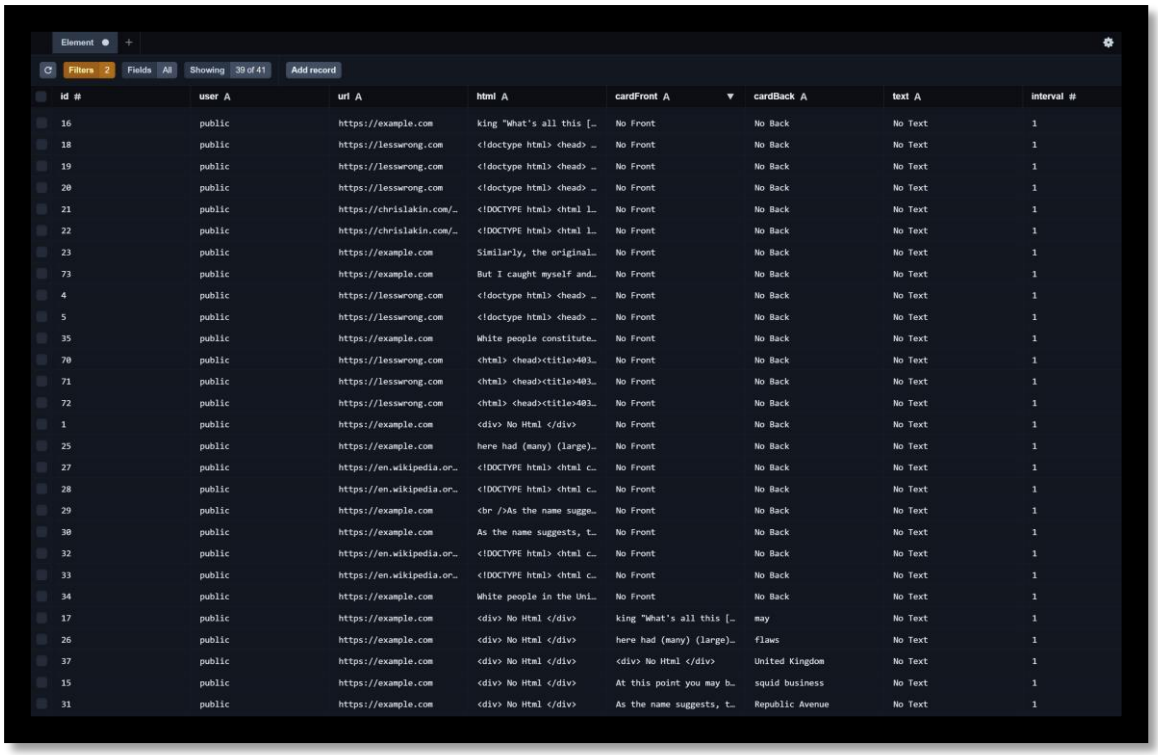

Figure 10 - the database (this viewer was not made by me; it is a part of the backend system I use)

Testing against spec

I managed to hit most of the most important specifications for the project, including the user being able to enter the URL of an article, the next button being able to show the next element and adding new topics that have been created into the queue.

This means that at this stage the application is now functional as a standalone read-it-later app, but with the added functionality that the user can create extracts and those extracts will show up on their own.

I was not able to add the functionality of the flashcards due to time constraints and unexpected difficulties, particularly during the section where I had to track which text was currently highlighted, but I intend to add the missing features in the next iteration.

Specification

Table 3 - Review of specification features after the first iteration

Requirement	Progress	Notes
The user can enter a link to an article which is saved	Functional	This was rather simple, as it mostly just required creating a textbox.
The article is saved into the database	Functional	I struggled for months with trying to pass the retrieved article straight to the frontend. In the end, I solved this by passing the article to the database (still in the backend) then retrieving the article on the frontend side *from* the database instead of passing it over directly
Pressing the "**Next**" button causes the next element to appear	Functional	Currently implemented just by incrementing the ID of the element from the database that is being shown
When a new section of text is selected and the user presses "**Extract**", a new topic is created with the highlighted text	Functional	Surprisingly hard to keep track of which text was currently highlighted, but ended up just using a "check what's highlighted" function
When a section of text is highlighted and "**Create Flashcard**" is pressed, a new flashcard is created, using the selected text as the answer field.	Not added	
When a new topic or flashcard is created, it is added into the queue	Not added	
When a flashcard appears in the queue, the answer section is	Not added	

hidden until the user presses "**Show answer**"		
Flashcards can be rated based on how easy they are to remember	Not added	
When a flashcard is rated, the time it takes for it to show up again in the queue is set	Not added	
The program works on a mobile phone	Not added	
The interface is uncluttered and clear	Not added	
The program gives visual cues that an article has been saved	Not added	
The program feels swift to use	Not added	

The key improvements that need to be seen are the implementation of the flashcard functionality and improvements to the user interface.

Iteration 2

This section explains the development of the second iteration, primarily exploring the process of implementing the flashcard functionality.

Second design

Flashcard creation

It was relatively simple to create a flashcard, as I simply created a new element that had a *question* and *answer* category and used the property for the text of the current topic as the *question* of the card with the highlighted section as the *answer* of the card (*Figure 11*).

But I had to decide between creating two types of elements in the database (one for flashcards and one for articles), or between having all of the elements in the database have fields for both article text and flashcard backs and fronts, then just leave the fields that are not needed empty.

I decided to unify all the elements this way (*Figure 10*), as I thought it would make the database management simpler. At the time, I wasn't sure if it would actually make the database simpler but looking back now, I believe it did.

```
const makeFlashcard = async () => {
  // Gets the selected text
  let selectedText = window.getSelection().toString().replace(/(?:\r\n|\r|\n)/g, '<br />')

  // Creates a version of the selected text where the answer is hidden to be used as the
  // question side of the flashcard
  let clozedHTML = currentElement.html.replace(selectedText, '[...]')

  // Creates a record that is ready to be added to the database
  let data = {
      url: currentElement.link,
      cardFront: clozedHTML,
      cardBack: selectedText,
  }

  // Passes the created record into an API call that communicates to the backend, where it is added to the database
  const response = await fetch("/api/createElement", {
    method: "POST",
    body: JSON.stringify(data),
  });

  console.log(response)
}
```

Figure 11 – the makeFlashcard() function, which turns the selected text into a flashcard

Flashcard rendering

At this point, it was necessary to create a different type of page, where if the next element to be loaded was a flashcard, it would not just load the body text and have the “Create Extract” button there.

I originally thought that it was important that the “Create Extract” button should not have been visible while a flashcard is being rendered, as you can’t make extracts from flashcards in SuperMemo. However, I didn’t think this was an essential feature to implement relative to more important features, so I continued on to implementing the flashcard rendering.

I created a conditional section of the code (an “if” statement) that would check whether the upcoming element was a flashcard, and if it was, then only render the question, with a “Show Answer” button instead of the answer. When selected, the “Show Answer” button would unhide the answer and show a “Next” button instead.

```
➢          {showAnswer && <div dangerouslySetInnerHTML={{__html: currentElement.cardBack}} />}
```

This is the code that decides whether the back of the flashcard is shown. Everything enclosed by the “ ” tag is the back of the card, and “&&” means that the div is only shown if the variable before it is true. Therefore, all I have to do is alter the value of “showAnswer” to directly control whether the answer is shown.

This was a rudimentary implementation of active recall. A large focus of well-made active recall systems is the variation in timings of seeing the card again, depending on how well the user rates themselves to have remembered the card (showing cards sooner the harder they were to remember) (Shaughnessy, 1977). As there is no rating system and the elements in the queue are being shown linearly, this likely means that the program at this stage does not have enough functionality to generate the benefits of spaced repetition discussed in the literature (Dempster, 1989).

Screenshots

Below are screenshots of the second iteration.

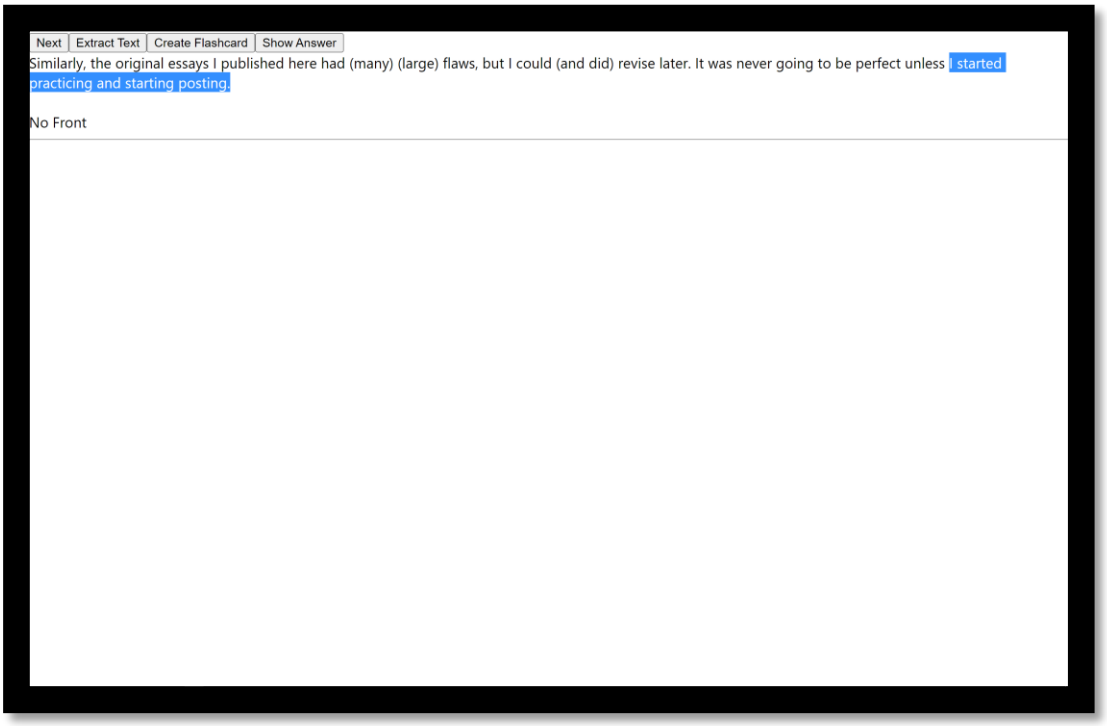

Figure 12 – An extract being read, where text is being highlighted in order to create a flashcard

Figure 13 - A flashcard being rendered before the answer is shown

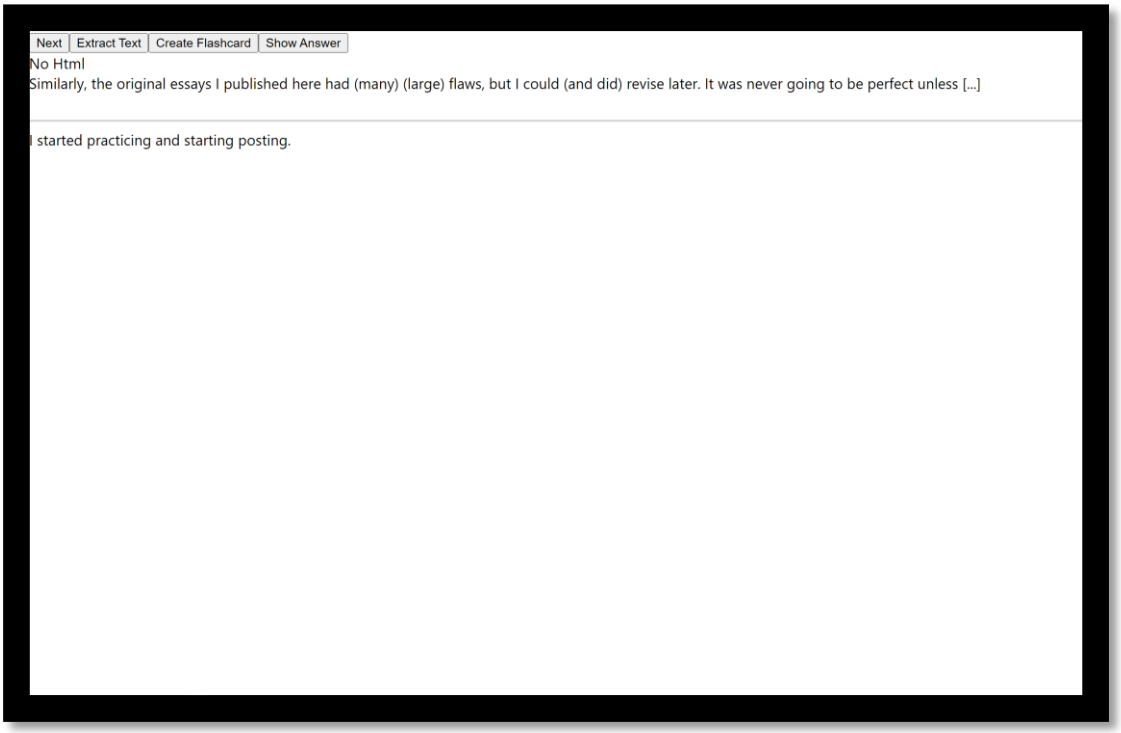

Figure 14 - A flashcard being rendered after the answer is shown

Testing against spec

Specification

Table 4 - Review of specification features after the second iteration

Requirement	Progress	Notes
The user can enter a link to an article which is saved	Functional	
The article is saved into the database	Functional	
Pressing the "**Next**" button causes the next element to appear	Functional	
When a new section of text is selected and the user presses "**Extract**", a new topic is created with the highlighted text	Functional	
When a section of text is highlighted and "**Create Flashcard**" is pressed, a new flashcard is created, using the selected text as the answer field.	Functional	

When a new topic or flashcard is created, it is added into the queue	Functional	
When a flashcard appears in the queue, the answer section is hidden until the user presses "**Show answer**"	Functional	
Flashcards can be rated based on how easy they are to remember	N	One aspect that I did not do was "rating" of the flashcard. As other sections took up so much time, I could not fit in the time to create many new buttons to provide different rating mechanisms, however it would be relatively simple to add just a thumbs up or thumbs down button, and schedule it to zero days in the future in the event that it's a thumbs down, and to two or three days into the future if a thumbs up is pressed. The absence of this feature likely decreases the quality of the user experience as their cards are shown to them indiscriminately, decreasing the benefits that would have come from using real spaced repetition.
When a flashcard is rated, the time it takes for it to show up again in the queue is set	N	I did not implement the ability to rate flashcards, which is required for this feature.
The program works on a mobile phone	Y	In theory, there is nothing stopping this program from working on a mobile phone, however I need to publish the app to the web in order to test this.
The interface is uncluttered and clear	N/A	The interface is not cluttered, as there are literally zero excess buttons. However, it does not look remotely appealing, and I have not added any spacing between elements on the page, making it look rather unappealing.
The program gives visual cues that an article has been saved	N	There are absolutely zero visual cues in the current iteration, however errors are thrown when an article is not successfully imported which should be sufficient.

The program feels swift to use	N/A	The highlighting functions are swift, however the act of moving to the next element — arguably the most important functionality to feel as instantaneous as possible — certainly has a small delay which is not pleasant. However, it is wholly unclear to me how to go about decreasing this error. Potentially through fetching the text in the next article *prior* to "next" being pressed, however I do not currently know how to program this.

Overall, I have managed to complete the most integral points surrounding functionality, in the original application. There are certainly further improvements I would like to make, however the product I have created is a functional minimal viable product.

However, I have failed with my goal to increase the quality of my mobile reading. Though *ReadAgain* has incremental reading and flashcard capabilities and Instapaper does not, my app is lacking many features that would make it pleasant enough to actually use over Instapaper.

Future Development

There are some overall quality of life features that, if given more time, would have made the program holistically nicer to use. For example, a method to export one's learning, preferably to SuperMemo or Anki (or even other note-taking applications such as Obsidian or Roam Research) would prevent user lock-in and provide users with peace of mind while using the program. This would be because they would know that access to their data was not dependent on me, the creator, deciding to keep the program online.

Additionally, a greater number of visual cues or animations in the app would give it a more concentrated feeling of progress, as though one is truly progressing between materials they wish to learn.

Doubtlessly there is utility in having a functional companion web app suited for the desktop application, however that is time-consuming and not where the comparative advantage of the product lies, and not the niche it particularly aims to fill.

Conclusion

My objective for this project was to create a way to learn productively while using my phone. I believe this criterion is mostly fulfilled, as I am now able to read over articles I thought worth reading while on the go and create flashcards of the sections that I deem worth remembering.

Unfortunately, it is not a native app, which makes it slightly less enjoyable to use, though this is not central to the functionality of the project.

In conclusion, my project to create an application for free learning and memorisation was successful. Though I hit many roadblocks, my aims to create an app that allows users to enter articles they would like to read, show them those articles again over time, and to create flashcards of the material they were reading, were met in a way I believe to be in accordance with my original goals.

Evaluation

Aims

In creating this project, my aims were to create a functional program to increase the quality and productivity of my experience of reading on my phone. Additionally, I intended to maintain and improve my web development and project creation skills, as at the time of starting this project I intended to be a web developer.

Though there is an incredibly large amount for me to grow on the path of being a web developer, I now picture my future with greater uncertainty into what career path I will go into. I hope to explore in more careers and settle into one that was far more alien and unpredictable than, looking back, the one I would have been able to say I was comfortable with as a teenagers.

Have I met my aims

I have somewhat achieved the aim of creating a functional program. While it works with barebone functionality on my local development environment on my desktop, I am yet to export it to the internet to test on my mobile phone. Though once I do export it, it should be functional.

However, I have failed with my goal to increase the quality of my mobile reading. Though *ReadAgain* has incremental reading and flashcard capabilities and Instapaper does not, my app is lacking many features that would make it pleasant enough to actually use. For example, outside of lacking aesthetics, it also lacks the ability to export data, has no animations or animations and visual cues, nor does it come with a pleasant-to-use web app companion. With all of these in total, it is the case that Instapaper is still the superior mobile-reading application, even in spite of not having incremental reading or flashcard functionality. Barring significant changes, I do not expect myself to be using ReadAgain for my reading on the go.

On the other hand, I have certainly met the goal of maintaining my web development ability. The process of creating ReadAgain required I reuse concepts that I had learned when I first learned web development, strengthening my knowledge of how to achieve certain goals when creating a website.

Another benefit regarding my web development skill is that creating the success criteria made the programming aspect follow a logical progression. I will be sticking to creating features in a logical progression for future projects.

When following a specific plan, I found that some aspects which I expected to be simple were indeed a lot harder and took up many more months than I originally thought, such as capturing which section the user was currently highlighting and turning that into an extract, as well as correctly fetching the article text from the sites themselves, as I was barred from the program from doing this on the client side, an obstacle I did not expect to encounter.

Additionally, I believe that working on ReadAgain has improved my web development confidence. I have long thought that being able to create a database structure that can manage extracts and articles within the same database and showing them in the queue was something that was well outside of my ability, though I found the part that I implemented to be surprisingly simple, increasing my confidence of what I am able to create.

However, I am not sure if I am able to completely claim it has improved my web development ability. Though I certainly did encounter a few new and interesting features (such as fetching text from articles and rendering flashcards), for the most part I only retreated programming ground that I have already covered, sticking to frameworks I was familiar with due to the scarcity of time, and not creating anything that was significantly beyond my pre-existing ability.

If I were to do this project again, I would insist to myself from the beginning on using and learning at least one new framework, such that completing the project would permanently increase my ability to create in the future. Additionally, I would focus more on polishing the application instead of just pure functionality, such that whatever I create is actually appealing to use.

When working on the project, I found (as is typical with programming) that steps I expected to take a few minutes routinely took days or weeks longer to resolve, as I encountered an unexpected bug that revealed an oversimplification in my understanding of the task at hand. If I were to do it again, I would extend the amount of time dedicated to each aspect of the project and in turn decrease the total amount of features I plan to implement.

Occasionally when stuck on something and returning to consider or plan out other features of the project, I would realise a simpler way to do it. If doing the project again, I would more consistently use writing about the bug I was receiving as a method of bug fixing itself, via stepping back from the problem and reviewing the constraints I believe I'm working under and considering alternative ways of getting around the problem.

Overall, I have greatly enjoyed working on this project, and look forward to seeing how what I have learned here will affect my future.

Bibliography

Anki, Inc. (2022). *Anki*. Retrieved from ankisrs.net: ankisrs.net

Aravindhan. (2022, 01 05). *Seek A Host*. Retrieved from https://www.seekahost.in/self-hosting-website/

Bae, C. L. (2018). Investigating the testing effect: Retrieval as a characteristic of effective study. *Leaning and Instruction*.

Becker, R. (2020, 08 14). *Minimal Viable Product*. Retrieved from Techopedia: https://www.techopedia.com/definition/27809/minimum-viable-product-mvp

Blunt, J. R. (2014). Learning with Retrieval Based Concept Mapping. *Journal of Educational Psychology*.

Community, A. (n.d.). *Incremental Reading*. Retrieved from Ankiweb: https://ankiweb.net/shared/info/935264945

Dempster, F. M. (1989, December). *Spacing effects and their implications for theory and practice*. Retrieved from Springer.com: https://link.springer.com/article/10.1007/BF01320097

Incogito, A. (2021, 06 07). *First Steps — SuperMemo*. Retrieved from supermemo.wiki: https://www.supermemo.wiki/en/supermemo/first-steps

Instant Paper, I. (2022). *Instapaper Premium*. Retrieved from instapaper.com: instapaper.com/premium

Java T Point. (n.d.). *Features of Firebase*. Retrieved from Java T Point: https://www.javatpoint.com/features-of-firebase

Krill, P. (2014, 05 15). *React: Making faster, smoother UIs for data-driven Web apps*. Retrieved from Infoworld: https://www.infoworld.com/article/2608181/react--making-faster--smoother-uis-for-data-driven-web-apps.html

Kromer, T. (n.d.). *False Positives and Product / Market Fit Survey – How Do We Test for It?* Retrieved from Kromatic: https://kromatic.com/blog/false-positives-and-product-market-fit/

Mark A. McDaniel, D. C. (2009, April 1). *The Read-Recite-Review Study Strategy: Effective and Portable*. Retrieved from Sagepub: https://journals.sagepub.com/doi/abs/10.1111/j.1467-9280.2009.02325.x

MDN Web Docs. (n.d.). *Window.getSelection()*. Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/API/Window/getSelection

Mulesoft. (n.d.). *What is an API?* Retrieved from Mulesoft: https://www.mulesoft.com/resources/api/what-is-an-api

Oren, S. (2014). Effects of Spaced Retrieval Training on Semantic Memory in Alzheimer's Disease: A Systematic Review. *Journal of Speech Language and Hearing Research*.

Rice, I. (2020). *Incremental reading in Anki*. Retrieved from wiki.issarice.com: https://wiki.issarice.com/wiki/Incremental_reading_in_Anki

Schwarzmüller, M. (2020, 07 02). *Frontend vs Backend*. Retrieved from ademind.com: https://academind.com/tutorials/frontend-vs-backend#:~:text=Frontend%20and%20backend%20communicate%20with,store%20it%20in%20some%20database.

Shaughnessy, J. J. (1977). Long-Term Retention and the Spacing Effect in Free-Recall and Frequency Judgments. *American Journal of Psychology*.

Statista. (2021, 09 16). *Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2021*. Retrieved from Statista: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/

SuperMemo Community Wiki. (2021). *Humans of SuperMemo*. Retrieved from SuperMemo.wiki: https://www.supermemo.wiki/en/supermemo/humans-of-supermemo

Tran, J. (2022, 01 12). *MagestorePWA vs Native App and how to choose between them*. Retrieved from Magestore: https://www.magestore.com/blog/pwa-vs-native-app-and-how-to-choose-between-them/#:~:text=PWAs%20are%20more%20secure%20than,worry%20they%20might%20be%20stolen.

Wales, M. (2021, July 1). *3 Web Dev Careers Decoded: Front-End vs Back-End vs Full Stack*. Retrieved from udacity.com: https://www.udacity.com/blog/2020/12/front-end-vs-back-end-vs-full-stack-web-developers.html

What is React Native? (n.d.). Retrieved from Oreilly: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html

Wozniak, P. (n.d.). Retrieved from SuperMemo Guru: https://supermemo.guru/wiki/History_of_spaced_repetition

Wozniak, P. (n.d.).

Wozniak, P. (1990). *SuperMemo.com*. Retrieved from 3.2. Application of a computer to improve the results obtained in working with the SuperMemo method: https://www.supermemo.com/en/archives1990-2015/english/ol/sm2

Wozniak, P. (2018, November 29). *Forgetting Curve*. Retrieved from SuperMemo Guru: https://supermemo.guru

Wozniak, P. (2018, 06). *History of Spaced Repitition*. Retrieved from SuperMemo Guru: https://supermemo.guru/wiki/History_of_spaced_repetition_(print)#True_history

Wozniak, P. (2020, August 3). *Less dependent on Internet Explorer*. Retrieved from SuperMemopedia: http://supermemopedia.com/wiki/Less_dependent_on_Internet_Explorer

Wozniak, P. (n.d.). *Incremental Reading*. Retrieved from Super Memory: http://super-memory.com/help/read.htm

ZD Net. (2003, 07 21). *Microsoft tests its own 'dog food'*. Retrieved from https://web.archive.org/web/20080621080007/http://news.zdnet.com/2100-3513_22-5047467.html

Annotated Bibliography

SOURCE	EVALUATION Reputation / ability to see / vested interest / expertise / neutrality/bias
Anki, Inc. (2022). Anki. Retrieved from ankisrs.net: ankisrs.net	This is the site of the people who made Anki, therefore the information is likely to be highly accurate. As the program is free and open source (meaning the code can be read) all information they provide on their software can be assessed by anybody, and they do not earn extra money from more people using their software.
Aravindhan. (2022, 01 05). Seek A Host. Retrieved from https://www.seekahost.in/self-hosting-website/	The authors have high "ability to see" as information on the advantages and disadvantages of web hosting is common information that is freely available on the internet. However, their reliability is decreased as they provide web hosting services, meaning they have vested interest in convincing the reader that self-hosting is a bad idea.
Bae, C. L. (2018). Investigating the testing effect: Retrieval as a characteristic of effective study. Leaning and Instruction.	This source is extremely relevant due to the expertise of the authors, as their publications have over 1300 citations as reported by Google Scholar. Additionally, the Learning and Instruction is run by Elsevier, which is a highly prestigious and long-running journal, increasing its reliability.
Becker, R. (2020, 08 14). Minimal Viable Product. Retrieved from Techopedia: https://www.techopedia.com/definition/27809/minimum-viable-product-mvp	Techopedia has a strong reputation for providing accurate information on tech-related information. Additionally, there is nobody who benefits from correct/incorrect definition of the term "minimal viable product", therefore there is no vested interest.
Blunt, J. R. (2014). Learning with Retrieval Based Concept Mapping. Journal of Educational Psychology.	The Journal of Educational Psychology is under the American Psychological Association, a journal that has been running since 1892, making it highly prestigious and reliable. However, the information was published almost a decade ago, meaning there may have been more recent findings that they were unable to integrate into their conclusions, decreasing the accuracy of their claims.
Community, A. (n.d.). Incremental Reading. Retrieved from Ankiweb: https://ankiweb.net/shared/info/935264945	Anki has a solid reputation for allowing real users to post reviews of Anki extensions on their site, and there is no functionality to tinker with the reviews that are left on the extensions you have published. Therefore, the reviews left on the site have high accuracy, especially as those who have used and tried the extension have high ability to see on its ease of functionality. However, the reputations of the individual review authors are unknown, meaning the information could be fabricated by themselves.
Dempster, F. M. (1989, December). Spacing effects and their implications for theory and practice. Retrieved from Springer.com:	Springer is one of the world's largest journal publishers, and they have a strong reputation for providing accurate papers. Additionally, the authors have little vested interest and high neutrality as they do not mention any specific publishers of spaced-repetition software. However, the paper was published

https://link.springer.com/article/10.1007/BF01320097	over forty years ago, meaning it is likely that opposing evidence has been found on certain points since then.
Incogito, A. (2021, 06 07). First Steps — SuperMemo. Retrieved from supermemo.wiki: https://www.supermemo.wiki/en/supermemo/first-steps	The authors of the site have high 'ability to see' on whether it is possible to download SuperMemo on the mac, particularly as SuperMemo would be heavily incentivised to advertise if it were possible to use it on the mac. Therefore, the information is likely to be accurate.
Instant Paper, I. (2022). Instapaper Premium. Retrieved from instapaper.com: instapaper.com/premium	Instapaper has a positive reputation for being an honest company, particularly due to its small size. As the information is directly from the company, they have expertise on the features of their own product. They may have vested interest in exaggerating the quality of their features, however there would be repercussions if they were to outright lie about their features, therefore they have minimal incentive to do so.
Java T Point. (n.d.). Features of Firebase. Retrieved from Java T Point: https://www.javatpoint.com/features-of-firebase	The authors do not have vested interest in Firebase as Java T Point and Firebase are separate companies. However, Java T Point is not a particularly well-known source in the web development community therefore its reputation is unknown.
Krill, P. (2014, 05 15). React: Making faster, smoother UIs for data-driven Web apps. Retrieved from Infoworld: https://www.infoworld.com/article/2608181/react--making-faster--smoother-uis-for-data-driven-web-apps.html	The authors are not biased as Infoworld is a separate company from React, which are not in competition (Infoworld is merely for information-providing services). Additionally, they have high ability to see the features of React as it is free and open source.
Kromer, T. (n.d.). False Positives and Product / Market Fit Survey – How Do We Test for It? Retrieved from Kromatic: https://kromatic.com/blog/false-positives-and-product-market-fit/	The information on the site is presented with graphics that make it easy to understand. Additionally, there is an open comment section, and the author goes by his real name, increasing the openness and therefore presumed reliability of the source. It could be argued that as there is no date, the information could be out of date, however as it is merely a definition this is unlikely to be true.
Mark A. McDaniel, D. C. (2009, April 1). The Read-Recite-Review Study Strategy: Effective and Portable. Retrieved from Sagepub: https://journals.sagepub.com/doi/abs/10.1111/j.1467-9280.2009.02325.x	The experimental methods were very thorough, comparing active recall to a wider array of study techniques than many other studies. However, Sage publications are a relatively new publishing house (forming only in 1965) so their reputation for the quality of reviewing they do to their publications is unknown.
MDN Web Docs. (n.d.). *Window.getSelection()*. Retrieved from MDN Web Docs:	MDN has an extremely strong reputation in the web development community for providing accurate, up to date information on web development programming languages.

https://developer.mozilla.org/en-US/docs/Web/API/Window/getSelection	
Mulesoft. (n.d.). What is an API? Retrieved from Mulesoft: https://www.mulesoft.com/resources/api/what-is-an-api	Mulesoft is a for-profit company specialising in the supplying of APIs, therefore they have vested interest in convincing the reader of the benefits of APIs. However, as a source it was rather neutral.
Rice, I. (2020). Incremental reading in Anki. Retrieved from wiki.issarice.com: https://wiki.issarice.com/wiki/Incremental_reading_in_Anki	The author Issa Rice is a relatively-unknown independent entity operating on their own blog, therefore their reliability is unknown. However, their ability to see is relatively high as Anki is free and open source so it is easy to see if their claims on an incremental reading functionality having are accurate.
Schwarzmüller, M. (2020, 07 02). Frontend vs Backend. Retrieved from ademind.com: https://academind.com/tutorials/frontend-vs-backend	As an information site, the authors do not have vested interest. Additionally, they have high neutrality as they are not providing a front-end / back-end service and definitions of the concepts are neutral and easy to find.
Shaughnessy, J. J. (1977). Long-Term Retention and the Spacing Effect in Free-Recall and Frequency Judgments. American Journal of Psychology.	The article was published in a well-established journal with a strong reputation, though it is likely to be inaccurate or have incomplete information due to the volume of data on the topic that has been published since 1977, as well as the consideration that this was long before the replication crisis (where the results of large amounts of scientific papers were unable to be replicated).
SuperMemo Community Wiki. (2021). Humans of SuperMemo. Retrieved from SuperMemo.wiki: https://www.supermemo.wiki/en/supermemo/humans-of-supermemo	This source is likely to be biased due to the selection bias: it is more likely that users who have found SuperMemo beneficial would take the time to comment on the SuperMemo site.
Tran, J. (2022, 01 12). MagestorePWA vs Native App and how to choose between them. Retrieved from Magestore: https://www.magestore.com/blog/pwa-vs-native-app-and-how-to-choose-between-them	Magento is a for-profit company, meaning they have vested interest in convincing the reader to use their platform for creating their web app. However, the source itself was rather neutral, fairly weighing the pros and cons. The promotion of its own software was brief and transparent at the end of the article. Despite this, the author of the article, “Jackie Tran” is unable to be found by a google search, suggesting he does not have a particularly strong reputation.
What is React Native? (n.d.). Retrieved from Oreilly: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html	The website has high ability to see as React Native is free and open-source, therefore information on it is easy to acquire and easily verifiable by third parties. It was neutral as a source.

Wozniak, P. (1990). SuperMemo.com. Retrieved from 3.2. Application of a computer to improve the results obtained in working with the SuperMemo method: https://www.supermemo.com/en/archives1990-2015/english/ol/sm2	As the author is recalling his own experiences, he is likely to be biased but also is likely to have the greatest expertise on the topic. However, he also has vested interest, being the creator and maintainer of the SuperMemo software and company.
Wozniak, P. (2018, November 29). Forgetting Curve. Retrieved from SuperMemo Guru: https://supermemo.guru	As the author has worked on a software surrounding this issue for over thirty years, he has a large amount of expertise. However, this could also decrease his reliability due to his vested interest.
Wozniak, P. (2018, 06). History of Spaced Repetition. Retrieved from SuperMemo Guru: https://supermemo.guru/wiki/History_of_spaced_repetition_(print)#True_history	The source is likely to be biased as it is the author recalling his own memories from 30+ years ago, and it is difficult to impossible for third parties to verify information from that time. Despite this, Wozniak is still likely to have the most expertise on the topic.
Wozniak, P. (2020, August 3). Less dependent on Internet Explorer. Retrieved from SuperMemopedia: http://supermemopedia.com/wiki/Less_dependent_on_Internet_Explorer	The source is probably biased. The author has vested interest in convincing users that the program will soon become less dependent on internet explorer in order to retain users. However, the page is edited to and contributed by those other than Wozniak, potentially increasing its reliability (though he still controls the site).
Wozniak, P. (n.d.). Incremental Reading. Retrieved from Super Memory: http://super-memory.com/help/read.htm	The author has a large amount of expertise on the topic of incremental reading, however, has vested interest in convincing the reader that it is useful / SuperMemo is the best program for it.
ZD Net. (2003, 07 21). Microsoft tests its own 'dog food'. Retrieved from https://web.archive.org/web/20080621080007/http://news.zdnet.com/2100-3513_22-5047467.html	The source is neutral in terms of providing a definition of the term, and it is easily verifiable as it is just a term.

Activity Log

Date	Activity
2021-09-15	I received the guidance from my EPQ teacher that I "Have to make something original. You are allowed to be creative in EPQ. If making something, have it be clear what you are expecting to learn from it and how it is differing from pre-existing things." I began making a list of things that I might be interested in. I'm thinking if I do something related to programming then it would provide skills important to the career that I would like to go into. But I have other interests, such as animation and writing, which are far less economically useful but would still be very fun to work on.
2021-09-17	I did more thinking about whether to make an app or write a book. If in Computer Science the NEA permits me to make an app, I'll use this time to write a book, if not I'll write a book for my project.
2021-09-19	I have decided to create t in Computer Science later this year. (Update: I now realise that I was mistaken about this.)
2021-09-22	I set up the code development environment and am working on fetching the data from the links the user enters so that if they put in an article URL the app can collect the text from it.
2021-10-11	Today I learned how to make references in Microsoft word. I have thought more about making a single web app and then just wrapping it for mobile — as in not really making a mobile application. I am still not sure. I also spent more time deciding what framework to use for mobile development, between React Native and Flutter.
2021-10-18	I started creating the Gantt Chart. I've been struggling a lot with fetching the html from links that are entered, which is supposed to be the first and simplest step. If the rest of the project is going to be this difficult, then adding that with the challenge of learning new languages simultaneously may make it too much to learn by the time that the project needs to be done, so I have decided to use React Native as the mobile development framework.
2021-11-08	Over half term I collected 15 sources. I also decided that I may not implement the flashcard aspect of the application, as it may be too much work to realistically do within the time constraints of the EPQ.
2021-11-15	Today I began programming with React Native. I'm using this source: Build an Instagram Clone with React Native, Firebase Firestore, Redux, Expo - Full Course for the beginning bit and at one point there was a bug during importing, and I fixed it by looking up the error message and following a stackoverflow tutorial.

	
2021-11-22	I changed the subtitles for the research review section to Incremental Reading / Competitor Analysis / Design Principles.
2021-12-08	I decided to change the subtitles again, this time to 1. Spaced Repetition 2. Incremental Reading 2. Competitor Analysis and 4. App Development
2022-12-31	I completed the research review. It was rather satisfying to put the differences between Anki, Instapaper, and SuperMemo into concrete comparisons. Though I still have a lot of work to do with actually coding.
2022-01-15	I am experiencing a bizarre amount of error with trying to just import the html of the pages. I am relatively upset at this because this was supposed to be one of the easiest initial parts of the solution. The program is telling me that it cannot fetch the sites using the “fetch” command (the command that all the tutorials tell me to use) because it gives me a CORS error that I can’t figure out how to solve by searching on stackoverflow. It’s not very rewarding trying to fix this.
2022-02-14	I finally managed to fix the bug! I spoke to a friend, and he told me that you can’t use fetch to get external sites from the front end, or else you get that CORS error. It is mildly annoying that I wasn’t able to find this information from the internet by searching for it for hours, but I’m happy I know what’s wrong with it nonetheless and I’m going to start implementing the fix.
2022-02-28	I managed to fetch the html of most sites. However, I am now facing difficulties with getting that received data and pushing it to the main body of my code. (It is stuck on the server-side, and I can’t pass it to the client-side, where all the good stuff happens.)
2022-03-14	I am still struggling to pass the data onto the client side. This was also supposed to be one of the first, easiest parts of the problem, and I have been stuck on it for almost a month. I’ve been looking up resources on how to pass the details from the fetch command over but haven’t been experiencing any luck.
2022-04-10	Outside of working on the EPQ, I am noticing that the amount I have been using SuperMemo for reading has decreased. I think this is because I am realising that I know plenty of extremely smart people who don’t use it, and it seems, counter to the rhetoric I have consumed, that it is (maybe obviously) very much possible to learn a lot from just reading books. However, this could also just be because my tolerance for computer programs that are difficult or slow to use is decreasing. I could of course be

	wrong about SuperMemo, and it is niche enough and unknown enough that it could actually be very useful for learning, but I will see if I continue to use it.
2022-04-15	I have started to read books when I am idle on my phone, using Google Play Books. It does not feel entirely transient, especially not as transient as Instapaper, as I am able to highlight and make notes which transfer to Google Docs. I now am not sure I will use ReadAgain when it is complete. I think it's possible that there is a deeper aspect of learning that is more complex than memorising facts or specific applications of what you have read, and that is more involved with learning something deeper and more meaningful from spending so much time with one author and their ideas. However, I will of course keep working on finishing ReadAgain.
2022-04-16	I initially thought that I may not implement flashcard functionality, however I've changed my mind. I think it will be a lot easier than I thought, especially after coding the article viewing software. I can see how all I need is an if statement to check if it's a flashcard, then to only render part of it whether it's a flashcard, and to change the variable that decides whether it's rendered when "Show Answer" is pressed.
2022-04-20	I completed the first draft of the EPQ and coded a minimal viable product for the application. I finished coding basically all of the initial features today. This was extremely fun and has made me change my mind on what is possible to create under immense time pressure when the stakes are high enough. In the end I actually expected it to take longer and was surprised at how relatively simple the tasks I had to complete were (though they were not necessarily easy). In the end its almost funny that the simplest parts (fetching the articles) took almost 10x longer than I expected, while these parts were almost the same order of magnitude *quicker* than I expected.
2022-06-06	I started to do edits on the first draft, changing sentence order and grammar of what I had written. I find editing grammar to be rather fun, so I got deeply into the flow while doing this and enjoyed it a lot. I look forward to being able to share the project once it is complete. However, I am conscious that ReadAgain is still not something I can see myself using at this point in my life, counter to the utility I would have gained from it had I been using it in September when I first started making it.
2022-06-07	I continued to make edits and was unable to find a source that reflected that spaced repetition was "niche", so I decided to compare the number of google searches for "Spaced Repetition" to the number of google searches for "Study techniques". Of course, my project concerns free learning, which I am realising now that I did not define in the main body of the text, but which refers to learning which is not done for school or other coercion.
2022-06-22	I changed the success criteria to tables, to make it easier to visualise what still had to progress. I think the tables look rather nice, though I am debating whether to use tables for the entirety of the research review section. As I am almost twice the words over the word count, this may be justified.
2022-06-23	I completed edits and increased the number of citations for my claims in the research review. I also changed to using tables for the success criteria. I also completed my first draft to the annotated bibliography. I have also made final changes on my timeline. I decided to use a table instead of using the Gantt chart as it was SO much easier to edit, and I'm happy about this choice.

2022-06-24	I created the presentation today. It was pretty fun and rather simple, I believe I will be able to speak about the topics naturally, as I have spent the last nine months thinking about them, and a *lot* of time before that. I also added screenshots of the code into my main body of text today.
2022-06-27	I read over my project and added more sources. I am doing my presentation later today. I am going to do final grammar and spelling checks soon. I have done my presentation, and it went well. I did two dry runs with two of my friends beforehand and made changes both times, I was lucky to receive very good feedback.
2022-06-28	I added more screenshots of code to my essay, and done extremely final changes. I am going to send it in today.

Timeplanning

Task	To Be Completed By	Actual Completion Date	Notes
Research app stack	September 15th 2021	February 1st 2022	I flipflopped multiple times between doing it in a native way or creating it as a website, and finally permanently decided to make it as a website in February
Write research, find initial sources.	8th November 2021	8th November 2021	Deadline for the first 15 sources was 8th November and I met this deadline
Write paper	20th April 2022	20th April 2022	

Get pulling information from servers	1st February	20th April 2022	This was supposed to be one of the easiest parts of the app early on, but I ended up getting stuck on this for months as it was unexpectedly difficult
Make it able to store things from web servers in the database	20th April 2022	20th April 2022	The rest of the features on the programming side had no specific deadline, and I created them on the night before the first draft deadline
Add sorting options to the database	10th February 2022	N/A	I have decided not to implement this feature due to time constraints, and its unimportance relative to other more central features that I prefer to implement.
Make it able to loop through sorted articles in the database	10th February 2022	N/A	Articles are not sorted, and this will not be complete on the iteration I am working on for my EPQ, however it will one day work

Highlight things on articles	10th February 2022	20th April 2022	Features relating to basic functionality were all implemented on the same day. It was extremely fun
Add ability to make and store highlights on article	10th February 2022	20th April 2022	Features relating to basic were all implemented on the same day. It was extremely fun
Make the ability for highlights to show up on themselves in the queue	25th February 2022	20th April 2022	Features relating to basic were all implemented on the same day. It was extremely fun
Make the ability to create flashcards	25th February 2022	20th April 2022	Features relating to basic were all implemented on the same day. It was extremely fun

Make the ability for flashcards to show up in the queue	25th February 2022	20th April 2022	Features relating to basic were all implemented on the same day. It was extremely fun
Make the ability to grade flashcards	25th February 2022	20th April 2022	Features relating to basic were all implemented on the same day. It was extremely fun
Make the ability to schedule flashcards based on grading	25th February 2022	20th April 2022	Features relating to basic were all implemented on the same day. It was extremely fun
Complete first written draft of report	23rd April 2022	23rd April 2022	I completed the first draft on time and received feedback on the appropriate day

Awaiting feedback	25th April 2022	28th April 2022	
Redrafting	15th June 2022	24th June 2022	I had a lot of time for redrafting
Polishing code	15th August 2022	31st August 2022	This is extremely fun, so it was okay to leave until rather late
Presentation	23rd June 2022	24th June 2022	I hope to finish practically everything related to my EPQ, including the presentation.

Project Q

Objectives

Student: Space Lutterodt-Clottey

Proposal - Objectives
Sections One and Two of the edexcel Project Proposal Form

Section One: Proposed Title
Creating an App for Free Learning

Title or working title of project (in the form of a question, commission, or design brief)

Creating an App for Free Learning

Project objectives (eg, what is the question you want to answer? What do you want to learn how to do? What do you want to find out?):

I want to create an app that allows you to incrementally read a large number of articles simultaneously.

If it is a group project, what will your responsibilities be?

Section Two: Reasons for choosing this project

I chose this project to maximise the amount I will learn in app development, and because I use a program that does this but poorly and would like to improve it and use it.

Date
08/11/2021

Supervisor Sign Off
Yes

Timescales

Student: Space Lutterodt-Clottey

Proposal - Timescales
Section three of the edexcel Project Proposal Form (Activities and Timescales). Activities to be carried out during the project could include research, development and analysis of ideas, data collection, numerical analysis, rehearsal techniques, production meetings, production of final outcome, administration, evaluation, preparing for the presentation etc.

Activities and timescales

Milestone one:

Background Research done (See gannt review for more details)

Target date (set by tutor-assessor):
01/01/2022

Milestone two:

First full draft (See gannt review for more details)

Target date (set by tutor-assessor):
19/04/2022

Supervisor Sign Off
Yes

Recources

Student: Space Lutterodt-Clottey

Proposal - Resources
Section four of the edexcel Project Proposal Form (Resources). Section five included for Performance or Investigation/Field Study Units (Not Dissertation or Artefact)

What resources will you need for your research, write up and presentation (eg, libraries, books, journals, equipment, rehearsal space, technology and equipment, venue, physical resources, finance):

I will need websites and documentation to receive information on how to code the specific features and problems I will come across while programming.

Academic sources would not be ideal as scientific journals are not made for programming.

What your areas of research will cover?

How to design an app, how to program an app. It will primarily involve resolving issues I have while developing the app.

I will also need to research optimum interval gaps for best learning.

Section 5: What problems might you have? (Used only for Performance or Investigation/Field Study Units)

Section 5: What will you do stop this from happening or if it does happen? (Used only for Performance or Investigation/Field Study Units)

Supervisor Sign Off
Yes