[Introduction 2](#_Toc47278433)

[Literature Review 2](#_Toc47278434)

[Self-Directed Learning 2](#_Toc47278435)

[How adults Learn - Andragogy 3](#_Toc47278436)

[Knowles’ 5 Assumptions of Adult Learners 3](#_Toc47278437)

[The Case for Mobile Technology 5](#_Toc47278438)

[Examining the market 6](#_Toc47278439)

[Next Steps 7](#_Toc47278440)

[Methodology and Design 8](#_Toc47278441)

[Requirements Gathering 8](#_Toc47278442)

[Base Requirements 8](#_Toc47278443)

[User-Elicited Requirements (Survey) 8](#_Toc47278444)

[WorkFlow & Tools 14](#_Toc47278445)

[MOSCOW Prioritisation 14](#_Toc47278446)

[Version control (GitHub) 14](#_Toc47278447)

[Design Patterns 15](#_Toc47278448)

[System Architecture 15](#_Toc47278449)

[CRC Cards 15](#_Toc47278450)

[Class Diagram 15](#_Toc47278451)

[The User Interface 16](#_Toc47278452)

[Accessibility Considerations 17](#_Toc47278453)

[Persistent Storage – The Database 17](#_Toc47278454)

[Next Steps 17](#_Toc47278455)

[Construction 17](#_Toc47278456)

[Language Selection 17](#_Toc47278457)

[Coding Standards 17](#_Toc47278458)

[Development methodologies 18](#_Toc47278459)

[AGILE 18](#_Toc47278460)

[TEST DRIVEN DEVELOPMENT 18](#_Toc47278461)

[Code Modularisation 18](#_Toc47278462)

[Activities / Build 18](#_Toc47278463)

[Main Menu Activity 18](#_Toc47278464)

[Subject Section 19](#_Toc47278465)

[Note Section 19](#_Toc47278466)

[Quiz Section 19](#_Toc47278467)

[Video Section 19](#_Toc47278468)

[Flashcard Section 19](#_Toc47278469)

[Evaluation 19](#_Toc47278470)

[Software Testing 19](#_Toc47278471)

[Unit Testing 19](#_Toc47278472)

[Usability Testing 19](#_Toc47278473)

[Expert Evaluation 19](#_Toc47278474)

[Conclusion 19](#_Toc47278475)

[Future Iterations 19](#_Toc47278476)

[Conclusion 19](#_Toc47278477)

# Introduction

No intro yet – last thing to do.

Kostas: I haven’t updated the lit review chapter with your feedback yet. You can skip it entirely. Thanks!

# Literature Review

This chapter reviews a range of literature covering the teaching of adults, self-directed learning and the concept of ‘andragogy’. The intention of this review is to collect information to understand how adults learn, so that it can inform the design process of my software. The knowledge learned in this section will also help decide which kind of application I build, and help shape the potential features and requirements that will then be put out to a survey. The literature reviewed in this chapter is done so in the context of how the information provided relates to software tools that can aid self-directed learning, or how it can be adapted to do so.

## Self-Directed Learning

Self-directed learning (SDL) is the *“process by which learners manage their own learning process from beginning to end” (Knowles, 1975).* One need only perform a simple internet search for self-education to be bombarded with offers from businesses which range from small private companies to massive public institutions. The vast majority of these offers take the form of ‘e-learning’ (marketing language for ‘online’) courses which cover any number of subjects, ranging from simple ‘how-to’ videos, to lectures and courses offered by active and often highly esteemed academics. Indeed the academic industry is now investing significant resources in e-learning infrastructure to support both teaching and learning (Islam, 2013).

Aside from the market potential, or perhaps due to it, the benefits of SDL have been increasingly researched. Studies has shown that SDL is positively related to many education-related constructs: academic performance, aspiration, creativity, curiosity, and life satisfaction (Boyer et al., 2014). Additionally, college students who are taught to be pro-active self-directed learners are better prepared as employees to anticipate the needs of their organisation, and better acquire “skills, knowledge and abilities to create values for their customers, employers, and organisations” (Tobin, 2000).

The purported benefits have been marketed by myriad e-learning and self-education firms looking to create the next mega-hub of online learning. The popularity of these courses is not to be underestimated, in fact, the global online education market size is forecast to increase to $319 billion (USD) in 2025, up from $188 billion (USD) in 2019 (Research and Markets, (2020).

Self-directed learning is a legitimate method of study, the industry is fertile, and the appetite exists for tools that can enhance one’s learning. It is therefore important to examine *how* adults learn so that needs and requirements can be established, shortcomings can be identified and corrected, and the design process can begin.

### How adults Learn - Andragogy

Until the 1970s, it was not well understood that children and adults require different approaches to learning (Knowles, 1975). It was Malcolm Knowles, an American adult educator, who first formally identified the difference, and was also the first to attempt to develop a comprehensive theory of the education of adults -- a concept which he later named ‘Andragogy’, as opposed to ‘Pedagogy’, the education of children (Knowles, 1980). His theory explores how adults have different motivations and methods for learning, thus educators of adults must tailor their content appropriately (Knowles, 1980). Generally, adult education practitioners welcomed Knowles’ theory, especially satisfied that it was applicable in practice (Loeng, 2018).

The theory of andragogy has been studied extensively ever since, though it has not progressed without criticism. Critics accuse Knowles’ theory of lacking an empirical basis (Jarvis, 1984) (Davenport III, 1987), a criticism especially focussed on his ‘assumptions’, which are explored in the next section.

Further criticism accused Knowles of idealism. It points out that Knowles failed to integrate real-world social, political, economic and historical context with his notion of the individual. Thus neglecting consideration of race, gender and class – and their associated privileges and suppression (Sandlin, 2005). This view is shared by others who claim that Knowles’ work fails to critically examine society and organisations, and does nothing to challenge the status quo (Finger and Asún, 2001).

In practice, in higher education institutions, pedagogical education has underwent a significant shift towards community-based learning models (Rovai and Jordan, 2004). During this same time, social networks such as Facebook have gone through explosive growth, initially driven by this student age group, which has led to a pairing of social media and education (Deng and Tavares, 2013). This naturally leads to the question of whether the same type of community-driven learning could be integrated with Knowles’ theory -- perhaps an area for future research to explore. In reality, many self-education systems do have some kind of social or community aspect to them which are examined in a later section of this chapter.

### Knowles’ 5 Assumptions of Adult Learners

Knowles published “The modern practice of adult education: from pedagogy to andragogy” (1980) in which he detailed 4 assumptions about the way in which adults learn (as opposed to children). These assumptions address personal characteristics that, Knowles argues, are common among adult learners. The 5th assumption was added in a later work (1984), and the collection is displayed in the diagram below (fig 1).

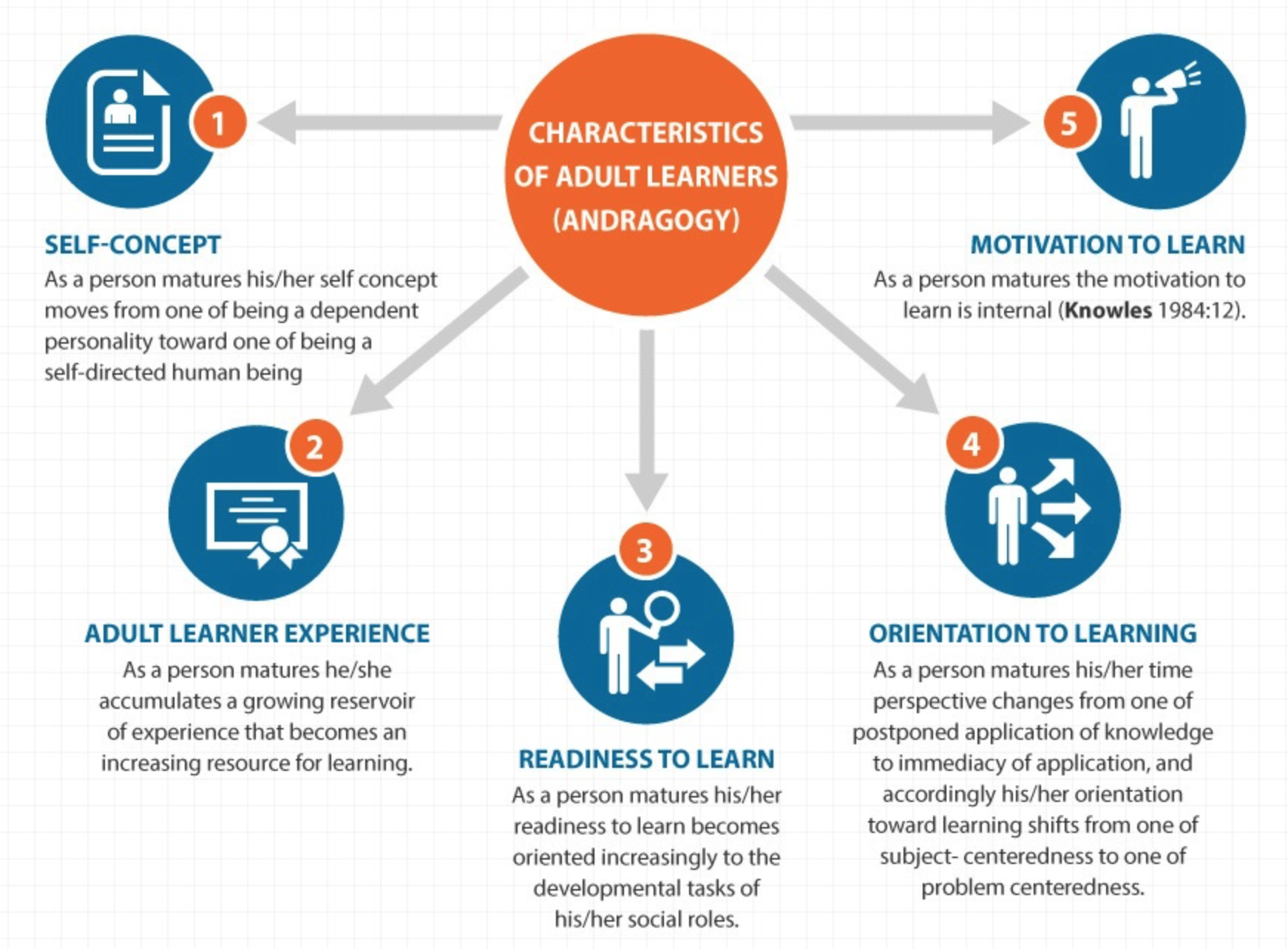


Figure 1 - Knowles' 5 assumptions of adult learners. Credit: https://elearninginfographics.com/adult-learning-theory-andragogy-infographic

From these assumptions can be extracted useful methods and guidelines which, when followed, can give more meaning and impact to learning experiences for adults. In a more immediate context, these assumptions can provide me - as a software designer - a sort of imaginary proto-user who possesses some, or all, of these characteristics as requirements. This section explores how these characteristics have been refined over years of research, and how that can inform the design process.

To satisfy the first assumption, it is important that learning consists of minimum instruction and maximum autonomy. Adults acquire new information more effectively if done so autonomously [CITATION NEEDED], therefore it is important that self-directed learning tools can accommodate an autonomous learning process. [SUGGEST WHAT THAT MIGHT LOOK LIKE?]

Knowles’ second assumption proposes that adult learners naturally have more life experience. Thus, it stands to reason that they typically have a wider knowledge base and are more likely to have different backgrounds, skills and experience levels in any particular subject of study. Not only that, but as age increases, the experience they have plays an increasingly important role in learning new things. Pragmatically, this means that tools to support adult learning should support a wide variety of learning methods, models and subjects to appeal to the broader user-base -- an idea which is upheld in later research (Kebritchi, 2008).

With his third assumption, Knowles asserts that adults tend to engage with learning that will benefit their social development; often we are more willing to engage with learning that can enhance skills that pertain to our social roles, of which one’s job is such an example. Application of this assumption could include features of a self-directed learning tool that emphasise networking and collaboration. Such inter-student interaction is shown to have a positive effect on learning outcomes such as course engagement, critical thinking, and individual development (Pike et al., 2011). Furthermore, the ability to support learning material from institutions that offer official certifications etc. could be a way to exploit this characteristic.

The fourth assumption contends that adults seek practical, problem-centred learning approaches. It suggests that adults learn new skills for specific practical reasons – such as encountering a problem or working in a new industry. Real-world application of characteristic could mean that lessons come in the form of real-life examples or could include some way of facilitating the application of the knowledge gained.

The fifth and final assumption says that as learners mature, their drive to gain new knowledge becomes internal. This contrasts with child learners whose motivations are often driven by external factors such as punishment for poor academic performance [CITATION NEEDED?]. This suggests that learners require a valid reason behind learning. Personal motivation could be aided with a way to track and measure progress towards personal goals.

Due to the deep user-centric insight into how adults engage with learning material, it is important that Knowles’ assumptions are kept in mind when moving to the research and design stages of this project. They will help shape the potential range of features that will be included in the requirements elicitation survey and will also contribute to the overall philosophy for the design process.

## The Case for Mobile Technology

In recent years, the booming self-education sector has been revolutionised by disruptive new technologies (Mac Callum et al., 2014). Mobile technologies (encompassing mobile phones, tablets and other ultra-portable computers) have long been earmarked as a useful tool for learning (Dawabi et al., 2004). Mobile technology has been very rapidly adopted across the world, and it is now increasingly common for people to have their phone on or near them at all times. Recent statistics show that Smartphone ownership in the UK across all demographics ranges from 95% to 99%, with the sole exception of the 55+ bracket at 70% (Statista, 2020). It is fair to say that even among the oldest members of our society, mobile technology is now ubiquitous.

A major strength of these technologies is their versatility. As these technologies have advanced in power, they are increasingly assuming tasks that were traditionally the exclusive domain of PC’s and laptops [CITATION?]. The potential for flexibility offered by mobile technology helps to accommodate different ability levels and learning methods (Kebritchi, 2008). Students looking to support their learning at university prize tools such as emails, YouTube and podcasts (Gosper et al., 2011) – all of which are easily and readily accessible on even the most basic of today’s smartphones. This is confirmed by a recent Pew Research report which found that video-sharing site YouTube was a ‘very important’ source of knowledge when learning how to do new things for about half (53%) of users aged 18-29 and for 41% of users aged 65 and over (Smith et al., 2018).

The efficacy of integrating mobile technology with education has been backed up by research. A 2014 study concluded that the use of technology enhanced student engagement with the material, which in turn improved overall achievement (Fonseca et al.). This conclusion supports the earlier findings of Trimmel & Bachmann (2004), which suggested that students who included technology in their learning reported more interest in learning, higher participation rates in learning and a stronger motivation to do well than those who didn’t use such technology. Additionally, the same 2014 study found a significant correlation between technology use and academic achievement, confirming the same findings from earlier works (Gulek and Demirtas, 2005).

It would seem, then, that today’s smartphones and tablets are an effective foundation on which can be built a tool to support self-directed learning.

## Examining the market

The current market for self-education content is very active, with several important players. It can generally be categorised by the type of content that is provided, or how it is delivered. This section records my personal exploration of the current market and examines the similarities or differences posed by other operators’ offerings in this space. It should be noted that in almost all cases, the sites mentioned in this section have a corresponding native app for mobile use, but here they are treated as the same for the broader purpose of this examination.

Many sites, such as The Great Courses Plus, YouTube and Masterclass depend almost entirely on content delivery via video. As noted by Smith et al. (2018), video is a ‘very important’ source of learning for a significant demographic. Therefore, it would be prudent to include or at least consider a form of integration or other way to accommodate video-heavy learning content.

Other sites, such as Khan Academy, Udemy, and even official online university courses offer a more ‘traditional’ learning experience that generally relies on the written word. Significant consideration must therefore be given to how these sources of content could be integrated or accommodated by a learning tool.

Further still, some apps try to … a more interactive approach to learning. Grasshopper, an app which teaches its user’s basic coding, it is possible to edit, compile and run simple code to perform an operation. This type of content delivery, although technically challenging to implement, could be an effective form of delivery as evidenced by Knowles’ fourth assumption of adult learners (Knowles, 1980).

Q FOR KOSTAS: Do I have to cite these apps/sites? If so, how do I do it?

## Next Steps

This chapter has explored the rise and growth of the self-directed learning movement, the research that has gone into this movement, and the technology that has allowed it grow even further in both its reach and efficacy. Based on the findings of this chapter, there is now a firm foundation on which I can begin to design my app.

Knowles’ theory of andragogy has provided a proto-customer to whom the software can be tailored. Furthermore, by satisfying the assumptions wherever possible, it should help improve user engagement with the software, and create value for the app by satisfying needs that are currently not met by other firms’ offerings.

Mobile technologies could provide an avenue to do this thanks to their versatility. It is evident that with the advancements in mobile technology, smartphones are now a common vehicle through which education can be delivered. The technology also provides the learner with an extra degree of control, which is imperative to increase student engagement, according to Knowles (Knowles, 1975).

To differentiate my software from that already on offer, and in order to avoid competing with vastly greater resources, I have decided that the software will be a type of utility tool, offering no educational content of its own, but instead providing a place to store, access, and review content that has been collected from other sources. By using the information learned in this review, I can begin to compose a more effective survey to elicit specific requirements for the software as I move into the design phase.

# Methodology and Design

## Requirements Gathering

### Base Requirements

The assignment lays out the foundationary requirements for the project. The full assignment can be seen in APPENDIX…

#### Explicit Requirements

The assignment sets out “must-have” requirements:

“The end-product must include functionality on the following aspects:

* *Assess readiness to learn;*
* *Set learning goals;*
* *Engage in the learning process; and*
* *Evaluate learning.* “

#### Contextual Requirements

In addition to these specific requirements, the assignment also includes more contextual requirements.

* “…develop a tool that helps students to familiarise with concepts within a discipline/field as part of their independent/self-directed learning.”
* “Requirements gathering and evaluation must involve users from your target audience.”
* “GUI implementation will be a “must-have” requirement.”

From these contextual clues, further base requirements can be extracted.

The first point indicates that the tool should help students “familiarise with concepts” from a discipline. This suggests to me that the tool should include a feature that allows revision of study material. Additionally, this requirement could be satisfied with a variety of ways to collate and review information.

The second requirement is satisfied and explained in fuller detail in the next subsection (User-Elicited Requirements).

The final requirement is to implement a Graphical User Interface (GUI). Again, this requirement is met and explained more in-depth in a later section of this report (Design - The User Interface).

### User-Elicited Requirements (Survey)

Bearing in mind the base requirements and using the knowledge collected from the literature review, I composed a survey to elicit further requirements from the potential userbase. The survey was made using SurveyMonkey, a popular, free, online tool for composing, distributing and analysing surveys.

Participants were found by advertising for them on social media, and through mass-messages to friends and family members who were, in turn, asked to invite their own friends and family members with a similar method. In the survey, participants were asked to read a short introduction which explained what was expected of them and satisfied ethical considerations imposed by the departmental ethics board. They were then asked a series of 10 questions regarding the development of the software. In addition to basic demographics questions, the participants were asked about accessibility considerations. The remainder of the questions probed their preferences for learning and which features they would like to see in a self-directed learning application. Questions were kept as clear and concise as possible.

In total 25 participants took part in the survey between 28th June 2020 and 5th July 2020, with a 100% completion rate averaging 3 minutes and 44 seconds per response.

Some, though not all, questions and their associated responses have been included in this section for analysis. The full survey can be found in APPENDIX … while the results can be found in APPENDIX …

#### Survey Limitations

Due to financial constraints, the free version of this software was used which incurred a limit of 10 questions for the survey and imposed other limits to its complexity. It is noted that with the ability to ask more questions and being able to compose a more complex survey, the survey could be used to extract more information than is done so here.

The impact of the ongoing COVID-19 global epidemic has resulted in extreme changes to the curriculum and rendered in-person, extended interviews very difficult to arrange and conduct. Furthermore, the inability to access the campus has reduced the potential number of participants. It is acknowledged that this imposes limitations on the value of the survey data and negates the possibility to follow up with questions on specific issues.

With these impediments in mind, the survey was composed to maximise user feedback, with questions often offering more than one response. This permitted me to extract as much data as possible from what I expected to be a small number of participants.

Criticism of the survey (after completion) was generally aimed at its verbosity. In future surveys, I would take more care to simplify the language used so that the survey and its instructions are as clear as possible.

#### Survey Results

Two questions in the survey establish the age-range and gender demographics of the participants. As can be seen in figure 2 below, these questions allow me to verify that I receive as wide a variety of responses as possible since the viewpoints and needs of people can vary wildly with changes in these demographics. An obvious example of this is that older users could more heavily focus on accessibility options than younger, healthier users. The figure illustrates a good variety of age groups represented in the survey, with at least 2 responses from each category (n=25).

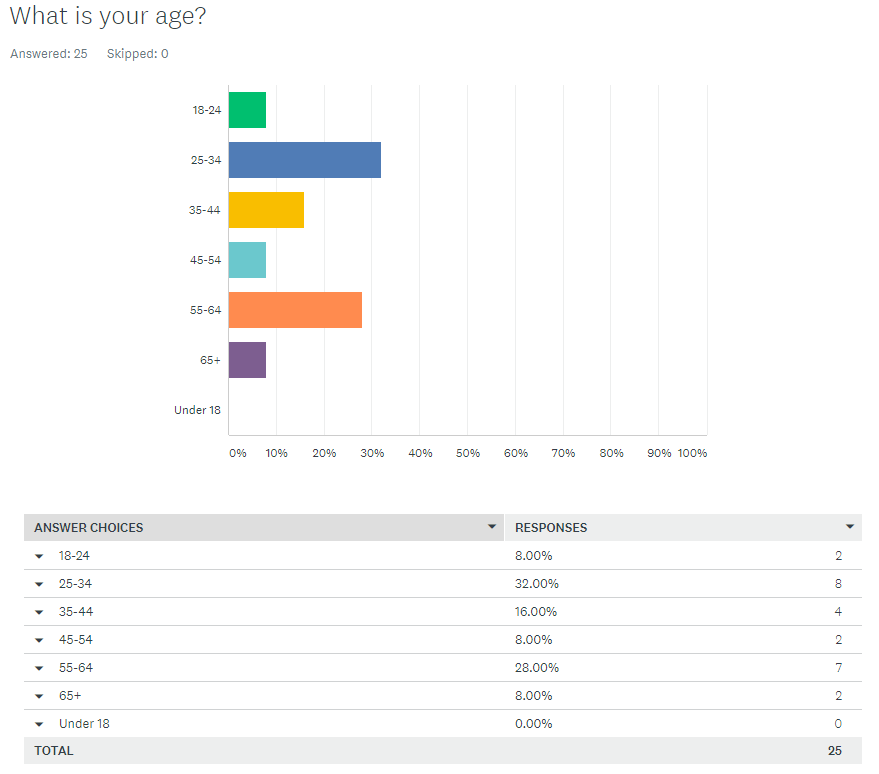


Figure 2 - Age demographics

Two further questions establish if there are accessibility issues that are likely to be encountered by users; and which common accessibility features the users would like to see included. The latter of these questions is seen in figure 3. The former, asking about specific disabilities experienced by the participants, received no responses on all occasions.

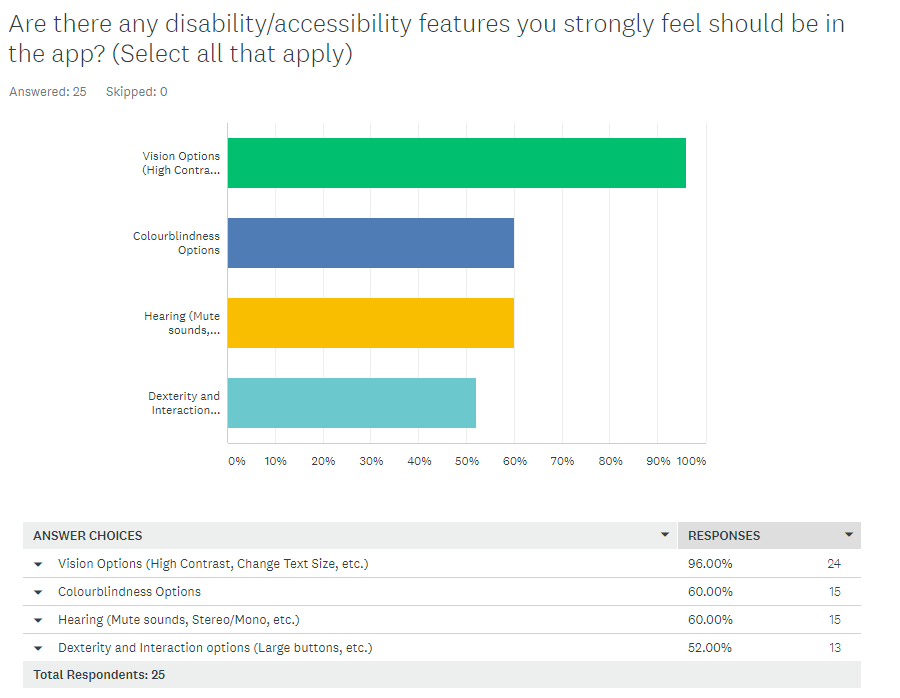


Figure 3 - Requested Accessibility Features

Accessibility features are clearly highly desired. All participants but one felt strongly that vision-impairment accessibility options be available in the app. Great support was also seen for the other 3 accessibility options offered, with each having over half of the total participants’ support. This highlights the importance of accessibility considerations which are otherwise often overlooked. Despite the support for accessibility options, the question that asked respondents if they had any disabilities themselves was skipped (no answer provided) in all instances.

The bulk of the survey asked questions about the learning habits and preferences of the participants. For example, they were asked to list the ways in which they preferred to consume educational content which allowed me to establish commonality between the participants and to see which media types should be supported within the application (see figure 4).

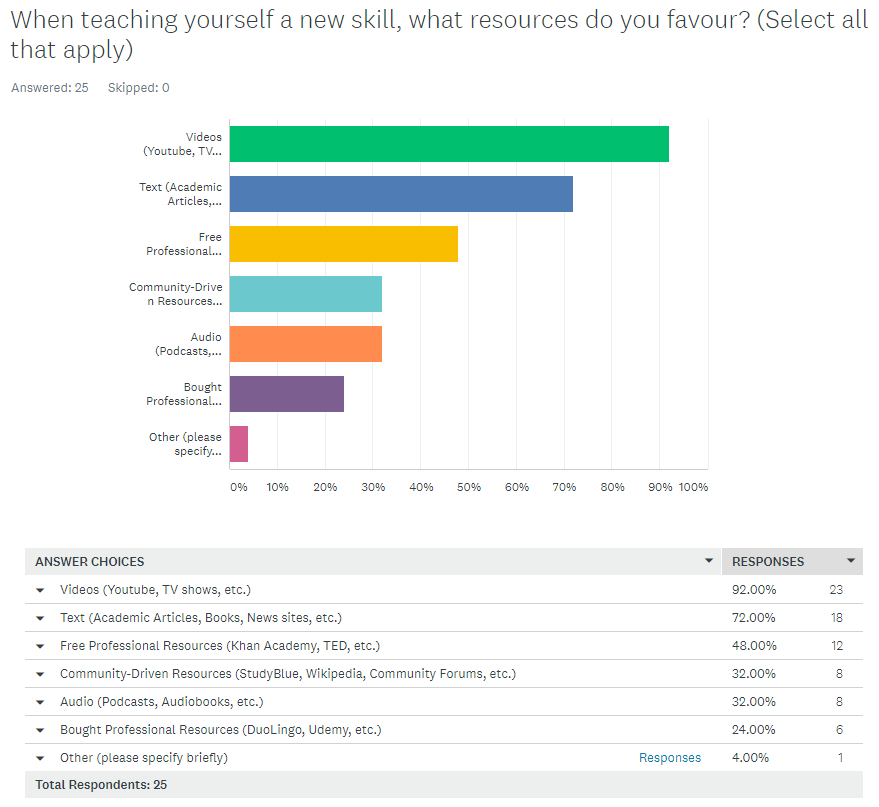


Figure 4 - Learning Preferences

Video is the clear winner in this respect, closely followed by textual sources. This aligns with research from Smith et al. (2018) as noted in the literature review chapter. The importance of video content should therefore not be underestimated. Furthermore, textual sources remain an important medium of learning – support for such material should be included. It is also important to note the popularity of free resources compared to paid ones, with less than a quarter of participants reporting it was their preferred way to learn.

In addition to preferred learning methods & resources, participants were asked which (potential) features they would find most useful for a self-directed learning companion app (see figure 5). In contrast, participants were also asked to mention which features they would not find useful. This allows a greater degree of granularity in the responses as not desiring a feature is not the same as actively discouraging one. The feature options offered were derived from the knowledge gained in the literature review and are inspired by other educational apps.

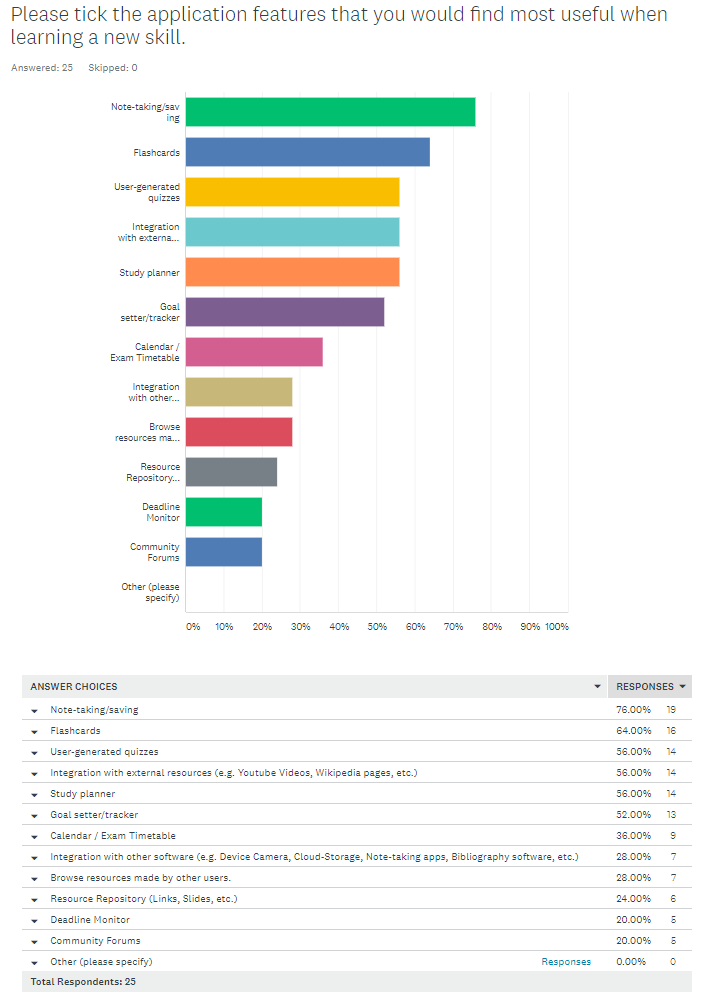


Figure 5 - Desired Features

Purely in terms of requirements gathering, this is a key question of the survey. The data shows note-taking, digital flashcards and user-generated quizzes as the top 3 requested features. This suggests a strong desire for revision-based features, and surprisingly, relatively little appetite for community-based features such as browsing other users’ resources and the use of community forums. This lack of appetite is reinforced when users were asked which features they believed were non-essential, with community forums (42%); deadline monitor (32%); and browsing other users’ resources (26%) occupying the top three spots of that list.

Another key decision to be made, which would affect the architecture of the software, was whether to support networked user interaction. That is to say, should the app support the ability for users to share their notes and other content with each other? Reception to the idea was lukewarm, with only 28% of respondents finding this feature useful. When given a more specific question about this, only 12% of respondents wanted this mutual note-sharing to be activated by default (fig 6). This was a surprise as it contradicts research by Knowles (1975) & Pike et al. (2011) which suggests adult learners prefer to learn in communities.

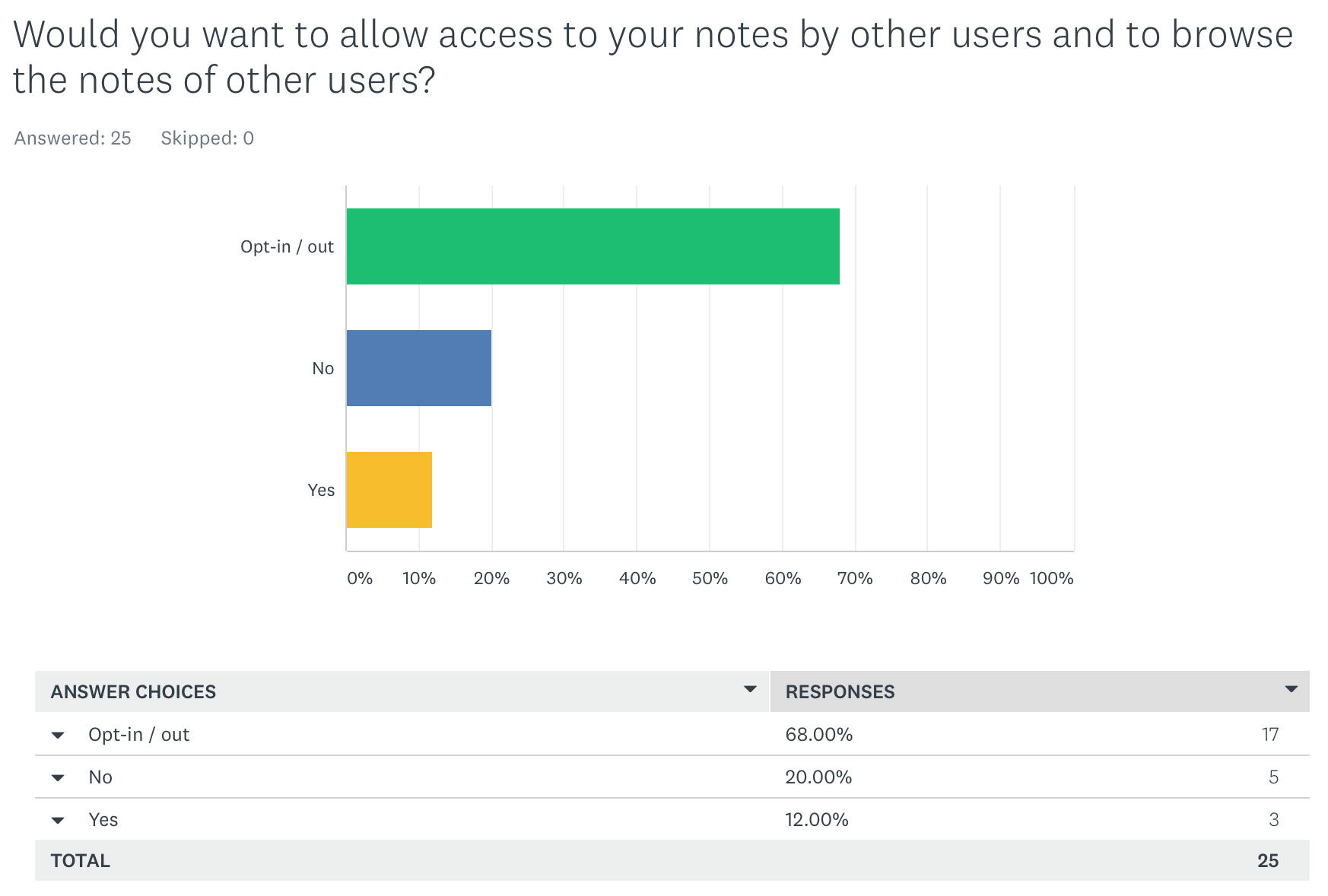


Figure 6 – User Interaction Preferences

Opinion on this feature was difficult to determine, but I concluded that sentiment was generally negative. More than two thirds (68%) of respondents would prefer an opt-in/out system, and there were more solid no responses than yes responses. Combined with the previous question responses, low interest in the ability to browse other users’ resources was identified. Because the time required to learn networking could be better spent in other areas that are more desired, I chose not to implement any networking functionality. This could be a point to expand upon in a later version of the software.

Finally, participants were asked from which devices they would most frequently access an application like this (figure 7).

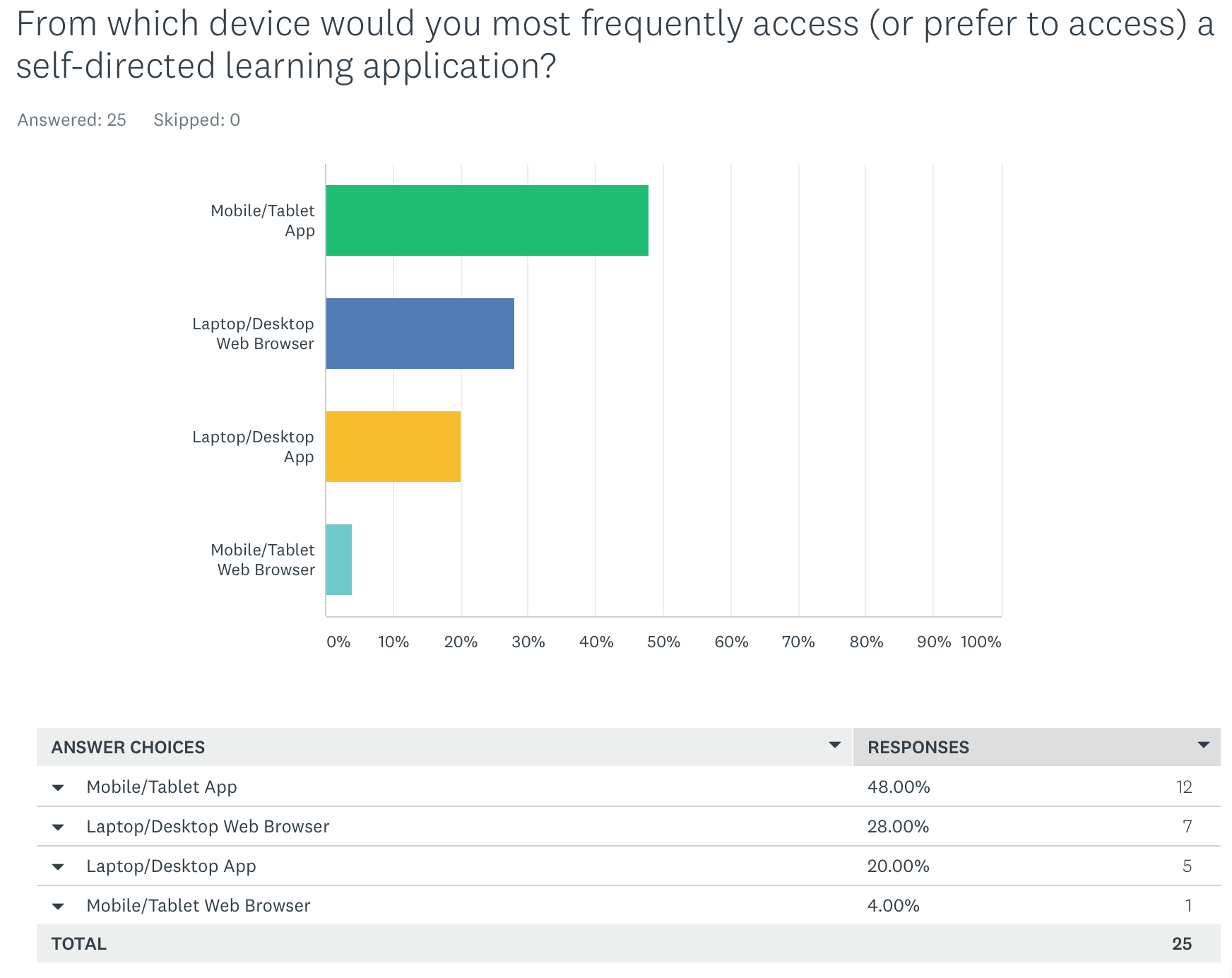


Figure 7 - User device preferences

Overwhelmingly, users would prefer to access such an application as a native app on mobile devices. Though there was relatively strong support for a web app, it was concluded that development would focus on a native mobile app given the time constraints involved.

#### Survey Conclusions

The survey provided several key insights that will prove very useful. Accessibility features are very important and were widely requested – with vision-related options were the most requested feature. In terms of content support, text and video are the most widely used educational media formats. Features that support or integrate these are must-have requirements. User networking features were not strongly desired and so they will not be included. Finally, the app will primarily be developed as a native mobile app.

For this project, user preferences determine prioritisation of development. I intend to work from most requested features to least so long as time permits.

## WorkFlow & Tools

Iterative process without rigid structure. Allows flexibility time-wise to learn new technologies.

Worked one activity at a time, basic front end, back end, completed front end.

Not chosen to follow agile framework but adapt useful techniques for my own use – e.g. epics and user stories?

### MOSCOW Prioritisation

Mainly based on the survey results and in addition to the base requirements, the ‘MoSCoW method’ was used to prioritise feature development -- features were categorised into four categories: Must-have, Should-have, Could-have, and Would-have. Each category receives successively less prioritisation than the last. Using other tools, such as version control, will allow me to track this prioritisation and track progress more effectively.

### Version control (GitHub)

The first step in creating this software will be to create a repository using a version control tool. Although version control programs resulted as a natural consequence of multiple authors working on a single ‘work’, modern tools also provide benefits to the solo developer.

Services like GitHub provide a remote backup of any digital documentation or code that is generated, which is easily accessible from a multitude of devices and operating systems. Generally, they also store backups of previous versions or ‘builds’ of the software, allowing changes to be quickly undone if something goes wrong down the line. Having a known, functional backup can free the developer to be more experimental with their code – further to this a changelog is normally a core feature of these tools. Should an ‘experiment’ go wrong, the changelog can be an invaluable source at locating the source of the fault. Failing that, the developer can revert to a previous version of their code. Finally, these tools allow other parties to collaborate or simply monitor the progress of the system. In my case, this will be ideal for my supervisor to oversee the progress of the project should the need arise.

### Design Patterns

Another design method employed in this stage was the use of design patterns, which bring a plethora of benefits to the project by providing reliable, known solutions to common problems encountered in any software design stage. Not only is time saved by not having to write and test the software/procedure anew, but time is also saved in the design stage by not having to create, from scratch, a certain procedure nor do I have to consider too deeply the implications of its implementation. Only a relatively small amount of effort will be needed to integrate any relevant frameworks with my code.

Early on, I identified at least one design pattern that would prove useful in my design - the singleton - which ensures that only one instance of a class can exist. In this case, it was noted that a singleton could be beneficial in instantiating only one database connection that could be recalled when appropriate, as each activity opening a new connection to the database would quickly result in excessive computational loads.

## System Architecture

### CRC Cards

A useful approach to creating object-oriented software was to use Class-Responsibility-Collaboration (CRC) Cards as a method of visualising the software’s architecture.

This design technique was particularly useful for establishing the classes that make up the system and the data that those classes would manipulate. Each class to be created was written on an index card with its responsibilities (i.e. the data it will manipulate) and the other classes with which it will collaborate. An example of such a card is shown below, and a full collection can be found in APPENDIX…

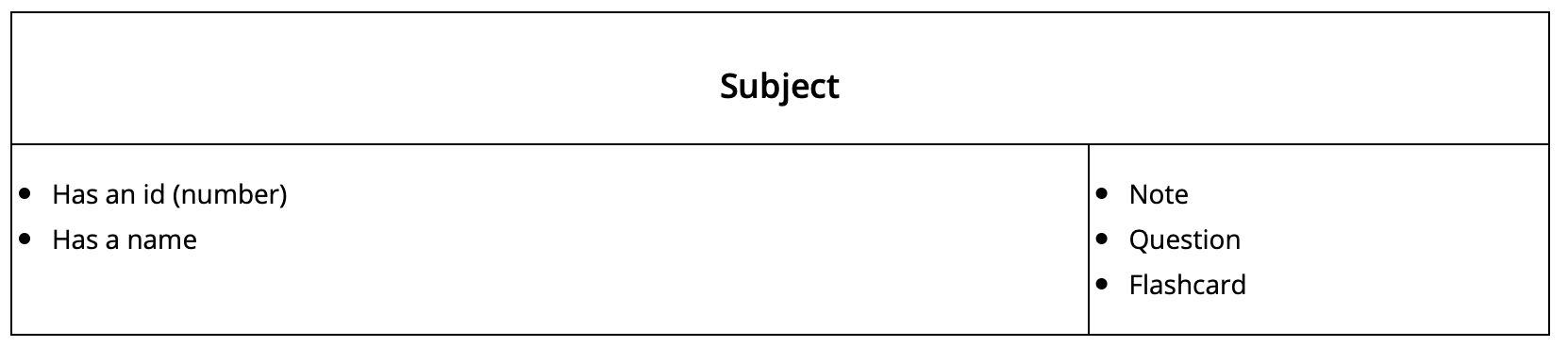


Figure 8 - A CRC card representing the 'Subject' class

By expanding this across the entire system, the task of visualising the system and the intricacies of its operation became much simpler. This technique excelled at highlighting potential sources of conflict within the system and undoubtedly improved the robustness of my code, long before any of it was written. Any abnormalities or problems uncovered at this stage were ironed out at this early juncture instead of costly (in terms of time) changes being required during the implementation stage.

With classes established, this information was further translated into a more graphic presentation and was expanded upon by developing a class diagram.

### Class Diagram

A natural progression to CRC cards, Class Diagrams present the architecture of the software, relative to its classes, data, and functions. Such a diagram was created (fig. 9) using free tools online that produce a visual diagram using unified modelling language (UML) – a code-language-independent interface for creating software architecture diagrams.

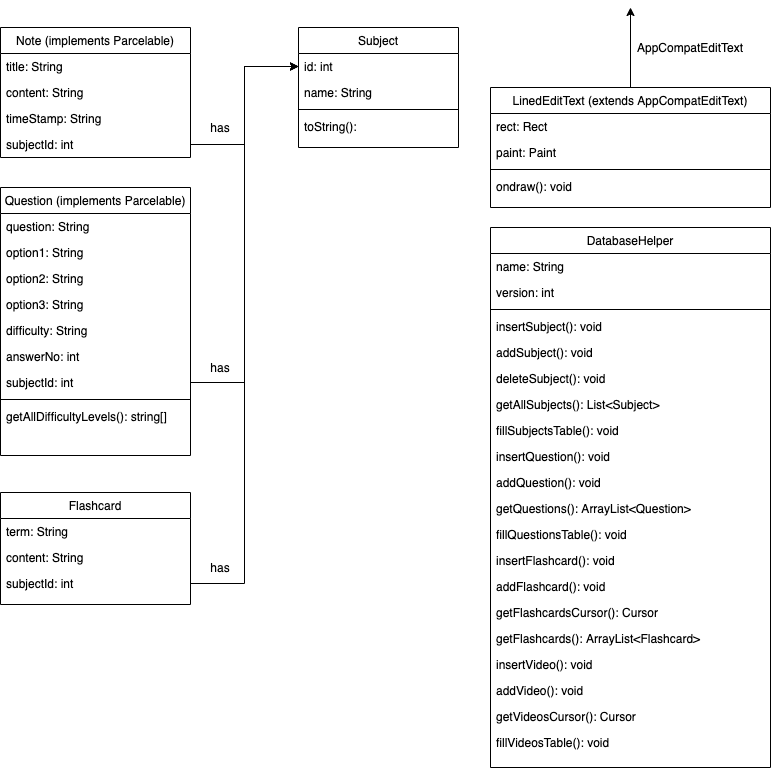


Figure 9 - The Class Diagram

The diagram provided me with a helpful visual layout of the software, always offering a next step of which code to implement. Additionally, though not necessary to this software’s development, these diagrams could be useful to future, non-technical collaborators of the software. Business analysts, for example, can use this information to model the system from a business perspective, perhaps to monetise the application among other things.

## The User Interface

The user interface – the way users interact with the software – was designed with the intention of being simple and logical to navigate. Data collected from the survey suggests potential users have a wide range of ages and therefore, experience with technology. It cannot be assumed that users are ‘digital natives’ therefore buttons and other interactable elements should be clearly labelled with the use of icons to denote actions reduced wherever possible.

The software’s functionality will be divided into sections (notes, flashcards, videos, etc.) which stem from a main menu and each section can be further categorised by subject. Screens will be kept as consistent as possible with each other to improve the system’s ‘learnability’ – how easy it is to remember how to navigate and use the software – and a colour scheme will be adopted that allows users to tell, at a glance, within which section of the app they are. This will be further highlighted by clear navigation patterns and informative feedback to the user (e.g. a toolbar which names the current screen).

### Accessibility Considerations

Accessibility options were very important to people in the survey. High contrast, large clear text whenever possible. Interface should negate clutter wherever possible. Buttons should be clear and easy to press. No complex gestures to be used. Simple swipes and taps only.

## Persistent Storage – The Database

Database schema and very short description.

Various ways of achieving data persistence. Shared prefs file was ruled out due to size limitations. Database good choice because large media files not stored. Can be expanded upon later.

## Next Steps

The documentation produced by these techniques will provide unambiguous direction on how to approach coding the system as I move to the next phase. Classes and their data have been specified by the CRC cards and a Class Diagram produced. By this point, I have a clear idea for which platform I will develop, the language I will use, and any relevant frameworks that I can exploit.

# Construction

This section will cover the coding stage of the software development cycle. It is here that a language will be chosen, and the various tools that will aid in the implementation of the system are identified. Additionally, some development styles will be discussed.

A good way of getting started with the code itself is to build a ‘spike’ – a feature of the software which uses as many ‘layers’ (e.g. database, front-end, etc.) as possible. This allows the me to see the scope of the project, get a feel for how the various layers will interact, and allows for experimentation with new or unfamiliar systems to take place at a time when mistakes can be made without severe consequence. By coding a ‘slice’ of the software in this way early on, any issues that could arise from interfacing layers can be eliminated quickly.

## Language Selection

Thanks to the data gathered from the survey, an android native app was selected for development. Android was selected over iPhone due to personal experience – this would reduce development time significantly.

Implementation of the software’s code will take place using Java. In addition to being the language with which I have the most experience, it is also a very versatile language that is more than likely capable of providing the required functionality for this type of system. Further to this, its versatility is such that it could be used to develop a desktop, web, or mobile application with little effort required to port the source code between platforms. It is important to consider these points of extension early in development so it can be exploited with minimal problems should the app be successful in the future.

Java has many IDE’s available for use, which come with many features. For mobile applications, Apple and Google have their own IDE’s for iOS and Android respectively. Google’s ‘Android Studio’ tool, the official integrated development environment for the Android operating system, supports Java natively and is the IDE with which I have most experience for developing apps.

### Coding Standards

By following conventional standards of programming, the code will be ‘readable’ – something which is especially important in this instance, for example when I share my code with my supervisor. This increased readability also aids in software testing and maintenance.

Although there is no formal requirement to adhere to any specific coding style, I will endeavour to follow conventional guidelines as much as possible for the reasons just outlined, and as practice for developing in industry.

## Development methodologies

### AGILE

‘Agile’ software development refers to any process within a group of project management methodologies that focus on iteration, frequent inspection and adaptation. At the end of each evolution cycle, a demonstrable product is shown to stakeholders, feedback and criticism can be received, then the prototype is iterated again. As its name suggests, this allows the developer to adapt to changes quickly and minimises overall risk by creating transparent communication between the developer and the client. Having to demonstrate the software also encourages high-quality code rather than bodged fixes.

This methodology was used for the reasons outlined above. In addition to these advantages, I personally benefitted from the clear, attainable goals and deadlines that were generated by referring to the documents produced from the design chapter.

### TEST DRIVEN DEVELOPMENT

As a technique that is commonly used in Agile development, this methodology drives development by establishing a software goal, writing a test that could verify the implementation of that goal, then writing minimum code that could pass the test. Then a new goal is set to improve upon the functionality or create some new functionality, and the process repeats. This ‘back-to-front’ approach permits the developer to focus on writing ‘clean’ (i.e. organised) code that works.

This style of programming also forces the developer to think through the design before writing the code. Code can be added knowing that the existing codebase is stable and works. This naturally has a beneficial effect when looking for bugs, as we know the problem is most likely to arise from recently added code or an existing system that is being reworked – furthermore, developers (original or not) can refer to previous tests so they can create new ones, or to understand how the code is supposed to function.

Many IDE’s, including the one I have chosen, include or support, plugins that can automate much of this process. Software testing will be examined more closely in the next section of this paper.

### Code Modularisation

For this software, I chose to follow a modular programming style. Modular code is that which is generated into ‘chunks’ called modules. A module is an independent section of code that provides aspects of functionality without relying on other modules. This approach to the software’s construction lends itself well to test-driven development techniques, and for object-oriented programming generally, as both methodologies encourage decomposition of computable systems into smaller pieces.

This modular approach allowed me to focus on learning how to implement certain systems almost as mini-projects of their own, then combine them when ready at a later point in the pro. This keeps software testing clear and concise, bug tracking easier to perform and keeps the code clean and readable on a grander scale.

## Activities / Build

### Main Menu Activity

### Subject Section

Explain this is the root. a Categorisation class by which all other functionality is organised.

### Note Section

### Quiz Section

### Video Section

### Flashcard Section

# Evaluation

## Software Testing

### Unit Testing

## Usability Testing

## Expert Evaluation

# Conclusion

## Future Iterations

Wider applications include use in educational institutions which could distribute material to be accessed by students within the app. Potentially the distribution could take place over the app.

## Conclusion

**Citations**

BOYER, S. L., EDMONDSON, D. R., ARTIS, A. B. & FLEMING, D. 2014. Self-Directed Learning. *Journal of Marketing Education,* 36**,** 20-32.

DAVENPORT III, J. 1987. Is There Any Way Out of the Andragogy Morass? *Lifelong learning,* 11**,** 17-20.

DAWABI, P., WESSNER, M. & NEUHOLD, E. 2004. Using mobile devices for the classroom of the future. *Learning with mobile devices research and development. Editado por Attawell, J. & Savill-Smith C***,** 55-59.

DENG, L. & TAVARES, N. J. 2013. From Moodle to Facebook: Exploring students' motivation and experiences in online communities. *Computers & Education,* 68**,** 167-176.

FINGER, M. & ASÚN, J. M. 2001. *Adult education at the crossroads: Learning our way out*, Zed Books.

FONSECA, D., MARTÍ, N., REDONDO, E., NAVARRO, I. & SÁNCHEZ, A. 2014. Relationship between student profile, tool use, participation, and academic performance with the use of Augmented Reality technology for visualized architecture models. *Computers in human behavior,* 31**,** 434-445.

GOSPER, M., MALFROY, J., MCKENZIE, J. & RANKINE, L. 2011. Students’ engagement with technologies: Implications for university practice.

GULEK, J. C. & DEMIRTAS, H. 2005. Learning with technology: The impact of laptop use on student achievement. *The journal of technology, learning and assessment,* 3.

ISLAM, A. K. M. N. 2013. Investigating e-learning system usage outcomes in the university context. *Computers & Education,* 69**,** 387-399.

JARVIS, P. 1984. Andragogy—a sign of the times. *Studies in the Education of Adults,* 16**,** 32-38.

KEBRITCHI, M. 2008. Examining the pedagogical foundations of modern educational computer games. *Computers & Education,* 51**,** 1729-1743.

KNOWLES, M. S. 1975. *Self-directed learning : a guide for learners and teachers,* Chicago, Association Press.

KNOWLES, M. S. 1980. *The modern practice of adult education : from pedagogy to andragogy,* Wilton, Conn.

Chicago, Association Press ;

Follett Pub. Co.

KNOWLES, M. S. 1984. *Andragogy in action,* San Francisco, Jossey-Bass.

LOENG, S. 2018. Various ways of understanding the concept of andragogy. *Cogent Education,* 5**,** 1496643.

MAC CALLUM, K., JEFFREY, L. & KINSHUK 2014. Comparing the role of ICT literacy and anxiety in the adoption of mobile learning. *Computers in Human Behavior,* 39**,** 8-19.

MARKETS, R. 2020. Global Online Education Market - Forecasts From 2020 To 2025

PIKE, G. R., KUH, G. D. & MCCORMICK, A. C. 2011. An investigation of the contingent relationships between learning community participation and student engagement. *Research in Higher Education,* 52**,** 300-322.

ROVAI, A. P. & JORDAN, H. 2004. Blended Learning and Sense of Community: A Comparative Analysis with Traditional and Fully Online Graduate Courses. *The International Review of Research in Open and Distributed Learning,* 5.

SANDLIN, J. A. 2005. Andragogy and its discontents: An analysis of andragogy from three critical perspectives. *PAACE Journal of Lifelong learning,* 14**,** 25-42.

SMITH, A., TOOR, S. & VAN KESSEL, P. 2018. Many Turn to  YouTube for Children’s Content, News,  How-To Lessons. Pew Research Center.

STATISTA. 2020. *Smartphone ownership penetration in the UK, in 2012-2020, by age.* [Online]. Available: <https://www.statista.com/statistics/271851/smartphone-owners-in-the-united-kingdom-uk-by-age/> [Accessed 15th July 2020].

TOBIN, D. R. 2000. *All learning is self-directed : how organizations can support and encourage independent learning,* Alexandria, VA, ASTD.

TRIMMEL, M. & BACHMANN, J. 2004. Cognitive, social, motivational and health aspects of students in laptop classrooms. *Journal of Computer Assisted Learning,* 20**,** 151-158.