Application to Help Students Prepare for Placement and Graduate Interviews

Dissertation Project

Date:16th September 2022

Supervisor: Martin Chorley

Student ID: 21018815

Course: MSc Computing

Abstract

This project investigates a mismatch between employers' and graduate expectations in the Technology Sector, which could be accounted to the graduates and Higher Education institutions not putting enough emphasis on the importance of soft skills, opposing to employers actively seeking them. Competency-based interviews are the most used tool for assessing graduate's soft skills, therefore, practicing those interviews will allow graduates to improve their soft skills and to articulating them in verbal communication.

This project aims to solve the problem by implementing a web application that helps students develop their soft skills through competency-based interview practice and explores the most effective ways to do so. Based on the findings, a survey is created, and requirements devised. This project focuses on "Competency-Based Interview Preparation", one of the 3 most requested functionalities, and proposes how the others can be implemented in the future to expand it.

In the final stage of the project, a web application is developed, where students can access interview questions, answer them and favourite them in their profile.

Table of Contents

Table of figures

Table of tables

1.Introduction

Computer Science is one of the most popular Higher Education disciplines for which the number of students has grown by 13% resulting in 158,340 applications for bachelor's degrees in the UK in 2022 (BCS 2022). It is believed that Computer Science degrees guarantee employability for students due to the high demand for technology professionals in the market. However, 7% of Computer Science students are reportedly unemployed, higher than the 6% unemployment average of other high education disciplines in 2020.

Why does this phenomenon occur and why is there a mismatch between the graduates' *and employers' expectations?*

This research investigates the reasons for the higher unemployment of Computer Science students and the expectations dissonance between employers and graduates in the Technology sector. It explores the importance of soft skills and the recruitment process for graduates and aims to help graduates and placement students prepare for competency-based interviews.

In the first section, this report defines its main aim and objectives. After, it presents a Literature Review outlining the gap between employers' expectations and graduates' skills, its rationale, researches the specific skills required to be successful in the job market and how to teach them effectively. In addition, the literature review identifies and assesses existing software applications to practice competency-based interviews and explores different technical practices and methodologies for designing applications. A questionnaire is designed based on it and distributed between students in their second or final year of Computer Science or related subjects (Bachelor's and Master's). The questionnaire results are evaluated in conjunction with the findings from

the secondary research. They underpin the requirements for an application that improves employability amongst students. The following chapters discuss and analyse the application's design, implementation and areas of improvement. In the final section, this report reflects on the achieved learning.

2. Aim and Objectives

2.1. Aim

This project aims to enhance and advance students' soft skills by creating a user-friendly software application that unifies good quality resources to prepare for competency-based interviews.

2.2. Objectives

- Research and understand the reasons for the unemployment gap among Computer Science graduates
- Investigate and evaluate successful approaches for creating e-learning applications
- Examine students' self-reported needs to prepare for the job market
- Devise functional requirements based on the primary and secondary research findings
- Design and implement a software solution that:
    a. Fulfils the functional requirements
    b. Follows software-engineering best practices and quality metrics
    c. Protects user’s personal information
    d. It is easy to use, aesthetically pleasing and follows an intuitive design
- Reflect on the solution and give recommendations for improvement

3. Literature Review

3.1. The Gap in Skills Expectations

There is an increasing demand for Computer Science graduates in the labour market, which is reflected in the double-digit increase of Software developers in the UK in the last 9 years (Alexandra 2022). The UK ranks third in the world for impact technology investment and that number has doubled since 2018.

Nevertheless, Computer Science graduates have one of the highest unemployment rates (7%) in the UK in comparison to other disciplines, as reported by Mantle (2021) and Smith et al. (2018). Thus, it could be claimed that graduating from a prestigious university with outstanding academic results does not guarantee a full-time job in the IT sector, regardless of market demand. As Bridgstock (2009), Joseph et al. (2010) and Schipper and van der Stappen (2018) reported, this poses a significant risk for the UK labour market as graduates do not appear to be “work ready” and suggests that employers’ needs and graduates’ skills seem to be misaligned (Shadbolt 2016; Mason et al. [no date]).

Almonte (2022) and Shadbolt (2016) hypothesise that employers are dissatisfied with “graduates’ soft or work readiness skills”, and Shadbolt (2016) concludes that the inability of Computer Science students to articulate their skills or lack of the same explain their high unemployment rate. Moreover, Bridgstock (2009) and Shadbolt (2016) continue to demonstrate that students favour specialist and theoretical skills over employability skills due to the vast number of vacancies in Computing. For example, employers take graduates’ “technical skills for granted but are looking for adaptability and transferable skills” (Department for Business, Innovation and Skills 2016; Mason et al. [no date]).

In addition, there is no universally agreed skillset for Computer Science jobs as it is a field with highly varied job opportunities. For instance, some employers look for graduates with a good technological foundation, and others look for the most recent and up-to-date programming languages and software engineering methods (Shadbolt 2016). Therefore, graduates need to be highly flexible and aware from the start of their career where they aim to be, which could be problematic when lacking work experience (Department for Business, Innovation and Skills 2016). Moreover, it is not feasible for Higher Education Institutions to drastically adapt the course curriculum every year, making it hard to keep pace with new knowledge and technologies. Given it is virtually impossible to keep their technical skills up to date, soft skills become critical and the only constant factor that can help students find a job in the Technology sector.

Thirdly, whilst previous work experiences such as summer internships have a significant positive impact on graduates' employment (CHERI 2002; Smith et al. 2018), many graduates describe their job-seeking experience as a "classic catch-22 situation" (Department for Business, Innovation and Skills 2016). They find that they cannot get a job because they do not possess enough work experience. Most employers prefer candidates with work experience, even for entry-level positions (Shadbolt 2016).

Furthermore, companies and recruiters follow a very structured and thorough application process to filter the vast number of candidates (Higginbotham 2021), making it harder for graduates to stand out. Thereby further emphasising the importance of students' employability and work-readiness skills.

Finally, unemployment amongst graduates is considered one of the main reasons for low life-satisfaction and high anxiety levels affecting their mental health (Graduates'

subjective wellbeing | HESA. 2020). The next sections dive deeper into the main reasons for unemployment amongst graduates.

3.2. Defining Employability

Employability is defined as “the extent to which somebody has the skills, knowledge, attitude, etc. that make them suitable for paid work”(Oxford Advanced Learner’s Dictionary 2022); however, this definition does not shed light on what factors influence it.

McQuaid et al. (2005) addresses two main theories addressing employability, a broad view proposing that it depends on individual factors, external factors and personal circumstances, and a narrow view focused on the personal readiness to work:

- Personal circumstances are defined as access to resources, caring responsibilities (for children and relatives), work culture within the social-economic circle and access to a support network. It could be argued that these circumstances are unique to the individual, rendering virtually impossible any generalisation; therefore, this study will not investigate further the impact of personal circumstances onto graduates’ employability.
- External factors are considered the labour market demand, access to public services and job-matching technology (McQuaid and Lindsay 2005). This research assumes that most students with computer science degrees have access to the internet and online job listings, and as Nation (2021) reports, the labour market demand is at its highest. Therefore, external employability factors are in favour of Computer Science graduates.
- Individual factors are where the two theories overlap and could be described as “an individual’s readiness to work” (McQuaid et al. 2005). The more significant

number and relevance of the students' skills, the higher the probability of landing the job. This research focuses on improving those skills to make Computer Science graduates more employable.

3.3. Individual factors

According to the Confederation of British industry (CBI) 2019 report (CBI 2019), employability skills are defined as a “set of attributes, skills, and knowledge that all labour market participants should possess to ensure they have the capability of being effective in the workplace – to the benefit of themselves, their employer and the wider economy”. Employability skills could be roughly split into technical and non-technical skills (Chou 2013).

Shadbolt (2016) and Sobral (2019) argue that Computer Science is a vast field subject to constant change and advancements with over 700 programming languages, rendering it virtually impossible to keep up to date with the changes. Although most popular programming languages remain, there is an overwhelming number of new frameworks and trends to follow (TIOBE Index. [no date]). Opposing this drastic change, the university curriculum provides foundational knowledge that enables students to continue their future education., thus it has not changed that much over the years (Shadbolt 2016). Therefore, when employers advertise entry-level positions based on specific technological skills, they are not facilitating graduates to join the labour market.

According to (Clarke 2009), “career and employment patterns have fundamentally changed and will continue to change in the future”; therefore, the traditional mindset of graduating from university and finding a job with technical skills is outdated (Chou 2013). As reported by Mansour and Dean (2016), non-technical skills (soft skills) have

higher demand in junior-positions and this is especially acute in the Technology sector, where companies are looking for professionals that could be easily trained and have experience learning new technologies (Todd 2017; McQuaid et al. 2005; Almonte 2022).

This is further confirmed by the 80,000 hours' "Technical report on Skills" analysis (Ponsukcharoen, 2017), where the most common technical and non-technical employability skills are evaluated. Figure 1 from the report shows that the most sought skills are non-technical. Programming ranks low when compared to skills such as judgement, decision making, critical thinking and time management. Whilst this shift appears in all industries, it is amplified in the Technological field due to the aforementioned fast-changing technical knowledge (McQuaid et al. 2005; Almonte 2022).

If soft skills are so crucial for employers, what are they? How can a graduate reach their full potential by learning them?

Figure 1. Skills that help employability (Ponsukcharoen 2017)

3.4. Soft skills

Searching for 'defining soft skills' on Google scholar provides over half a million results. Whilst, this research cannot claim to have found a universal definition of soft skills and their categorisation, there are commonalities between the employers' requirements for soft skills. For example, Chou (2013) defines 11 'smart' soft skills classified into 6 categories (Table 1).

Categories	Functions
Communicating (the absolutely necessary)	Communications Smart
Dealing with people (the essential)	People Smart Marketing Smart
Dealing with self (the basics)	Work Smart Time Smart Career Smart
Dealing with boss (earning trust and recognition)	Job-interview Smart Boss Smart
Dealing with staff (inspiring loyalty and productivity)	Motivating Smart Delegating Smart
Being visionary	Beyond the Box

Table 1 - Soft skills classification by Chou (Chou, 2013)

Alternatively, Misra and Khurana (2017) define 22 similar skills in 6 categories (Table 2).

Skills	Action Verbs
Technical skills	Basic literacy, Learnability, Technology skills, Numeracy skills, Adaptability
Higher order thinking skills	Occupational Knowledge, Learning, Reasoning, Creative Thinking, Decision-making, Problem-solving
Personal Skills	Knowledge, Integrity, Self-control, Self-confidence, Emotional literacy, Initiative
Social Skills	Teamwork, Respect, Ethics and Values, Networking, Interpersonal Skills, globally aware
Generic Skills	Leadership, Teamwork, Project Management, Oral Communication skills
Self-perceived employability skills	Resilience, Behavioural Skills, Social Networking Job-Seeking Skills, Labout Market Knowledge

Table 2 - Soft Skills classification by Misra and Khurana (Misra and Khurana, 2017)

Whilst both discuss communication skills, they break them down differently. For instance, Chou (2013) classifies them as a category and Misra and Khurana (2017) split different aspects of communication between all categories. In some cases, the classification is the same between the authors, e.g., dealing with self-skills (Self-perceived employability skills). Similarly, the 'Tuning' methodology (Tuning 2008) defines 3 categories of skills: Instrumental, Interpersonal and Systematic. In Misra and Khurana's definition, instrumental Competences correspond to 'Higher-order thinking skills'. Interpersonal competences are categorised as individual and social, which is similar to Almonte's (2022) and Chou's (2013) statement of soft skills being about "dealing with people" and "dealing with the self". Finally, systematic competences are analogous to Misra's and Khurana's "Generic skills" of Leadership, Teamwork, Project Management and Oral Communication skills. Therefore, whilst each skill can be split into different categories, the skill core factors overlap across the definitions; see the example of communication. In addition, Ethics and Psychology theory roots can help explain the different categorisations but similar meaning of soft skills.

"Practical intelligence in IT: assessing soft skills of IT professionals" describes soft skills as "the managerial, intrapersonal, and interpersonal skills that are used to resolve IT-related work problems (Joseph et al. 2010). They took IT professionals at different levels and presented them with situations that required them to demonstrate the possession (or lack of) different soft skills. One of the article's conclusions is that the main difference between graduates and experienced IT personnel is their ability to handle different situations that test their soft skills (Joseph et al. 2010).

This leads to the argument that soft skills are not static and can be learnt. Most people would learn them as they become more experienced; however this can be achieved earlier through:

- game-based, project-based and competition-based learning,
- a combination of formal, non-formal and informal education,
- self-development practices and technology-based learning,

Being the most popular strategies.

3.5. The recruitment process and role of soft skills in IT

The recruitment process differs between companies; however, competency-based, also known as behavioural interviews, are generally used to assess candidates' soft skills and cultural match (Devins and Hogarth 2005; Falls et al. 2014; Soft skills – how to assess them. [no date]). Therefore, graduates must demonstrate their soft skills during the interview process.

Preparing for those interviews can be difficult and time-consuming (Coach 2020; Mitzi [no date]) . Whilst most universities, including Cardiff, provide employability coaching and mock interviewing services (Careers and Employability. [no date]), there is a limited capacity of the hours a university can invest per student. In addition, it could be claimed that most students do not have experience applying for jobs and, therefore, lack confidence in their skills (Mitzi [no date]).

Even though there are many resources to prepare for interviews available online, these resources are not always of good quality and might not be free of charge, resulting in students utilising inappropriate materials and hindering students of lower socioeconomic backgrounds.

In conclusion, it could be argued that teaching students employability skills and how to articulate them might improve the unemployment rates with the least cost.

3.6. Best approaches for learning applications

E-learning is an idea that dates back to the emergence of the web. Marc Rosenberg (2001) discusses it alongside the new (at the time) delivery of the internet to phones, accompanied by an example of a state-of-the-art digital assistant, PalmPilot (Figure 2)(Rosenberg 2001)

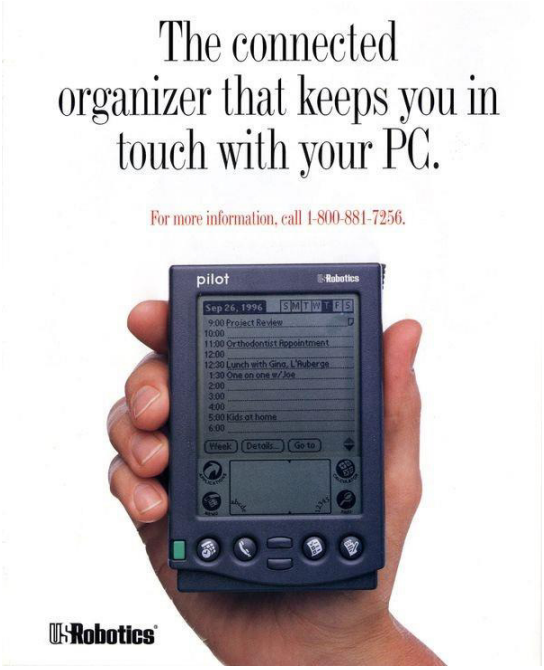


Figure 2 - PalmPilot device (USRobotics 1996)

Evidently, technology has changed dramatically since Rosenberg discussed the concept of e-learning 21 years ago; however, one of the case studies in his book analyses the adaptation of soft skills learning course into an e-learning platform (Rosenberg 2001, p.). The contributing factors of unsuccessful e-learning applications reported by Rosenberg (2001) are similar to those presented by Sharma in the article from 2020 (Sharma 2020). The 2001 publication conducted user testing on the soft skills available in the e-learning platform and reported users finding it “plain boring” due to the endless pages of text, uninteresting layout, and lack of responsiveness of the web. This corresponds to another article by Patel (2019) examining the importance of easy-to-use and interactive user interfaces with the right amount of detail. Rosenberg (2001) and Sharma (2020) identify that the information's accuracy is vital for e-learning platforms' success. Once learners are drawn to the information quality and design, users must be retained on the platform through the principles and techniques to create habits.

These principles and techniques have been proven by the language-learning application “Duolingo”, which has over 500 million total users, 40 million monthly active users, ranking it the world’s leading language learning application. Upon opening the app, users can navigate to a lesson with two clicks. Moreover, classes are split into small, easily digestible tests that reward points upon completion.

Duolingo lessons also follow Merrill’s First Principles for instruction based on 5 statements that are found true for effective learning and often applied to e-learning applications (Table 3). The users of Duolingo are shown new language content in various real-live situations (Demonstration and Task-Centered principle) that they must apply (Application principle). Often recordings and a Duolingo Podcast contribute to making the learning as close to the real world. Finally, there are often quizzes and refreshers on older lessons' material.

In summary, the application is intuitive, simple to navigate and offers instant gratification with minimum effort. This immediate satisfaction it brings is at the root of creating habits, a pivotal strategy of “Duolingo” (Duhigg 2013). For instance, The app keeps track of how many consecutive days users have used it and rewards them with a “streak”; however, they need to start building their streak again if a single is missed.

The design is crucial to creating an engaging e-learning application and Duolingo ticks the box for each guideline. For example, it has an inviting colour scheme with smooth edges, creating a contemporary and approachable feel (Rauser 2021). In addition, “Duolingo” attempts to appear approachable with encouraging and friendly language messages (Nare 2021) to boost user motivation through their learning journey, which is a crucial component to forming long-term productive habits (Duhigg 2013).

In conclusion, attractive e-learning applications tend to be simplistic and approachable, make users feel comfortable and reduce their mental load to a minimum with easy navigation and limited decision making.

Principle	Steps	Statements
Demonstration principle	Observe a demonstration.	Learning is promoted when learners are engaged in solving real-world problems
Application Principle	Apply new knowledge.	Learning is promoted when existing knowledge is activated as a foundation of new knowledge
Task-centred principle	Engage in task-centred instructional activity.	Learning is promoted when new knowledge is demonstrated to the learner
Activation Principle	Activate prior knowledge and experience.	Learning is promoted when new knowledge is applied
Integration principle	Integrate their new knowledge into their everyday world.	Learning is promoted when new knowledge is integrated into the learner's world

Table 3 - Principles of instructional design (Pappas 2017)

3.7. Evaluation of existing applications to prepare competency-based interviews

The researcher evaluated the current applications available to prepare for competency-based interviews and summarised the results below.

Name	Platform	Benefits	Disadvantages
Job Interview: Question and Answer	Mobile	The application has high reviews on google play and provides a variety of interview questions.	The app is only available for Android. It does not allow users to answer questions.
Interview Trainer	Mobile	Various questions are split into categories by skills or industry—hints to help the user answer the question. The tone of voice is analysed, giving the user feedback based on emotion, language, and social tones. Users can record their answers as videos, transforming the speech into text input to mimic a real-life interview.	No focus on the level of experience.
Interview Questions	Mobile	Variety of questions and example responses from professionals and entry-level candidates with appropriate reasoning for the answers and tips functionality. Ability to input your response and memorise it in your account and favourite questions.	No significant disadvantages are found in the functionality. UI and UX design is not attractive/friendly.
Glassdoor	Web	It gives the users the opportunity to input questions from actual job interviews and to identify relevant questions for individual companies.	The focus of Glassdoor is not interview preparation and it's widely focused on everyone, not graduates explicitly.
Career Side Kick	Web	Gives a variety of questions and example answers.	No ability to create a profile and save answers there.

Table 4 - Comparison of different applications on the market

The review found that a wide range of mobile applications aid in preparing for competency-based interviews. However, the web applications only provide guidance for answering questions but no functionality to input answers or personalisation. Thus, there is a gap for an application to prepare for competency-based interviews on the web platform. In addition, mobile applications provide less flexibility on average (Gazzawe 2017) and web applications are easier to integrate with the existing learning systems at universities, for instance, Learning Central at Cardiff university. No desktop applications are found potentially because of the lack of portability.

All the reviewed applications provide slightly different functionality. However, two exhibit all the principles of instructional design discussed previously in the context of the Duolingo app (Allen 2011). These are the Interview Trainer and Interview Questions mobile applications; therefore, they are used as a blueprint for the website in this project. After reviewing the highest-rated features in Google Play and App Store in comparison to the objectives of the projects, some appropriate features are:

- Ability to create and save user answers
- Access to a variety of questions and answers
- Ability to ‘favourite’ relevant questions
- Ability to analyse the tone of voice based on different factors
- Transferring speech to text to record answers easily

3.8. UI and UX Design

As established in the previous section, proper design and instant positive reinforcement are critical for e-learning applications to make users feel comfortable and motivated to continue their learning. Therefore, a user-friendly and aesthetically pleasing design is a requirement for the success of an application as captured by the intersection between User Experience (UX) and User Interface (UI) design:

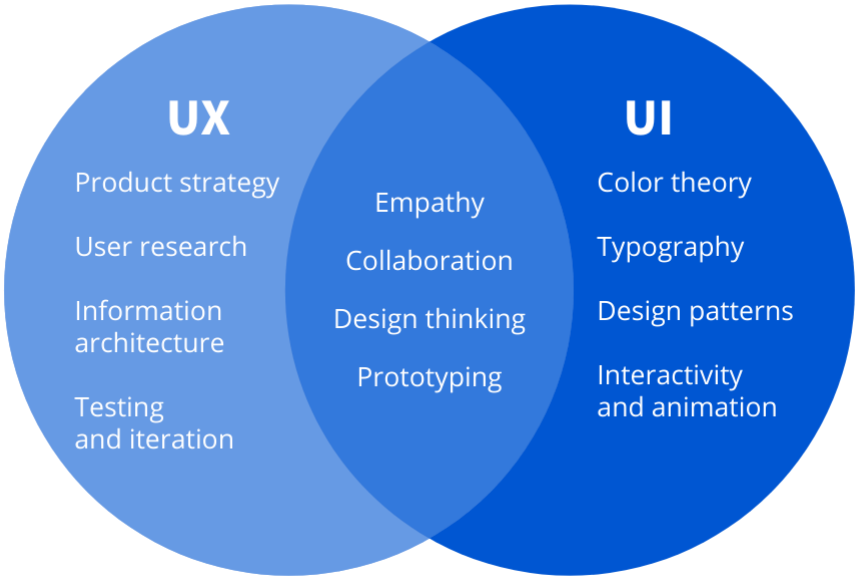


Figure 3. UI vs UX Design (Corsera,2022)

3.8.2. User Experience (UX) Design

UX design includes user perception, efficiency and ease of use to help improve the application usability (Krug 2006; 9 Rules For UX Design. 2018; Professional Web Design Process Explained in 5 minutes 2020). The usability is achieved through in-depth knowledge of the user and their needs and by reducing the cognitive load as much as possible (Krug 2006; 9 Rules For UX Design. 2018). Good UX design is consistent and prioritises function over aesthetic, accounts for user errors and is compatible with common assumptions made by the user. For example, users

recognise that an enabled button on the web is clickable; therefore, if they click on it and do not receive a result, this can confuse them (Krug 2006).

Nevertheless, accessibility should be considered, and the product should be tested on a variety of users, including users with disabilities (Accessibility for Teams. 2022). After establishing the user's needs, site objectives and functionality, UX design levarages Information Architecture principles to structure the data to facilitate understanding (information design). In addition, it also deals with navigation design, which helps a user to easily move through the information architecture (Babich 2022). In the final stages of UX design the interface elements that user interacts with are planned, usually by creating wireframes.

In practice, creating a good user experience involves in-depth user research, creating user stories, personas and wireframes to deduct the most effective design for the application to maximise the usability and accessibility of the product.

3.8.3. User Interface (UI) design

Once the UX is completed, the User Interface is designed, which includes establishing interface elements, their design and branding. UI design focuses on creating the most aesthetically pleasing design to engage the users and correctly convey the application's message (What is User Interface Design? [no date]).

UI design is also described as graphic design for applications based on principles categorised between familiarity, clarity, simplicity and user control (The 6 Key Principles of UI Design. 2021).

These principles are implemented via a variety of techniques. For example, maintaining consistency through repetition makes the design familiar and enhances usability by increasing familiarity and clarity. Furthermore, using an easy-to-read font,

well-spaced layouts and contrasting colours supplements readability (clarity) and positively impacts the usability of an application. Following similar layouts and being consistent in the design themes, colours and font makes navigating easier (Krug 2006; The 6 Key Principles of UI Design. 2021).

Moreover, establishing a good visual hierarchy also makes navigation straightforward. This could be done through the website's typography and via forms and colours. For example, using the same styling for main headings signals the user what topic the paragraphs below will discuss.

Shapes and colours could incorporate the product's message and have proven to convey a powerful psychological impact. For instance, a very famous study on the effect of colour is the "Baker-Miller" pink, where it is concluded that the presence of a certain shade of pink (Figure 4) calms prisoners. Unfortunately, this is only a short-term effect, and in the long term, the colour triggered more violence (Drunk Tank Pink. [no date]). Regardless of its success, the study shows how powerful the psychological impact of colour could be and the importance of conveying a message through it. A similar psychological impact is demonstrated in the Duolingo example, where soft edges and curves convey friendliness and amiability.

Finally, in today's world, there are various screen sizes and display options for content. Graphic User Interfaces should be adapted so that they are not only aesthetically pleasing and usable on as many devices as possible (Friedman 2018).

Figure 4 - Baker-Miller Pink Prison in Europe (Pinterest,2017)

3.9. Software development lifecycle

Software development lifecycle dates back to the 1970s and is developed to control and systematise the ever-increasing complexity of business systems (Elliott 2004).

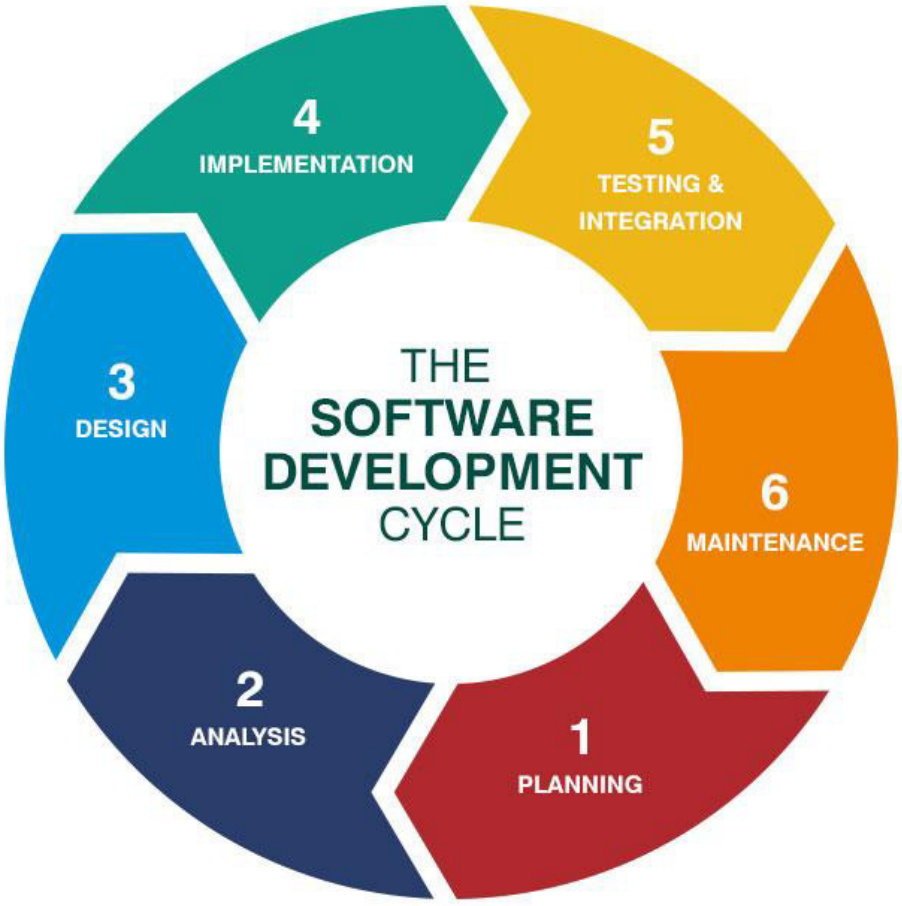


Figure 5 - Software development lifecycle *(Prokopiško 2019)*

There are different versions of the theory; however, this section explains the most popular one consists of 6 steps (Figure 5).

Planning: This is the project's inception phase, where its feasibility is determined. In this phase, the project risks and opportunities are identified and used to devise the project's requirements (Newell [no date]). This is also called “requirement gathering” phase or “initiating phase” (Boyde 2014; Author 2021).

Design: In this stage, detailed system specifications are devised from functional requirements and non-functional requirements such as security concerns are

established (Boyde 2014; Author 2021). Examples of the variables assessed during this phase are mitigating risks, technologies to be used, team capability, project constraints, time and budget.

Implementation: The team builds software solutions according to the given architecture and designs in this phase. The high-level view of the system is now translated into individual tasks and day-to-day schedules (Newell [no date]).

Testing & Integration: The system is comprehensively tested, and bugs are raised with the team. Different levels of testing are executed to confirm the reliability of the product.

Maintenance: After the software is released to customers, the team focuses on maintaining it by releasing new features, fixes and addressing bugs promptly.

Depending on the chosen methodology, these steps are executed once (e.g. Waterfall) or iteratively (e.g. Scrum, Kanban).

3.10. Software development methodologies

As the previous section discusses, software creation is often executed through the different phases of the software engineering lifecycle. These phases are commonly applied in other engineering disciplines and are executed in parallel by different teams collaborating with each other. Naturally, collaboration here is vital, as some tasks cannot be executed before others. Therefore, there is a plethora of software development methodologies such as waterfall, iterative, incremental, spiral, evolutionary, aiming to optimise this process (Al-Saqqa et al. 2020). These methodologies typically differ on two main criteria:

- Linear or non-linear software development lifecycle

- Amount of preparation, document and project management required

Two of the most popular and ever-opposing approaches are Agile and Waterfall. Waterfall is a linear methodology that follows the Software Development Lifecycle. This approach has ‘fallen out of fashion’; however it is still widely used in situations where iteration is not possible, the project is constrained by time or cost and the requirements are very well understood (Raza 2020; When to Use Waterfall vs Agile - Macadamian. [no date]).

Often claimed to be a methodology, Agile is defined as a set of lightweight principles that serves as a basis for various Agile methodologies (Al-Saqqa et al. 2020). This approach is defined in the Agile Manifesto, in 2001 (History: The Agile Manifesto. [no date]). The main objective of Agile Software Development is flexibility and at its core are the four pillars of Agile:

1. Individuals and Interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Agile has 12 main values, see image below:

1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	7	Working software is the primary measure of progress.
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	9	Continuous attention to technical excellence and good design enhances agility.
4	Business people and developers must work together daily throughout the project.	10	Simplicity–the art of maximizing the amount of work not done–is essential.
5	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	11	The best architectures, requirements, and designs emerge from self-organizing teams.
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 6. The 12 principles of Agile (Agile Aliance 2015)

Those principles expand on the four pillars of Agile and define the basis for different methodologies. As evident from principle 3, Agile methodologies follow an iterative approach, meaning that the execute the Software Development Lifecycle non-sequentially. Moreover, what makes this approach different is also the focus on collaboration and team self-reflection (principle 12) and self-organisation (principle 11). Feedback tends to be delayed in waterfall approach and only given at the end of the Software development lifecycle, making the design and requirements a constant work in progress. That is also a pivotal difference between the two approaches, as Agile does not require complete requirements (Raza 2020; When to Use Waterfall vs Agile - Macadamian. [no date]).

Nevertheless, changing requirements obstruct the establishment of clear deadlines and resource planning (Disadvantages of Agile. [no date]). Some consider that the working software, centricity of Agile, leads to limited documentation in comparison to Waterfall approach (The Disadvantages of Agile Methodology. 2019; Disadvantages of Agile. [no date]). More differences between Agile and Waterfall development are discussed in Figure 7.

In conclusion, software development methodologies should be studied and evaluated carefully, according to the project's needs, the team's size and available time and financial resources.

Figure 7. Agile vs. Waterfall (Codegiant 2020)

3.11. Software design principles and patterns

This section explains the benefits and disadvantages of software development practices, designs and patterns later used in the Implementation section.

3.11.1. SOLID Principles

Since Robert Martin published SOLID principles, the term has gained popularity and is now taught at universities and tested on technical interviews. Those principles enhance software quality criteria factors and eliminate “rotting design” practices (Martin 2000; Oloruntoba 2021). A brief definition of each of them is found below:

- **Single-responsibility Principle (S):** A module should have exactly one reason to change. In other words: a class should have exactly one purpose (Oloruntoba 2021).
- **Open-closed Principle (O):** A module should be open for extension and closed for modification (Martin 2000). Paraphrasing, one should be able to change and build on what a class does without changing the code of the class.
- **Liskov Substitution Principle (L):** Subclasses should be substitutable for their base classes (Martin 2000). For example, if a class X takes and argument of type Y, then the derived class of class Y can also be used as an argument in class X.
- **Interface Segregation Principle (I):** Many client-specific interfaces are better than one general-purpose interface. Meaning should a change is made to a general-purpose interface all the derived classes are affected and their behaviour needs to be confirmed. Hence, general interfaces make the system more rigid and hard to test, scale and maintain (Martin 2000).

- **Dependency Inversion Principle (D):** Depend upon Abstractions. Do not depend upon concretions. This principle allows decoupling of dependencies and enhances the software quality factors (Oloruntoba 2021).

In conclusion, SOLID principles allow the system to be more reliable, maintainable and testable. They provide ease when debugging and provide better scalability.

3.11.2. Multi-tier architecture

In this type of architecture, software components are organised in tiers based on their functionality. Three-tier architecture is the most common occurrence of this architecture and is the one that this report explains. As illustrated in Figure 8 this architecture consists of three tires:

- **Presentation**: This is Graphical User Interface and is a tier responsible for the app's presentation, be it in a browser, mobile application or another platform. Typically, it communicates with the Application tier through API calls, following a protocol.
- **Application**: This is the middle layer of the application that encapsulates the business logic. All interactions between the data tier and the presentation tier go through here.
- **Data**: The data storage layer of the application.

The benefit of this pattern is that it decouples the three tiers and makes the application easier to scale and maintain as fewer dependencies are managed. Layers could be separately updated or replaced with newer technologies and that will not affect the application. Finally, security is improved as the Presentation tier never communicates to the Data tier.

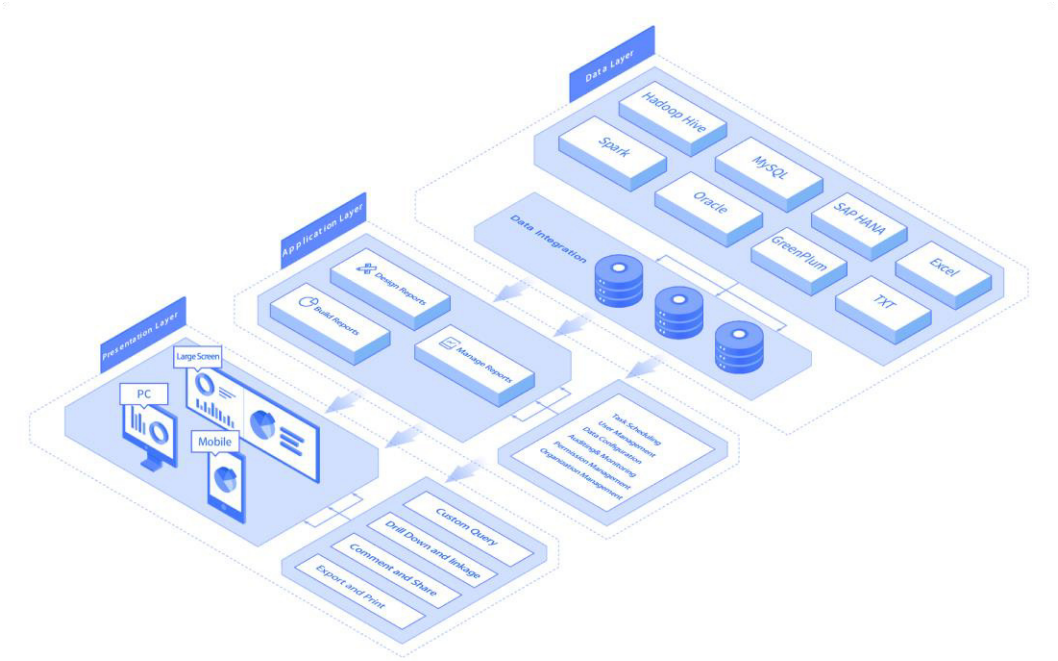


Figure 8. - Three-tier Architecture (Finereport, 2021)

3.11.3. Microservice architecture

Instead of one monolithic system, the application is defined as a set of loosely coupled collaborating services. Each service is independent of the other and the data is maintained consistent between them. The services communicate through different protocols such as HTTP or message broker.

3.11.4. Domain Driven Design

“Domain-Driven Design is an approach to software development that centres the development on programming domain model that has a rich understanding of the processes and rules of a domain” (Fowler 2020). The domain logic becomes the foundation for building the application and external dependencies are abstracted to allow for various integrations and technology changes. Furthermore, DDD ensures that the domain is well understood before development begins, thus allowing for less error in logic overall and better maintainability. A full description of Domain Driven Design could be found in Eric Evans’ “Domain-Driven design” (Evans 2003); however,

this project describes only some main concepts that are directly utilised in the application design phase.

Domain Driven Design proposes two types of design, strategic and tactical. Strategic design describes the concepts and abstract definitions of the domain, diving into an in-depth definition of the business problem and processes, whereas tactical design defines classes and modules (Evans 2003).

Strategic design uses a variety of tools to achieve its goals:

Ubiquitous language: same words are used between all team members for the same concepts.

Bounded context: This is a logical boundary where domain names have a unique interpretation, creating the ubiquitous language. For example, the concept of appointment could have different meanings between the Social Media and Hospital bounded context (Table 5). A Context Map groups bounded contexts used across the entire application.

Social Appointment	Medical Appointment
- Different number of people - Never urgent - Flexible time limit	- Always for one person - Urgency level - Strict limit

Table 5 - The concept of "appointment" in different contexts

Shared Kernel: this is typically a core domain, or a set of generic subdomains shared between different domains

Tactical design goes into a more granular and technical level in the application, it refines the domain model in a way that it is convertible to code. This is achieved by using the following concepts (the list is exclusive):

Value Object: an immutable object whose value is of importance and encapsulates domain logic. Two objects with the same value are considered the same. (DDD Part 2: Tactical Domain-Driven Design | Vaadin. [no date]).

Entity: a mutable object created for its identity. For example, two people with the same name will be considered two different people, unlike in the Value Object definition. Thus, two entities with the same type and id are always considered the same, regardless of their value (Evans 2003). The aim is to have fewer entities and more Value Objects (DDD Part 2: Tactical Domain-Driven Design | Vaadin. [no date]).

Aggregate: This is a collection of domain objects that logically belong together. Aggregates have an entity root, only accessible from the outside and holding reference to the other objects within the aggregate (Avram and Marinescu 2006).

Domain Driven Design follows a project structure, where everything depends on the domain, as evident in Figure 9, where the dependency flows outwards, e.g. Infrastructure depends on all the inner concepts and the Domain, a.k.a. the Entities, Contexts, Aggregates are encapsulated.

Domain Driven Design has a high entry cost that reduces over time because of the software's increased maintainability, scalability and testability (Vlahovic [no date]). As this project's outcome is relatively small and the application is developed iteratively, simplification decisions are made to avoid over-engineering.

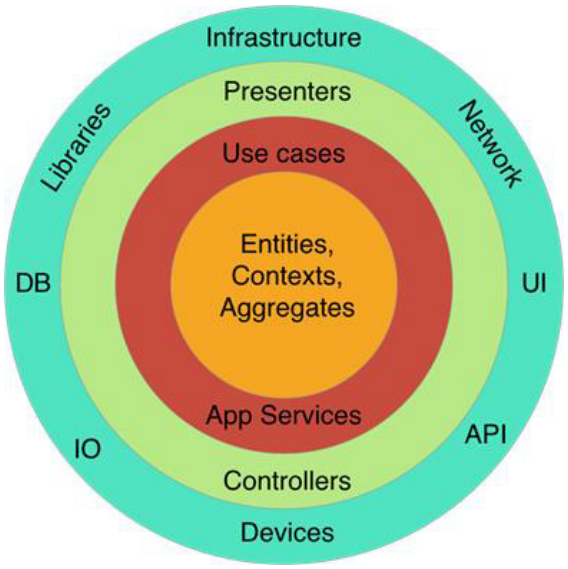


Figure 9- Domain Driven Design Layers (Dupeyrat 2022)

3.11.5. Dependency injection

Dependency injection is a design pattern that decouples an object creation from its usage. Thereby it follows SOLID's dependency inversion and single responsibility principles (Janssen 2018).

There are three types of dependency injection:

- Constructor Injection
- Property Injection
- Method Injection

This project leverages constructor injection as the most common and recommended implementation, in which the dependency is injected as a parameter to the constructor.

3.11.6. Repository design pattern

Repository pattern enhances testability and abstracts the technical database implementation from the business logic (Cubet 2015). The repository pattern is a mediator between the domain logic and the database implementation. This is achieved by creating an interface that defines the DB operations required by the business. After that, the interface is implemented with the concrete DB implementation, e.g. SQL Server. The object is injected via dependency injection into the class that will use it, e.g. a controller.

3.11.7. DTO pattern

The Data Transfer Object (DTO) pattern helps to decouple domain models from the presentation layer. It also aims to encapsulate logic that belongs with specific technology implementation from the domain, for example, how the data is saved in a

particular data storge. Thereby, DTOs are used to store, serialise and parse data and do not implement any business logic (baeldung 2021).

3.11.8. API Gateway pattern

The Application Programming Interface (API) Gateway pattern provides a single API entry point for different types of client applications (e.g. mobile, web) and services to connect. Upon request from the client, the API coordinates which services it should be sent to and is used in microservice architecture so that multiple services can be accessed via a single gateway (entry point) (Microservices Pattern: API gateway pattern. [no date]).

3.12. Software testing

Testing is the process of verifying the application correctness and expected behaviour (IBM [no date]). Quality control is important to fix defects before the software goes into production and makes applications more robust, reliable and maintainable (Sharma 2022). There are different types of software tests such as Unit tests, Integration tests, end to end tests, manual tests, performance tests and load tests.

Test coverage “measures how much tests are covering things like test requirements, code, different user scenarios, and platforms such as mobile devices or browsers” (Kinsbruner [no date]). Test coverage is measured by “coverage metric”. Some are line coverage or how many lines of the code are tested and branch coverage, defining how many outcomes are tested

Whilst a lot of high-scoring test coverage metrics contribute towards software quality, a common mistake is to use line and branch coverage on its own. To avoid that Khorikov (2020) recommends to measure quality of testing based on verified application behaviours.

4. Primary research

Computer Science students' job-search journey is essential for achieving the aim of this project. Hence, a survey is conducted amongst last and second-year Computer Science students that:

- evaluates how many of the responded students have received a job offer
- explores patterns in students' behaviour depending on their employment status
- investigates time frames and frequency of applications between those two groups
- uncovers how students prioritise employability skills, mentioned in the LR section
- determines the desired specifications for the system, according to the students

The survey is completely anonymous, and an ethics review is conducted and approved by the COMSC office at Cardiff University. The complete survey and results and the ethics review form are available in Appendix 1 - Ethical Review Approval and Appendix 2 – Full Survey Results.

4.1. Questionnaire Design

The chosen method for conducting the survey is an online self-administered questionnaire. It allows for more flexibility as it can reach more respondents at once. This also makes it easier to preserve anonymity.

The questionnaire splits the students into two categories – ones that have landed a job and others that are still looking – to observe if there are any patterns in the attitude of those groups towards application preparation and practising different skills. Furthermore, the study assesses how long it takes and how many applications on

average one needs to get a job so that it can estimate for what period students will be using the application and evaluate the difficulty in joining the job market after university. There is a series of questions assessing students' attitudes towards

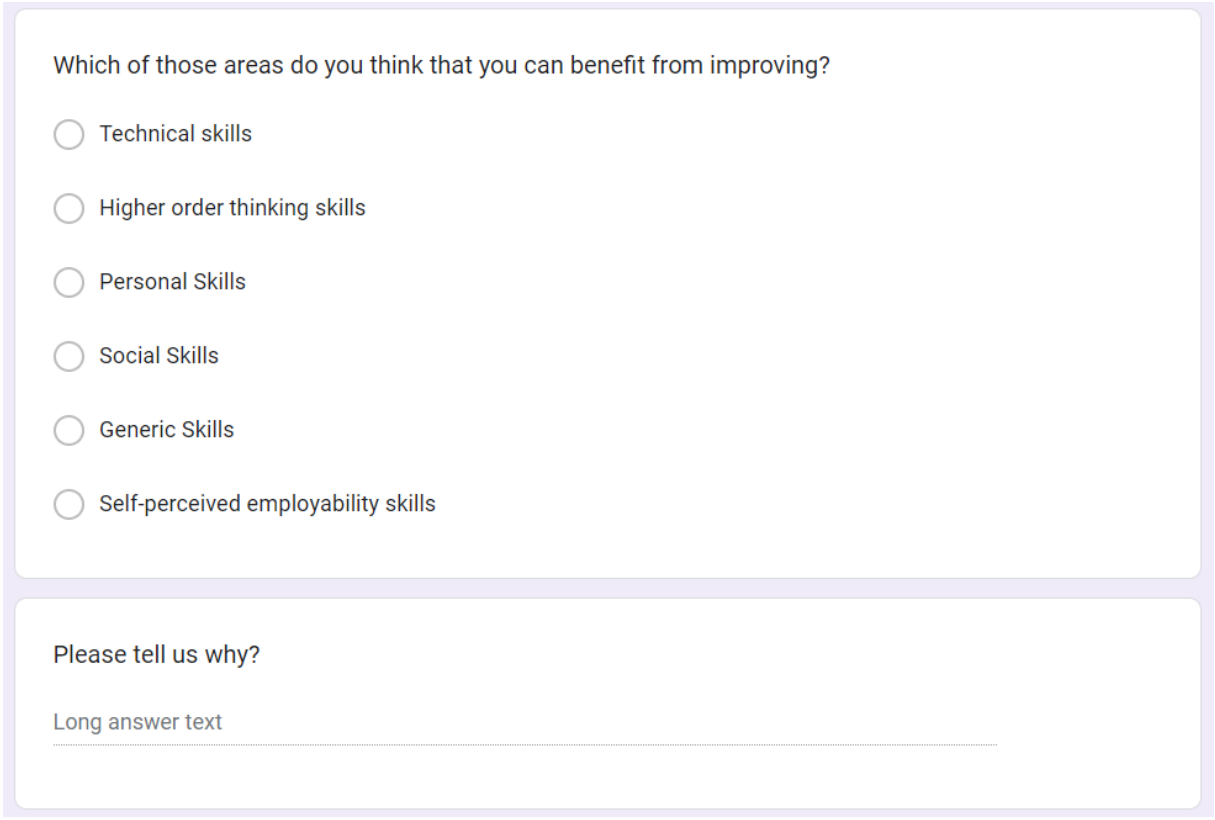


Figure 10 - Snippet from the quiz students were given

employability skills. Including closed-end provides the means to categorise the information easily and open-end questions give an inside of the individual's circumstances, considered in employability theories. In Figure 10 the first question provides an uncomplicated way to assess the students' focus when preparing for the job market, whereas justifying their choice gives a way to personalise this and give context to the data.

Finally, students are asked to share their interview preparation and what application functionality they would benefit from during the process. The given list of features is devised based on the discussion in the Literature Review. This Question is used to prioritise the specifications for a minimum viable product (MVP). The survey employed

Misra's definition of Soft skills as the most concise option and the respondents will not be overwhelmed by the number of options, providing a more genuine response.

4.2. Questionnaire Results

The survey is completed by 26 graduate and placement students, 84% looking for job opportunities, making the application highly relevant. Most students have applied for approximately 1-20 positions, with the ones who have not received an offer applying for the shortest time of 1-3 months. A third of the graduates that received an offer within 5-7 months after their first application and most of the graduates landed an offer after at least 3 months of applying (Figure 11).

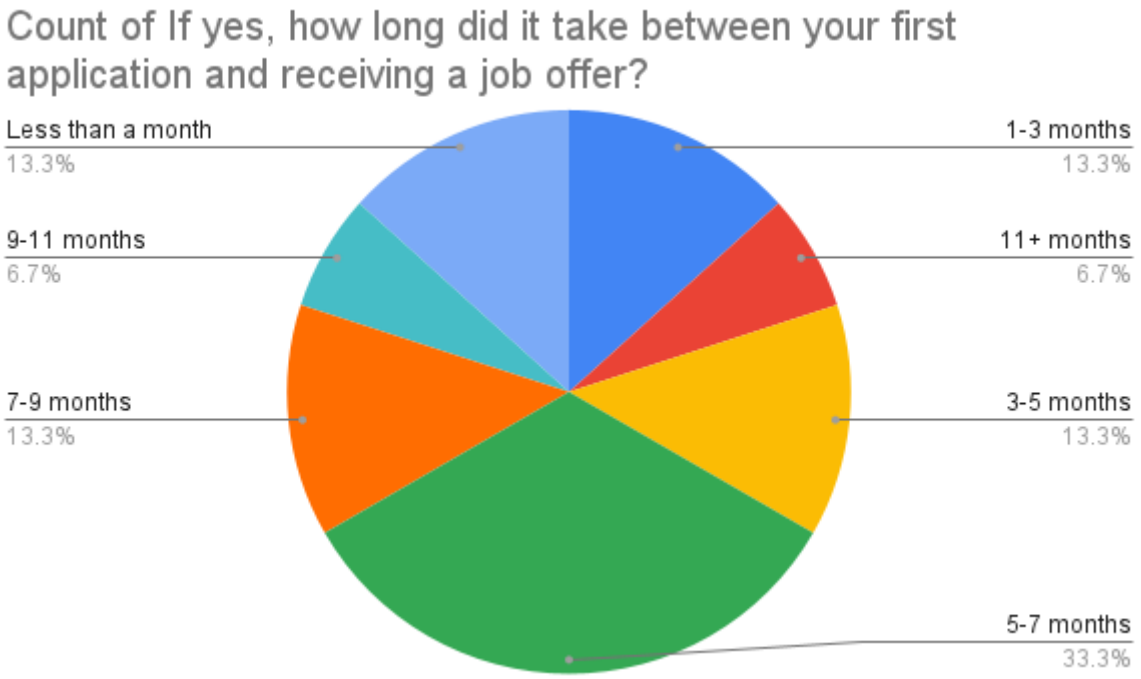


Figure 11 - Time that took students between their first application and receiving an offer

These results make the competency-based interview practice extremely important in that timeframe, being the most used approach for interviews; therefore, the website is relevant to the students' needs. To further improve this, the interview practice materials should be enough for 3-5 months of training to keep users engaged and

actively practising. A too short learning programme will lead students to abandon the application quickly. In contrast, a too-long one may create cognitive overload and discourage students from completing it, as discussed in the Literature Review.

When voting about the different types of employability skills and their importance, self-perceived employability skills have been voted least significant, whereas the rest of the skills are very similar in their scores. Nevertheless, when asked to self-assess what they could improve more than half of the graduates said Technical skills and none said Generic Skills defined as Leadership, Teamwork, Project Management and Oral Communication skills.

This observation could be on account of two causes. The first one is reinforcing the gap in expectations between employers and graduate students, where employers are looking for strong soft skills that can even make up for the lack of some technical skills and students are only focusing on technical skills. A second justification could be that employers and universities are not clearly explaining their criteria to graduates. Changing big institutions would be more complex and not as straightforward to achieve because there are more factors beyond the personal willingness of a student to learn soft skills. Those two rationales can co-exist; however, this study will focus on educating the students and leading them in the right direction.

Choosing technical skills as most important are various is based on various reasons. Many graduates feel they have not learned enough during their degree, which could be explained by the vast pool of technical skills required by different companies, making students self-conscious about their coding skills. Furthermore, some students reply that they are not well equipped for coding interviews, which could sometimes be a part of the application process. Some respondents mention that they are on a

conversion degree from different disciplines. Thus, they have learned soft skills in their previous career path and are confident in them. This reinforces the idea that employability skills could be taught from having prior job roles and work experience. The students, who believe that their technical skills are insufficient, have also predominantly focused on CV writing and programming for their preparation.In contrast, the rest of the students worked on a wider variety of skills, including CV writing and coding skills but also did interview preparation, behavioural tests or used real-life interviews as practice. Therefore, it could be argued that not many graduates will be interested in improving their soft skills, despite having a platform for practising them. Therefore, it is essential to advocate and highlight the importance of the product of this study. Universities or career services in those institutions can help promote the product. Furthermore, it is important also to make the platform easily accessible. No correlation is found within either of the groups to reinforce that either approach is better for receiving a job offer.

Finally, students are required to assess how useful different features would be for them in an application to improve their soft skills. Figure 12 shows the 3 features most wanted by the respondents. Features wanted by less than half of the respondents are not going to be included in the final product specification.

The most wanted feature is the ability to find and save different interview questions. 84.6% of students voted for it, making it a must for the MVP:

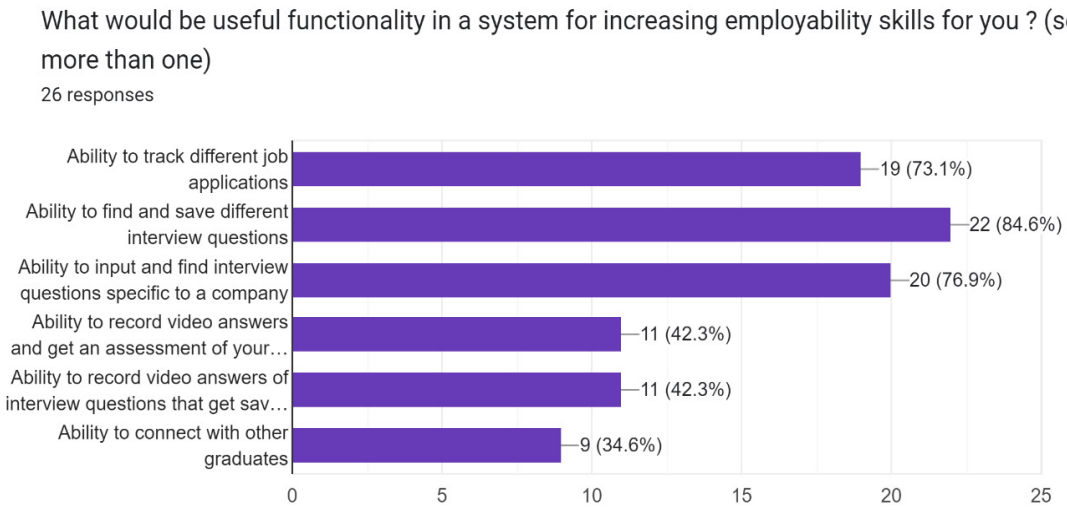


Figure 12 - Students preferences for application functionality, snippet from survey results

The second and third voted features (76.9% & 73.1%) are considered an extension of this project as increasing the scope of the MVP would affect the dissertation timeframes. Thereby they are not part of the implementation of the product but are considered as part of the design and future work. Furthermore, only the interview practice feature affect graduates' soft skills learning. Instead, the others("Ability to input and find interview questions specific to a company") give the tools for more targeted preparation and ensure time management regarding job applications.

In addition, the functionality to input and find different interview questions duplicates Glassdoor, which follows a strategy where people must share interview questions to view others' input on the platform. In this way, they have created a data resource of interview questions, salaries, jobs and employer information. Similar approach could be taken here; however, with having such a big competitor on the market, functionality is considered out of scope for MVP.

5. Product Justification and Specifications

5.1. Functional Requirements

This section aims to finalise the primary and secondary research to utilise the main conclusions for devising a set of functional and non-functional requirements for the system. It serves as a transition to the next Chapter that discusses the product and the approach to developing it.

The Literature review discusses the gap between student's and employer's expectations, where it uncovered that:

- Students lack soft skills and the ability to elaborate on them.
- Universities cannot teach every technology; thus, graduates do not feel prepared for the market.
- Graduates often require previous experience to get any work experience and feel overwhelmed.

This study confronts the lack of soft skills problem and devises a plan to contribute to solving it by:

- Exploring the topic of employability and graduate employability,
- Defining the most used processes to target during the application process, investigating similar applications
- Deducting appropriate approaches for creating one.

As reported by the Evaluation of Existing applications, it is considered most appropriate to use the web as a platform for this project.

According to the Literature Review, employability can be split into external and individual factors and for computer science, the external circumstances are in favour

of the students. Thus, an improvement is required only within the individual skills, classified as soft and technical skills. The survey uncovered that most respondents are actively practising coding tests and technical skills; thus, they are already dedicating the required effort to technical skills. Furthermore, it is identified that soft skills are highly required by employers in all industries and are considered even more valuable for new joiners on the job market.

Companies use competency-based and behavioural interviews to assess those skills; thus, an application to aid the learning is a solution to the “gap in expectations”. Furthermore, the primary research proved that an application to save and answer interview questions is the most desired functionality amongst the respondents. Therefore, the first user stories for the application could be defined as:

1. As a **user** I want to have access to, save and answer interview questions **so that** I can organise my competency-based interview preparation.
2. As a **user** I want to have a profile where I view the questions I have responded to and the questions I wish to respond to **so that** I have easy access to the application's resources.
3. As a **user** I want to edit my interview questions responses **so that** I can organise my competency-based interview preparation.
4. As a **user** I want to view example answers and prompts that will show me how to answer specific interview question **so that** my learning is guided, and I can meet employers’ expectations.

Soft Skills are split into categories, which one could have unbalanced abilities, as discussed previously. Some soft skills provide more training opportunities than others.

Therefore, it is appropriate to split the questions into categories, called “Topics”, depending on their targeted skill:

5. As a **user** I want the questions to be split in topics **so that** I exercise my less advanced soft skills to improve my employability skills.

The topics that the application will implement in MVP are Collaboration, Problem Solving, Adaptability, Organisation, General. Topics could be changed and added to in later stages of the project.

With these five user stories the essential functionality of MVP is defined.

Optional functionalities for later stages of the project, based on the survey results are:

6. As a **user** I want to receive trophies and prises for completing questions **so that** I am motivated to continue my learning.
7. As a **user** I want to be able to relate interview questions to companies and filter by company **so that** my learning is targeted to the specific interview I am preparing for.
8. As a **user** I want to track my job applications **so that** I can maintain good organisation while applying for placement/graduate opportunities.

All the optional requirements take an example from Duolingo, where learners are given more functionality and instant gratification to keep returning to the application. Furthermore, user stories 7 and 8 are the second most wanted amongst graduates and are aimed to either create more targeted interview preparation, in the case of story 7, or better time management and organisation, which allows more time to be focused on interview preparation for story 8.

5.2. Non-functional Requirements

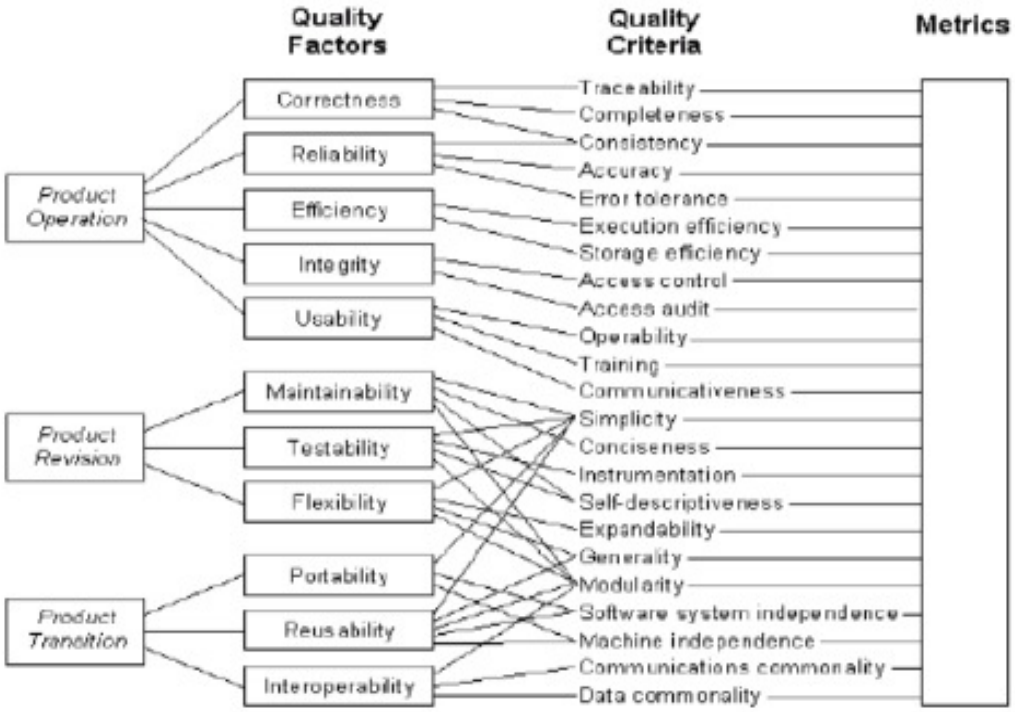


Figure 13 - Software Quality Factors (McCall,1977)

With the user stories being defined, the non-functional requirements could be addressed. Non-functional requirements help satisfy software quality factors (Figure 13) by measuring and satisfying the Software Quality Criteria (McCall's Quality Model. 2020) along with the system's operational bound.

Security:

- The user interface (web application) should keep track of user authentication and authorisation status before rendering any content. Therefore, the pages should be private by default and only accessible to authenticated users.
- Limited API Surface Attack could be achieved by limiting the data exposed in the backend APIs. Thus, the data should not need to be hidden in the user interface.
- Database enforces authentication and authorisation via username and password; therefore, every client must attach the credentials upon opening a connection. The backend service should be the only software component with the credentials to connect to the database. The credentials are managed as environment variables and unavailable in the source code.

Capacity and Scalability: The application should implement asynchronous programming to avoid blocking threads during I/O requests, achieve concurrency and remove the thread per request capacity ceiling. In addition, the application should enable horizontal scaling with multiple deployments.

Maintainability and Testability: When justified, the application must implement design pattens and software development good practices to increase the system's maintainability. In terms of testability test and code coverage are considered. The application should have 100% test coverage and over 70% code coverage in Unit Testing. The application should be tested end-to-end.

6. Approach

This section discusses the chosen approach to fulfil the objectives and satisfy the aim of this project. The methodology is devised for an individual project instead of a team one and uses elements of two Agile methodologies, Scrum and Kanban, adapted for more efficient work outside a team. Furthermore, it follows the software development lifecycle principles explored in the Approach section. It is preferred to use an Agile methodology over waterfall due to its focus on continuous feedback and self-reflection, a requirement for the dissertation project. Moreover, it allows for an exploratory approach to requirements and guarantees working software early in the development process. Nevertheless, the downsides of Agile are considered. Having this report as documentation of the process partially eliminates Agile's lack of documentation problem for the planning and design phase.

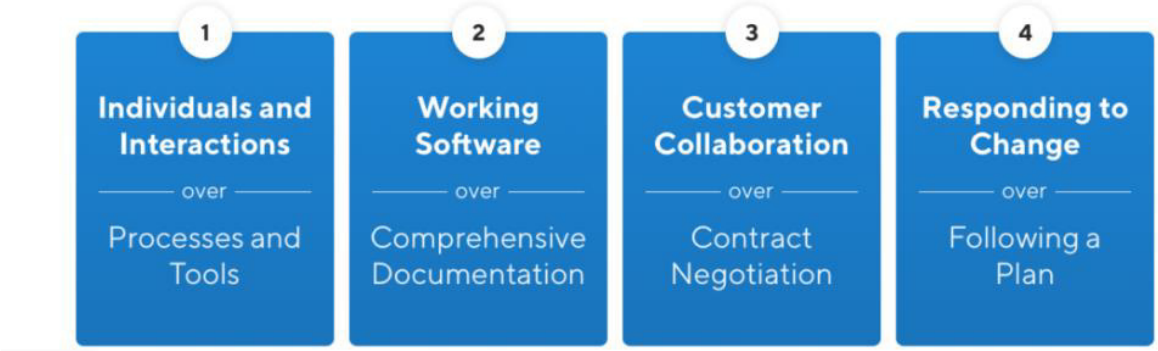


Figure 14 - Agile values (McCloskey [no date])

Furthermore, Jira software discussed later in the chapter serves as a chronological history of the implementation phase of the project and the priority of different tasks. In addition, Git, a version control software, is used during the implementation phase, thus allowing documentation of different states of the code. A disadvantage of using Agile is the difficulty in estimating the delivery date to match the deadline for the master's degree. How this challenge is overcome is discussed later in this chapter.

The project focuses only on three of the four Agile values (Figure 14), excluding the "Customer Collaboration" one as this is not applicable for this project where the researcher takes on all the roles during the software development lifecycle.

The first value is prioritising individuals and interactions over processes and tools. Since the team aspect does not apply to this project, it could simply be reshaped as prioritising the project's progress over processes and tools. Thus, a minimalistic methodology and project management techniques are used to achieve the objectives. A combination between Scrum and Kanban is used to ensure continuous progression. For this purpose, a project management software Jira is used. Tasks, called stories, are written into a product backlog and moved through a workflow (Appendix 3 – Jira Issues). The workflow consists of four stages:

- **To do:** every story has clear objectives and acceptance criteria; it is estimated and prioritised accordingly.
- **In Progress:** A story that is currently being completed. No more than 4 stories at the time are put in progress.
- **Testing:** Unit testing and manual testing is done for each story
- **Done:** A story is moved into "Done" once all acceptance criteria are satisfied. Those stories are archived to serve as documentation of the project.

All stories are prioritised and estimated in story points, using Fibonacci numbers. A story point is a unit of work representing the overall effort required for a story to be implemented (Atlassian [no date][a]).

The second value is creating working software over comprehensive documentation. This project has only one contributor; thus, there should be just enough but not

excessive documentation for them to understand their motives. This report and the completed stories are used as documentation of the project.

The third value is “Responding to Change over following a plan”. Scrum ceremonies are implemented to satisfy this and achieve maximum flexibility. For this project, a retrospective is done approximately every month. There the work approach is evaluated consistently. Moreover, a refinement session, where the new stories are estimated and prioritised, is scheduled weekly.

Nevertheless, these timeframes are flexible. Sessions are scheduled depending on the project's current needs and can occur more often. Sprints are not strictly implemented so that the project could benefit from just enough project management without overwhelming it with too many processes.

The MVP user stories are broken down even more into epics and then further into tasks. New tasks are added and removed at any time to maintain flexibility in the design requirements as the project evolves, more similarly to Kanban than Scrum.

7. Product

After completing the SDLC planning phase in the prior sections, the next phases are discussed in this section. It is important to note that these are executed non-sequentially and iteratively, as discussed in the Approach section. To achieve better structure and clarity they are discussed in sequential order.

All the code referenced in this section and a demo video is available in the submission or on the following links: code and demo video.

7.1. Design

This application follows a three-tier architecture model where the concerns are separated:

- **Presentation tier:** The web client application that is solely responsible for what the user sees and how they interact with the website
- **Application tier:** The backend application contains the controllers, models and the domain logic. This tier integrates with the Presentation tier (the frontend) through Hypertext Transfer Protocol (HTTP) requests and leverages the JavaScript Object Notation (JSON) data standard
- **Data tier:** The data storage is externally managed by a MongoDB database

7.1.1. Backend Design

Strategic design

The first step in the application design is to gather all the available knowledge to define the domain. There are two bounded contexts for the extended version of the application based on the primary and secondary research: “Competency-based interview preparation” and “Application Tracking”. The Competency-based interview preparation context has two sub-contexts (subdomains): “Competency-based interview practice” and “Company intelligence” that correspond to the features requested in the questionnaire (Figure 15).

The focus of this report and the MVP is to develop the functionality based on **Competency-based interview practice**.

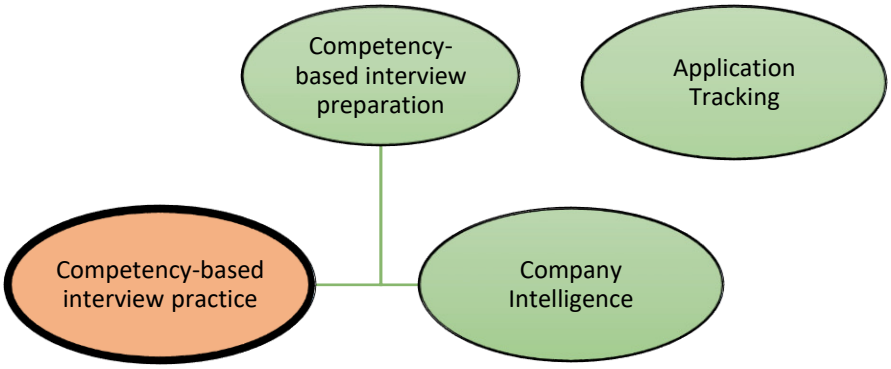


Figure 15 - High-level context map of the domains

Tactical design

As mentioned during the tactical design, additional granularity is provided to the schema, making it easier to transfer into code. Three primary root aggregates are identified within the two main domains and two subdomains: Interview Question, User Identity and Application (Figure 16).

I. Interview Question

The interview question is an aggregate displayed to the user in the UI. It consists of an id and a title (the question) and encapsulates three Value Objects – Topic, Example Answer and Prompt (those objects are valuable for their data, not identity). It is modelled to satisfy the business criteria of supplemented learning applications and the requirements defined and justified in section 5. This aggregate is the shared kernel between the Company Intelligence and Interview Practice domains.

II. User Identity

User Identity is a shared kernel between the Interview Practice and Application tracking domains that allows the application to identify the user so that they can

interact with the application and gives the means to satisfy the “Protects user’s personal information” objective. It consists of two entities:

- User Profile: used to interact with Applications and Interview Questions and encapsulate the user in the context of the application
- User Personal Data (Identity): This context is all the personal information required to identify the user, currently the value objects Email and Password.

The implementation of using two entities is a design decision that also allowed the scalability of authentication. For example, using OAuth with Google or Facebook will only give a token and a user id that could be easily related to a User Profile.

III. (Job) Application

This aggregate allows a user to create an application for a company and add scheduled Application Events such as interviews and assessments to achieve the desired Application Tracking functionality. Application Event is a value object because an event for the same Application.

As a last step of the tactical design, the relationships between domain objects are established and the model is finalised for implementation (Figure 16).

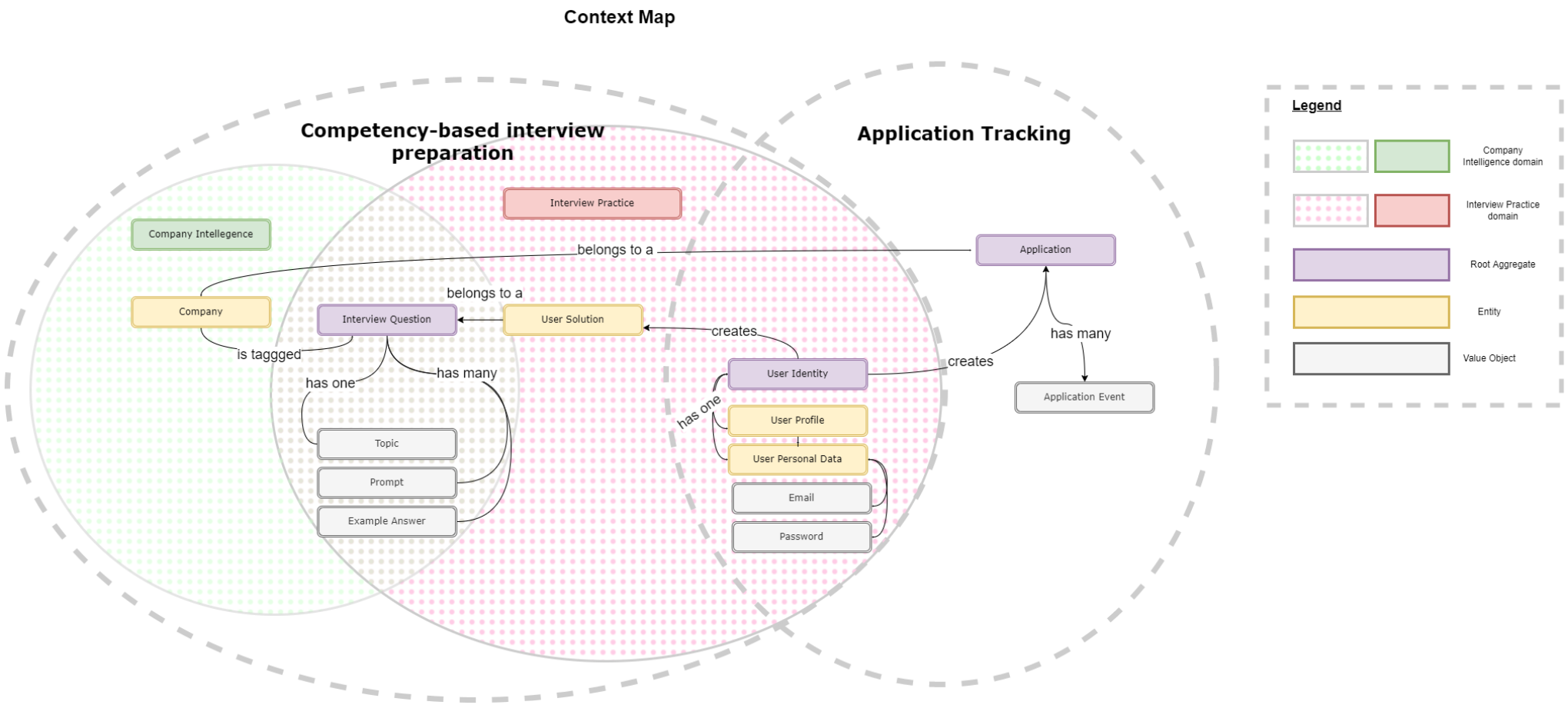


Figure 16 - System Context Map

7.1.2. Frontend design

As mentioned in the Literature Review section, visual design and ease of use are essential for e-learning applications. Reducing cognitive load and creating a non-judgmental and approachable learning environment is most important for its success. The design process began by outlining the wireframes for the application and deciding on the navigation structure of the MVP with 5 pages:

Landing Page: It is the application's starting point from where the user can navigate to other pages (Figure 18 and Figure 17**)**. Text here is minimal, and the user will be presented with important information about why the application is valuable. The Navigation bar is persisted across all other pages.

Profile Page: This page will display the user's responses and favourite interview questions. Users can see their trophies for completed lessons and learning progress along with their applications and favourite companies in the future. Naturally, this page is available only for registered and logged-in users.

Questions Library: This page is the hub for all the core functionality of the Competency-Based interview preparation domain. The low fidelity wireframe shows how the list of questions is arranged, see Figure 20. This page has two states: for a guest user or a logged-in user. Guest users will only be allowed to view the questions and they will not be able to click, answer a question or favourite it. Authenticated users can click on the questions, answer them and mark them as favourites, which is visible in the navigation bar and questions card displayed in Figure 22. Authenticated uses can also filter the questions by topic.

Individual Question Page

Once a question card is clicked the user is redirected to the Individual Question page where users have a workspace to answer the question, see Figure 21. All application pages are available here via the navigation bar. Moreover, one can switch to the following question in the currently opened list by clicking on the arrows in the navigation bar. The order from the question list on the previous page is persisted for clarity. For example, selecting a question from the topic “General” will result in the whole list of General questions being displayed in the same order as they appear in the Question Library. This is presented in the demo video.

Log in and Registration

Log in and Registration pages contain forms with the relevant information required. Both frontend and backend validation is required to prevent attacks and ensure data consistency.

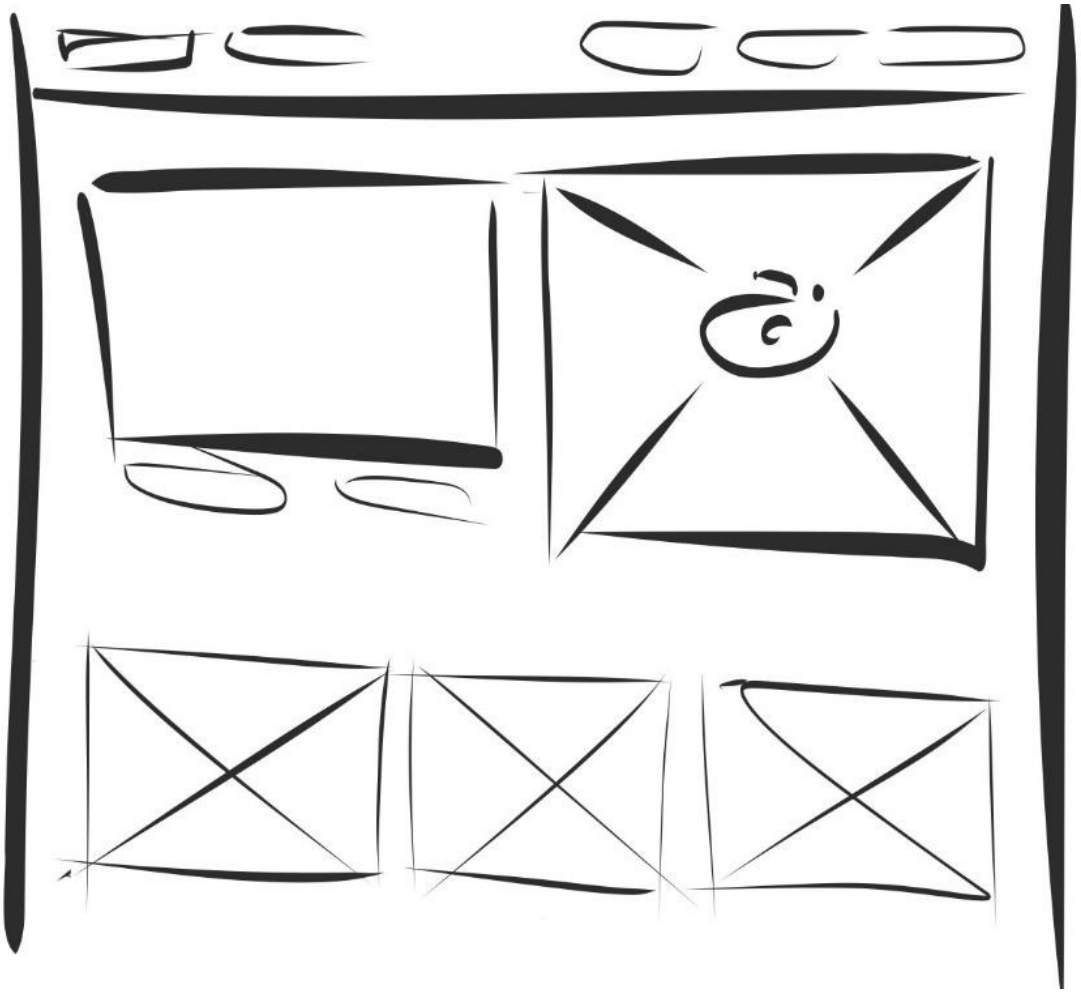


Figure 18 - Landing page low fidelity wireframe

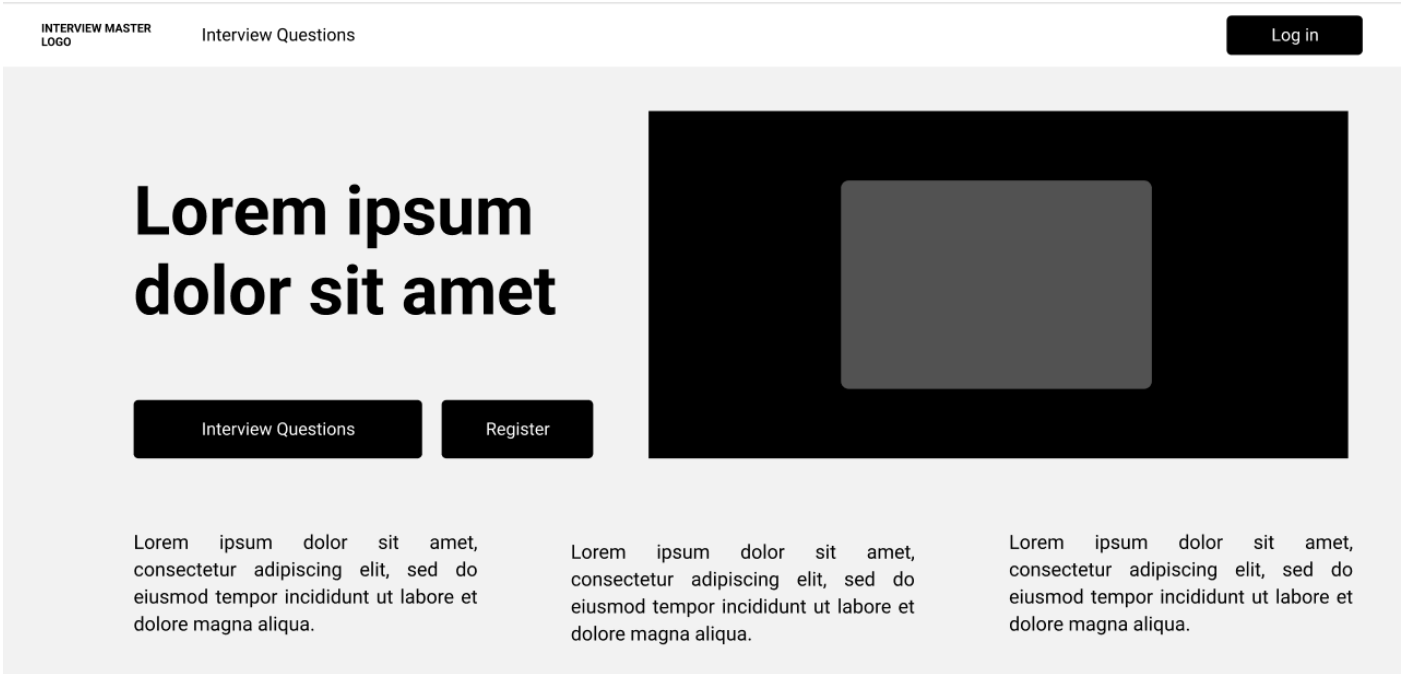


Figure 17 - Landing Page high fidelity wireframe

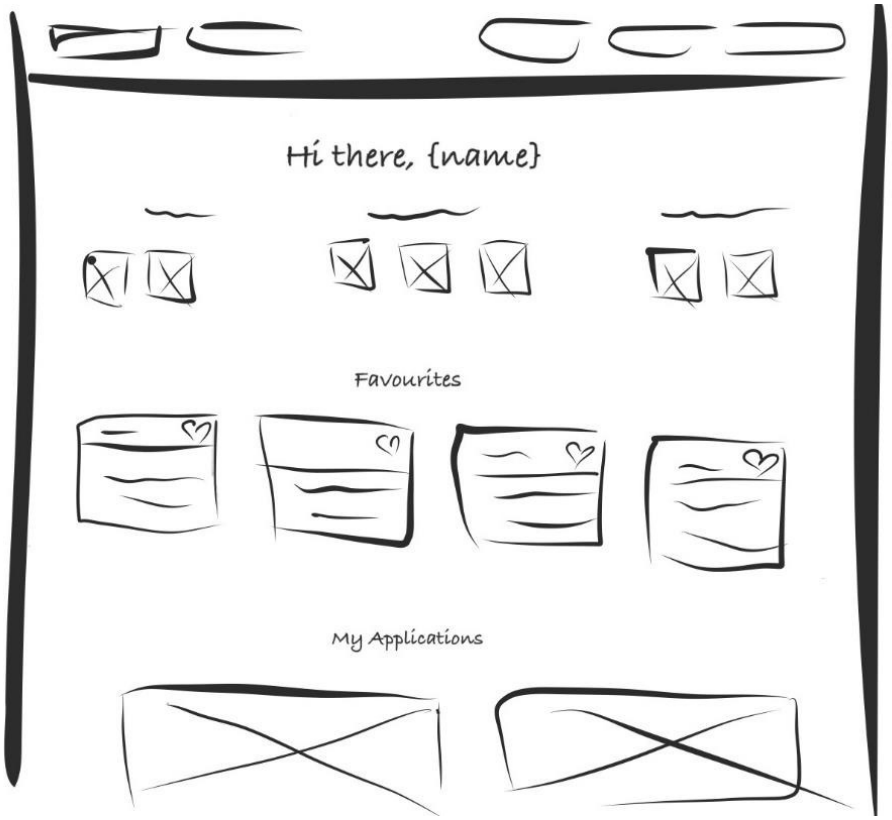


Figure 19 - Profile page low fidelity wireframe

Figure 20 -Questions library low fidelity wireframe

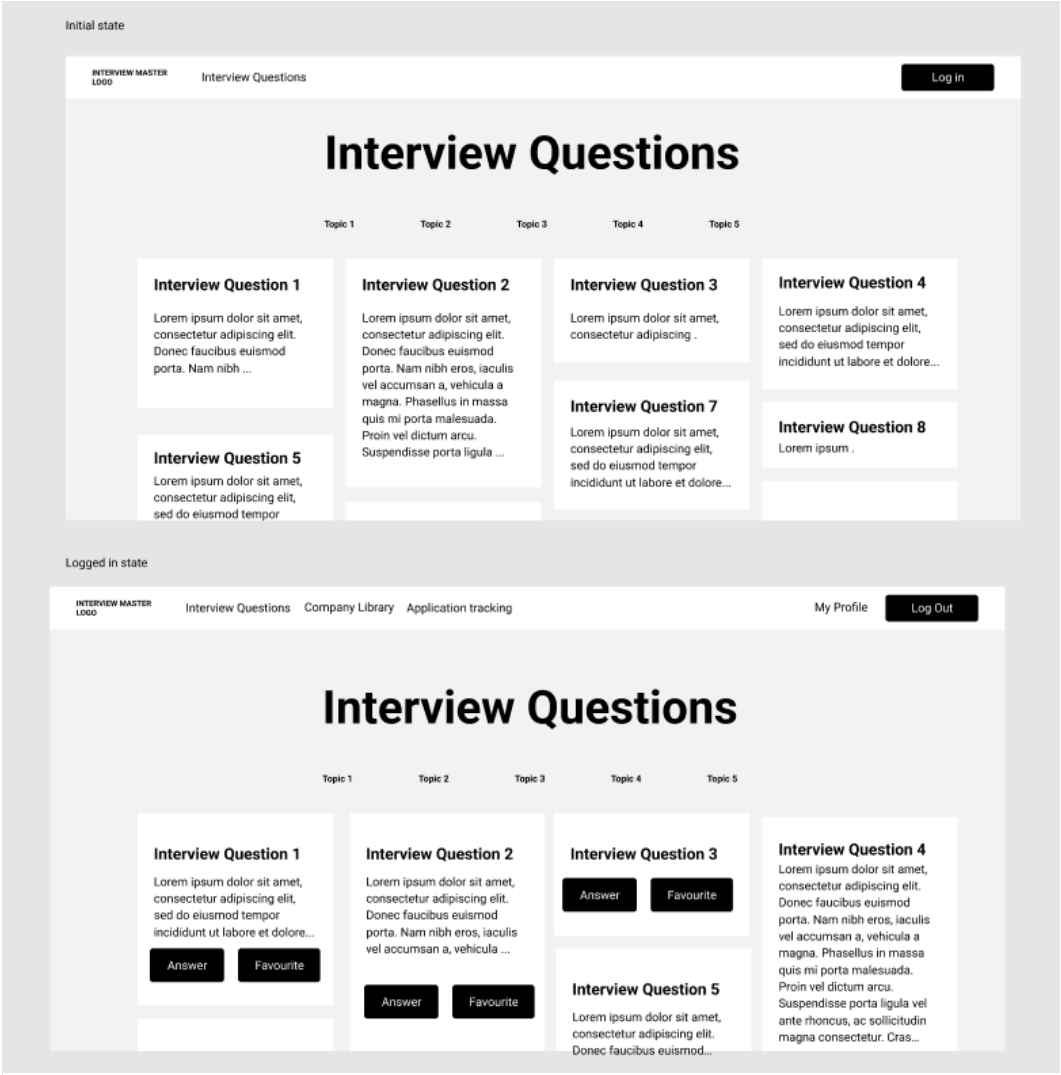


Figure 22 - Questions Library high fidelity wireframe

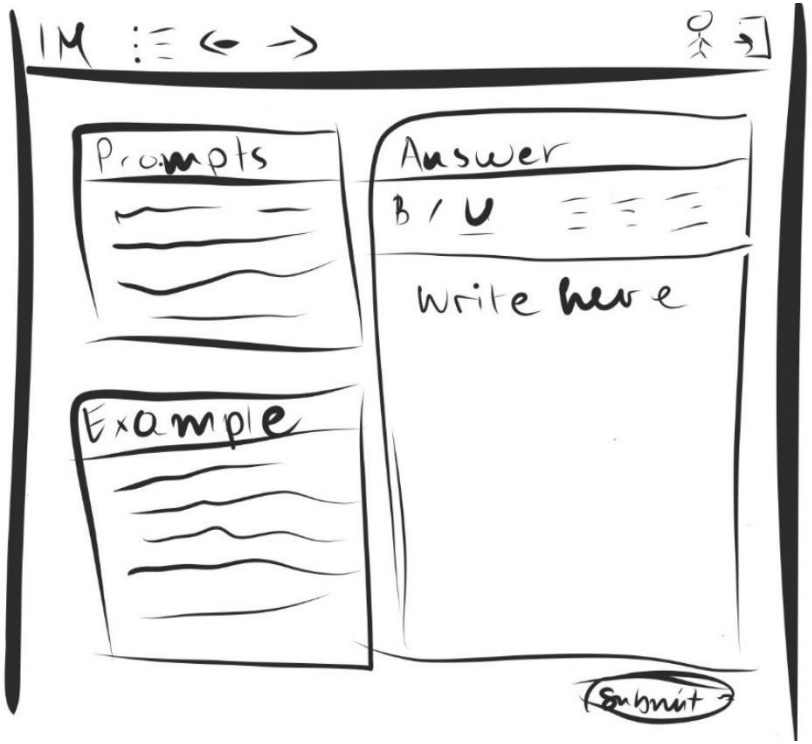


Figure 21 - Individual Question page low fidelity wireframe

Visual Design

The application's visual design is established after sufficient clarity is achieved for the UX design.

Figure 23 - Color palette

Firstly, a colour palette with bright and contrasting colours is selected, see Figure 23. This palette helps people with visual impairments to see the website. Furthermore, a dark theme is used so that the website is less stressful in the eyes and easier to use for extended periods (Figure 24). The gamut from blue, purple and pink is chosen because it implies professionalism and creativity (blue and purple) but also safety and harmony (pink), which corresponds to the idea of creating a safe environment to foster students' employability skills. The chosen shapes are soft and round, which, as discussed, appears more approachable. Figure 24 demonstrates this in a few of the background swatches that are created for the project.

Figure 24 - Background swatches for the website

7.2. Implementation

7.2.1. Backend

This section discusses the development of the backend application and highlights important implementation details. The application follows Object Oriented Programming, SOLID principles, and implements DTO pattern, Repository pattern and uses Dependency Injection. Asynchronous programming is used to satisfy the non-functional requirements, make the app more performant.

Technology Justification

Initially, the backend application was developed with Python and FastAPI, a web framework. Python was selected because of its fast development, the researcher's familiarity with the language and the language's popularity for open sourcing the application at a later stage. FastAPI was chosen as it has a better performance compared to other Python frameworks such as Django and Flask (FARM Stack Course - FastAPI, React, MongoDB [no date]; FastAPI. [no date]).

However, as the project grew, Python's fast development came at the expense of maintainability, scalability and code readability. As a result, the application was migrated to C# and .NET core as it provides type safety and dependency injection facilitating the DDD implementation. The benefits of choosing C# opposing to Python are:

- Performance: C# has better performance than Python as it compiled to an Intermediate Language (IL) to achieve multiplatform support and then compiled to machine code.
- Ease in implementing OOP principles with interfaces, inheritance and type safety.

- C# has Dependency Injection, which eases adopting SOLID principles, whereas python does not provide this out of the box.
- Both C# and Python support Object Oriented Programming. However, in Python, procedural programming is also supported; therefore, it gives the flexibility to break OOP principles and start writing less maintainable scripts. C# enforces OOP, thus eliminating this risk (C# vs Python: Head to Head Comparison [Updated]. 2022).
- C# has better code readability and more consistent syntax than Python, making it more readable and easier to maintain.
- .NET Core framework is flexible, lightweight, and not platform-specific like the pure .NET framework. NuGet packages add extra functionality to the .NET Core framework (Microsoft 2022).

Given that C# results in longer development times, the scope of the MVP was reduced to exclude the Job Application Tracking Functionality that was included in the first phase of design.

The proof of concept developed in Python is beneficial as it aided the design process. In addition, the Frontend application was subject to minor changes during the C# migration due to the low coupling provided by the API contracts. This contributed to further refinements of the Context Map during the Design. As a reference, the Python code is available in the repository:

https://github.com/MagdalenaVelkova/InterviewMaster-Legacy

Data Storage

MongoDB, a NoSQL database, was preferred over a relational database because NoSQL databases provide simplicity when programming, positively affects the

development timescales and allow for frequent application changes during the Agile development process. Furthermore, NoSQL databases provide better scalability and fast response times; therefore, they affect the usability and efficiency of the software solution (McCall's Quality Model. 2020; NoSQL Vs SQL Databases | MongoDB. [no date]). Mongo Atlas, a cloud-based MongoDB provider, was chosen over setting up a local database as it allows for better data security and provides more reliability due to being hosted on AWS servers. The official MongoDB Driver NuGet package connects the application with the database.

Authentication

User Authentication is stateless and implemented using JSON Web Tokens (JWT) attached as a header to every network request. Unlike Stateful authentication, stateless authentication does not require the server to store any session information in memory. Hence, the backend application has better scalability as it can be scaled horizontally through multiple deployments and increases reliability as the server does not require any in-memory data storage that might corrupt in case of crashes (Dwyer 2021).

JWT is chosen as it is simple to implement and enhances security compared to token authentication methods such as Simple Web Tokens (SWT). JWT is a standard used to transmit Java Script Object Notation (JSON) objects and is usually sent in the "Authorization" header of HTTP requests (Auth0 2022). The sequence diagram in Figure 25 outlines the Authentication flows of the application for login and registration. Each JWT holds a header, payload and signature separated by dots (".") and is signed with a symmetrical key available in the backend application. The JWT payload contains the user ID in this project to facilitate the backend to manage the requests.

Another critical aspect of authentication and security is the handling of personal data. User passwords are salted and hashed with a SHA256 cryptographic hash function, as the National Institute of Standards and Technology recommends (NIST). A password salt is a randomly generated string created when the user registers and is appended to the plain text password to introduce deterministic randomness in the hashing process. The salt and hashed password are stored in the database to calculate the hash in future login attempts, protecting against rainbow attacks where hackers use pre-calculated hashes to find matching passwords (Rainbow Table Attack. 2022).

In addition, it was decided to separate personal data in an individual collection that later could become its encryption-protected database, separated from the rest of the data to directly satisfy the objective of protecting users' personal data.

This implementation satisfies the objective of protecting the user data for MVP. The "Analysis" section discusses the downsides of this implementation, and the "Future work" section continues that discussion on how it can be improved as the project matures.

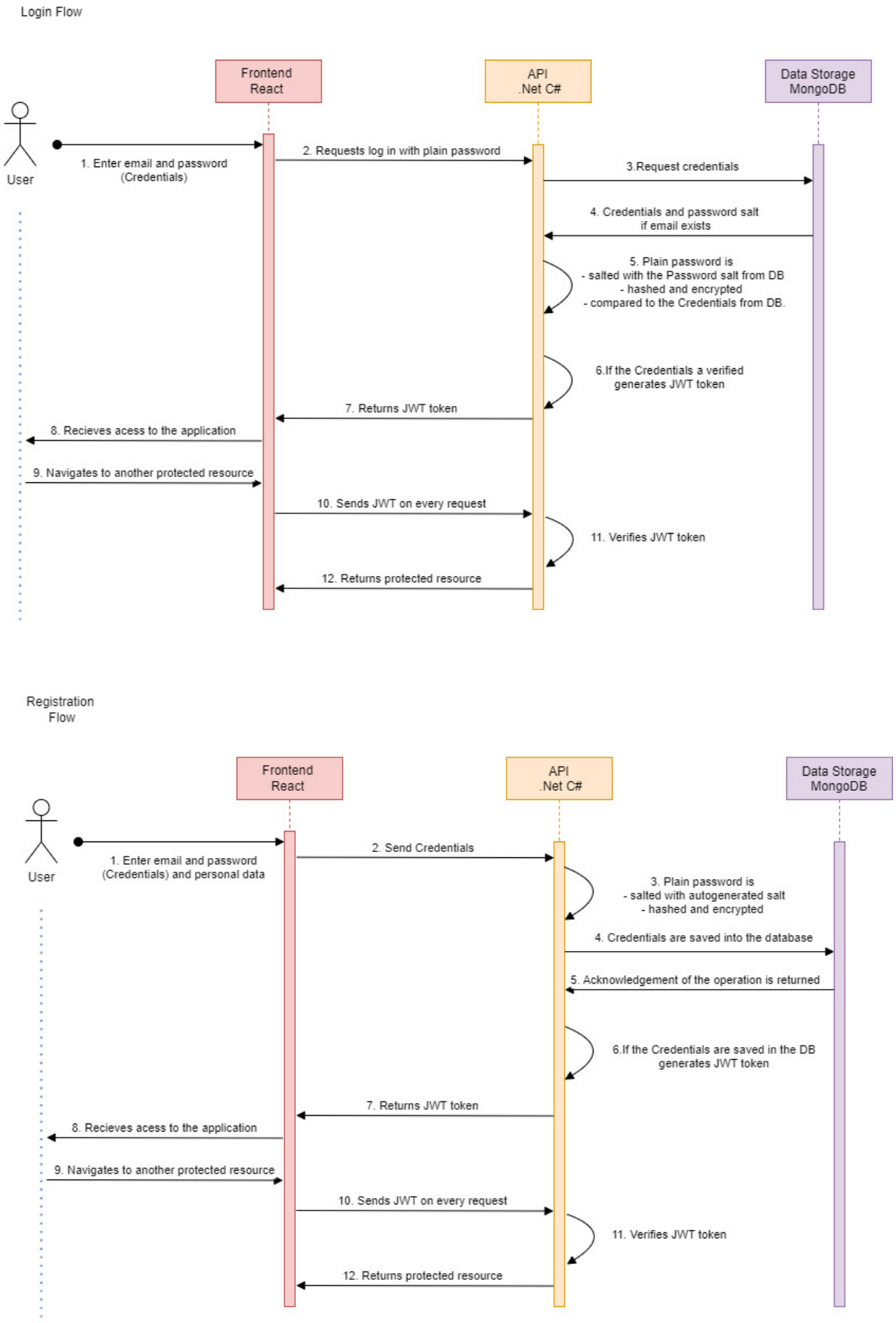


Figure 25 - Log in (top) and Registration (bottom) sequence diagrams

Backend structure

An API starter project is created and wrapped in a project solution to satisfy the DDD structure and patterns described in the Design section. After that, it is modified to satisfy the layers described in Figure 26 left, and the structure was established as shown in Figure 26 right. All outer layers (Figure 26 left) depend on the inner ones and no dependency flows in the other direction. This way, the logic belonging to the Domain and Application layers is encapsulated.

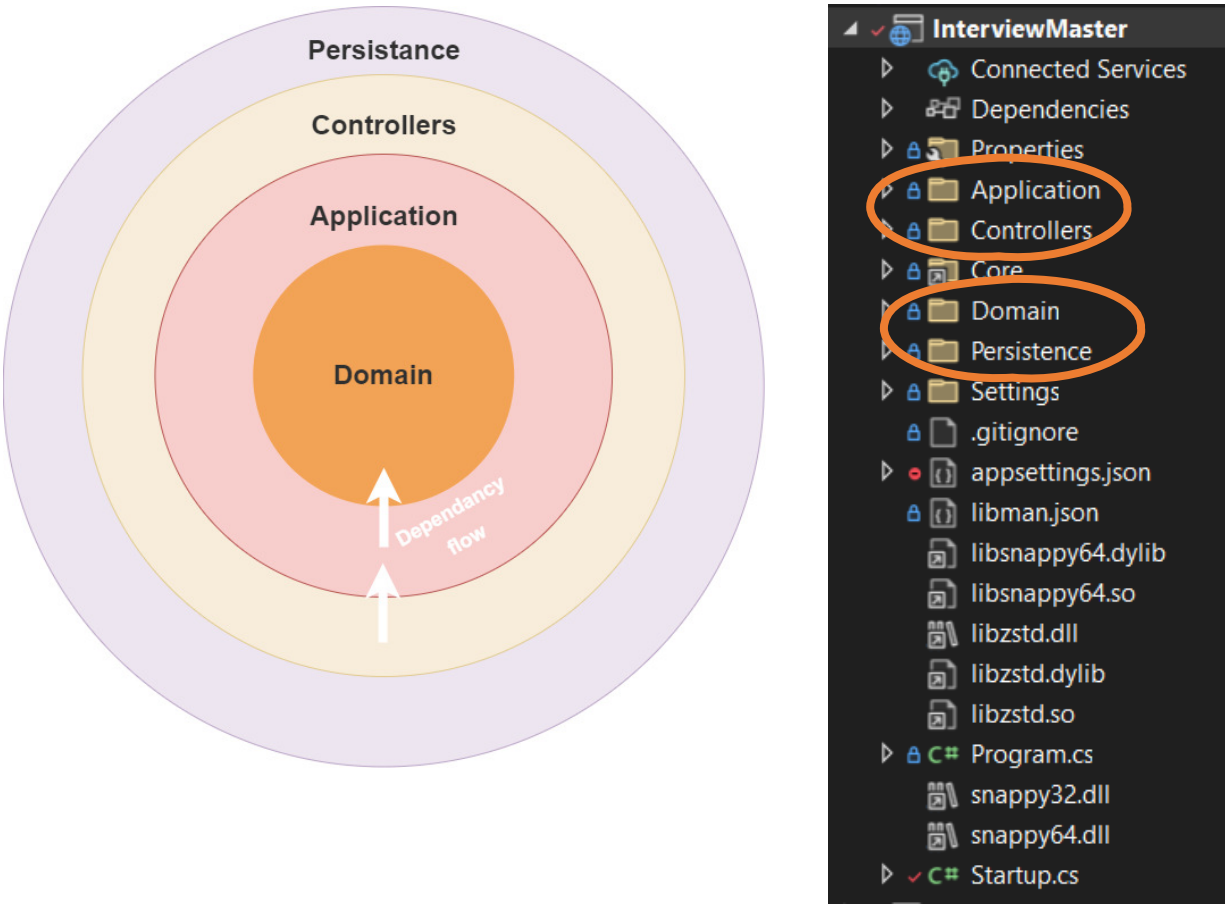


Figure 26 - API structure diagram (left), Implementation Folder Structure (right)

Domain layer

The first development phase is creating the Domain layer representing the Context Map with the User (Identity), Interview Question and User Solution (Interview Preparation) aggregates. The domain layer does not depend on any other system's layers, as illustrated in Figure 26.

The Classes and what type of domain object they represent have been specified in Table 6 where the “purpose” field describes their real-world representation and justification for the classes and demonstrates clearly how the “Single-responsibility” (**S**OLID) principle is implemented. For example, an Interview Question would only change should the business needs about the Interview Question change. In Figure 27 the domain structure of all domain objects and their properties is visible.

The User aggregate described in the previous section is not implemented as a class as it would only hold references to UserAuth and UserProfile classes, which adds unnecessary complexity to the code. Validation is added to some of the domain objects for example Topic, a value object. A Topic object cannot be created if the topic value does not exist.

The Domain implementation directly contributes to fulfilling the Functional Requirements as it creates the foundation for building the application's logic. For example, to satisfy user story 5:

As a **user** I want the questions to be split in topics so that I exercise my less advanced soft skills as a means to improve my employability skills.

The Topic value object was created and validation of its values was implemented.

Domain	Class Name	Type	Purpose
Identity	UserProfile	Entity	To hold all the information about interview Practice and other application features. It has the same id as UserAuth and together they form User aggregate*.
	UserAuth	Entity	To hold all personal information about the user such as their email and encoded password.
	Credentials	Value Object	To represent the User when logging in. A matching email-password value represents exactly one identity.
Interview Practice	InterviewQuestion	Aggregate	Combines all domain objects necessary for the concept of interview question.
	UserSolution	Aggregate	Combines a response to a question for a user so that the user can see to which questions they answered and what response they have created.
	ExampleAnswer	Value Object	The value of an example answer. Two identical example answers are considered the same
	Topic	Value Object	Enumerator of possible topic values
	Prompt	Value Object	Questions/Advice that aids the user to respond to the question e.g., “Use SMART objectives”
	Response	Value Object	A domain object that holds the value of a sponse. This object doesn’t link it to a user interview question.

Table 6 - Implementation of context map

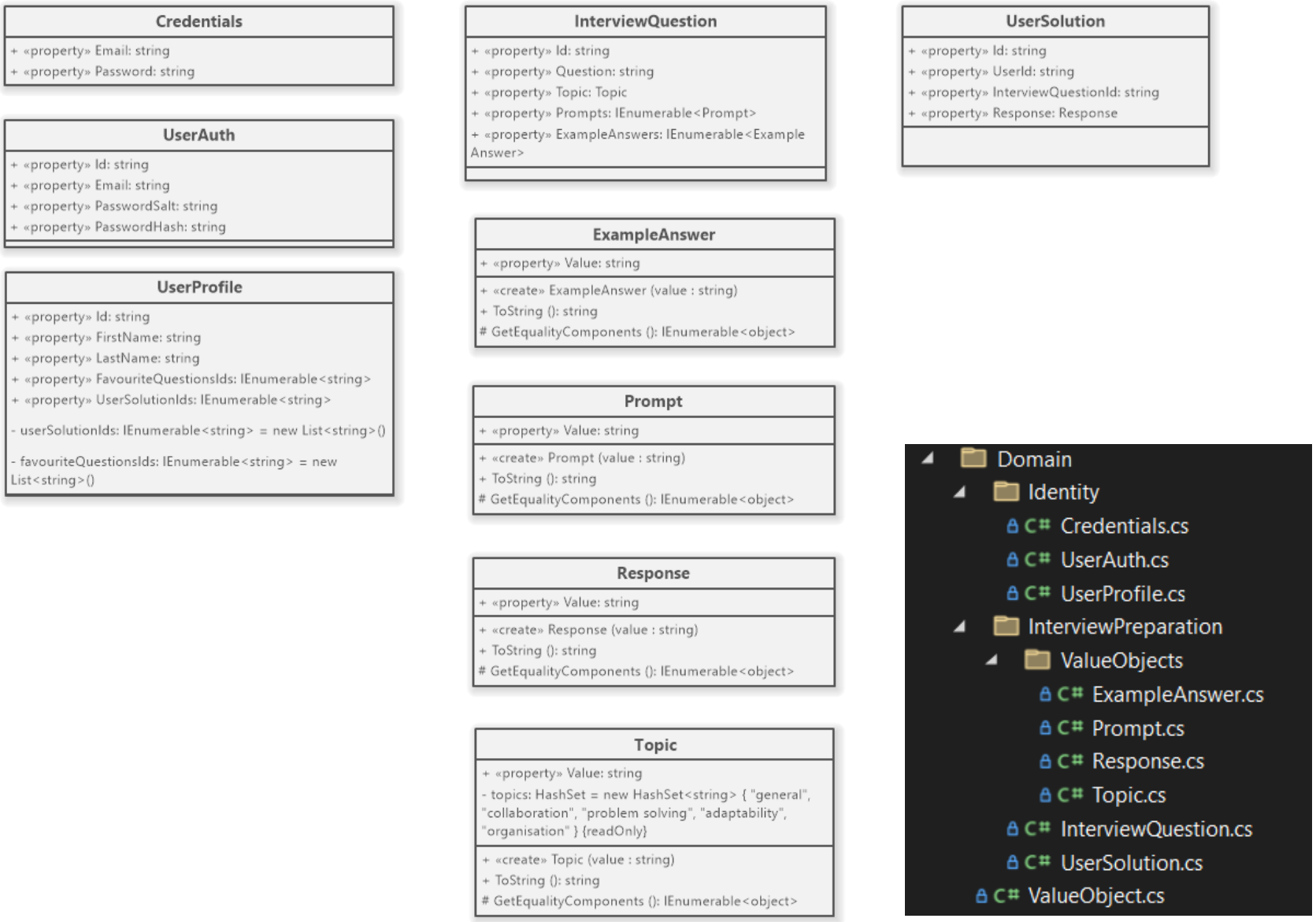


Figure 27 - Domain object implementations

Application, Controller and Persistence layers

The Application layer of the project encapsulates all logic and services. The controllers' layer defines the API endpoints acting as the entry points to the backend application. Finally, the Persistence layer is an infrastructure layer that deals with the communication with MongoDB (or any other Database).

Services are only created if there are more than two logical actions (for example, if statement or validations) in the controller, in contrast to the traditional DDD implementation. This rule is implemented to avoid over-engineering and creating single-method Services.

Application layer contains four interfaces, implementing the repository pattern and an Identity Service class. The purpose of the Identity Service class is to encapsulate all authentication logic and contains the methods for all the steps described in the sequence diagrams in the Authentication section (Figure 25). It also helps keeps the API endpoints private, satisfing the non-functional requirements. By using a service class, implementation specific details are abstracted and should a change be necessary for example for generating a different token or using a different hashing algorithm the changes are easily implemented in one file, reducing the development timeframes and increasing the readability of the code.

Dependency injection is used to inject the different classes across the application. An example of this can be seen in the Users Controller (Figure 28). Dependency injection is typically used with interfaces, however it was implemented for a concrete class IdentityService in this case. This contributes to the testability of the project as the class could be later injected in the Unit tests.

This abstraction of the database implementation into an interface and injecting it into the controller is another way of implementing dependency injection and is called Repository Pattern. The four repositiory interfaces are defined in the application layer (Figure 29) and then implemented to concrete classes in the Presistance layer (Figure 29). After that they are registered in Service Configuration and injected via constructure in the Controller layer (Figure 28). This means that the Application layer has no dependency on the Controller and Percistance layers. The benefits of that are scalability and reusability. There is no dependency on external technology implementation or changes in the technoolgy used. For example if a change in technology is necessary for example from MongoDB to SQL Database, only the concrete class will change but the logic, encapsulated in the Application layer remains the same.

Therefore, the Dependency Inversion Principle (SOLI**D**) is followed as *"Entities must* depend on abstractions, not on concretions*"* (Oloruntoba 2021). This principle is also conducted by creating the BaseRepository abstract class that is the blueprint of how to connect to Mongo Collections and perform actions that was used to inherit from and send requestes to MongoDB. Moreover, Dependency injection is also used to inject the MongoDB database as a singleton to each of the repositories. This allows for easy implementation of other technologies and was used to simplify unit testing.

DTO pattern is used from incoming data from the frontend such as data from login and registration forms (Figure 30,top right) and also implemented for objects saved to the database. In this way there is no dependency between the domain layer and the presentation or persistence layers (Figure 30,left, right bottom). This helps abstract implementation details of domain objects.

```
namespace InterviewMaster.Controllers
{
    [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
    [Route("api/[controller]")]
    [ApiController]
    3 references
    public class UsersController : ControllerBase
    {
        private readonly IUserProfileRepository userProfileRepository;
        private readonly IQuestionsRepository questionsRepository;
        private readonly IdentityService identityService;
        private readonly IIdentityRepository identityRepository;

        1 reference
        public UsersController(IUserProfileRepository userProfileRepository,
            IQuestionsRepository questionsRepository,
            IdentityService identityService,
            IIdentityRepository identityRepository)
        {
            this.userProfileRepository = userProfileRepository;
            this.questionsRepository = questionsRepository;
            this.identityService = identityService;
            this.identityRepository = identityRepository;

        }
```

Figure 28 - Code snippet from UsersController, Injection of Identity Service via constructor

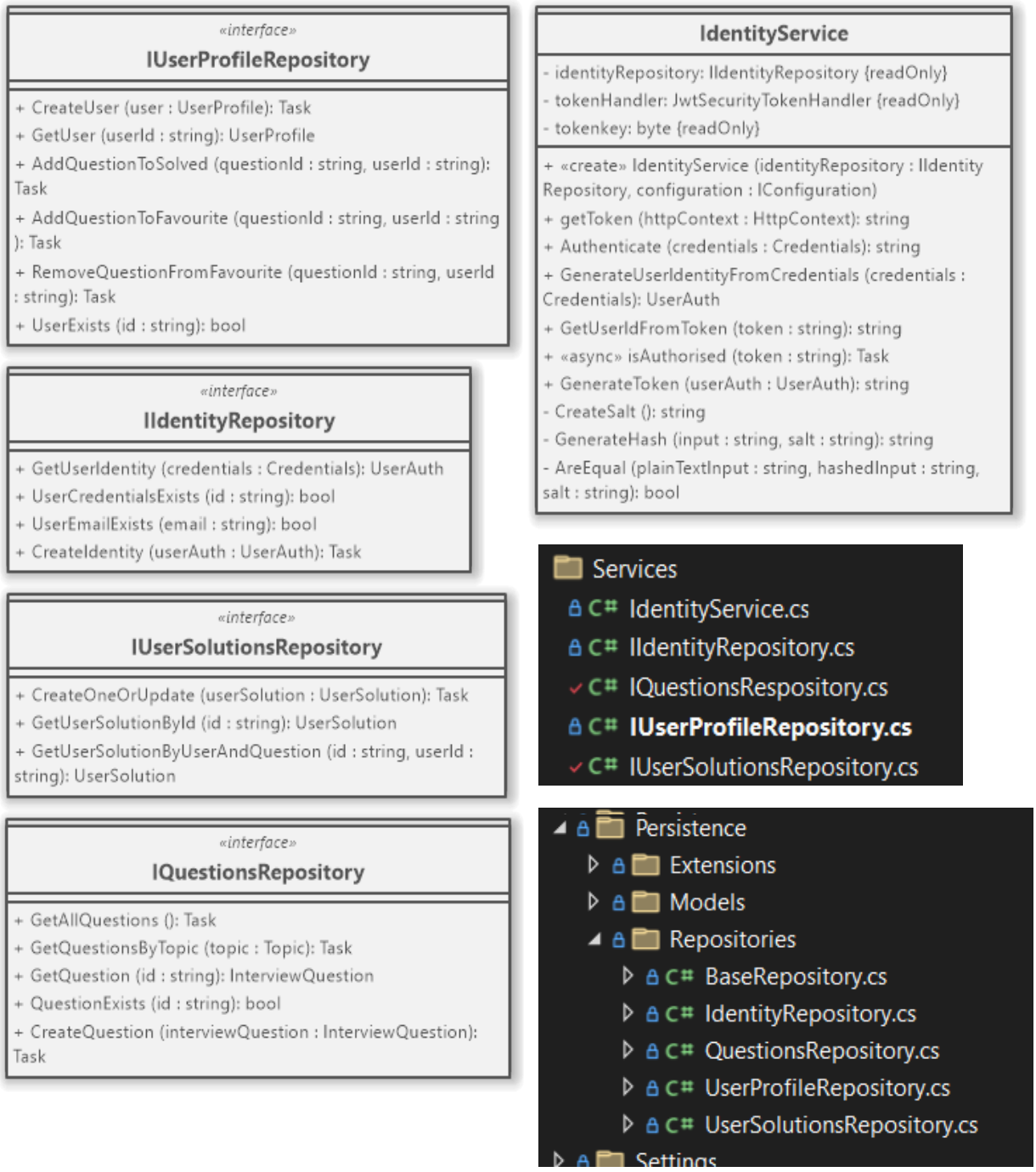
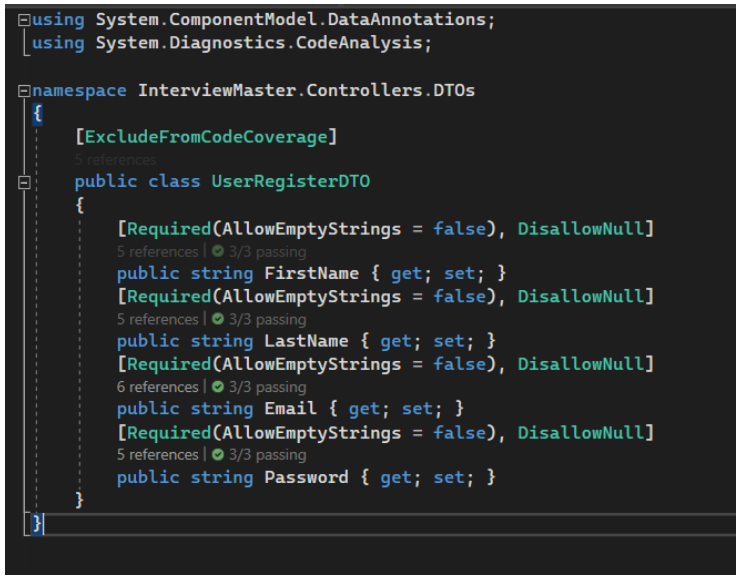


Figure 29 - Repository pattern implementation, Identity Service implementation(top right), Services and Persistance

```
using InterviewMaster.Persistence.Extensions;
using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;
using System.Collections.Generic;
using System.Diagnostics.CodeAnalysis;

namespace InterviewMaster.Persistence.Models
{
    [ExcludeFromCodeCoverage]
    [BsonCollection("Questions")]
    11 references
    public class InterviewQuestionDTO
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        20 references | 7/7 passing
        public string Id { get; set; }

        [BsonRequired]
        [BsonElement("question")]
        8 references | 1/1 passing
        public string Question { get; set; }

        [BsonElement("topic")]
        12 references | 2/2 passing
        public string Topic { get; set; }

        [BsonElement("prompts")]
        8 references | 1/1 passing
        public IEnumerable<string> Prompts { get; set; }

        [BsonElement("exampleAnswers")]
        8 references | 1/1 passing
        public IEnumerable<string> ExampleAnswers { get; set; }
    }
}
```



```
using System.ComponentModel.DataAnnotations;
using System.Diagnostics.CodeAnalysis;

namespace InterviewMaster.Controllers.DTOs
{
    [ExcludeFromCodeCoverage]
    5 references
    public class UserRegisterDTO
    {
        [Required(AllowEmptyStrings = false), DisallowNull]
        5 references | 3/3 passing
        public string FirstName { get; set; }
        [Required(AllowEmptyStrings = false), DisallowNull]
        5 references | 3/3 passing
        public string LastName { get; set; }
        [Required(AllowEmptyStrings = false), DisallowNull]
        6 references | 3/3 passing
        public string Email { get; set; }
        [Required(AllowEmptyStrings = false), DisallowNull]
        5 references | 3/3 passing
        public string Password { get; set; }
    }
}
```

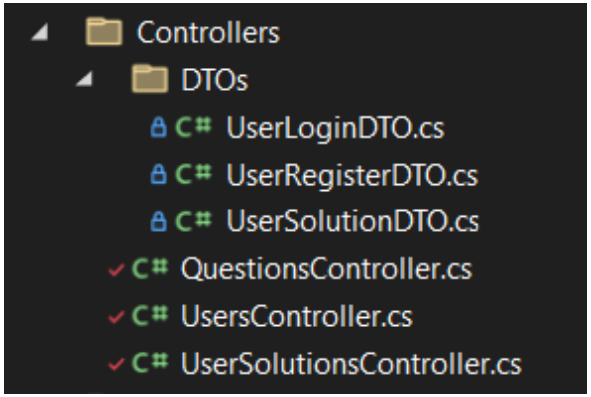


Figure 30 - DTO pattern, left MongoDB DTO, right top API DTO, right bottom DTO folder

7.2.2. Frontend

Technology Justification

The presentation tier of the application is a web application built with React, a reactive and declarative framework for JavaScript (Reactjs 2022). React was selected because of its popularity, extensive official documentation and community support (Jaiswal [no date]). Furthermore, React encourages creating reusable components and utilises a virtual DOM to track updates and only re-render the UI sections subject to updates, which results in faster development and better application performance (Jaiswal [no date]).

Alternatives to React such as Svelte were considered for the UI. Svelte is a front-end framework that has gained 150% popularity in the last two years (Twardowska 2022). Svelte has a compiler that provides better application performance and smaller file sizes, however, it is less scalable, it does not support code reusability, has less documentation and smaller community support. Thereby, React's faster development and scalability were preferred over application performance (LePage 2022).

React Project

The application follows the Single Responsibility Principle and encapsulates components, styling, pages, state management and routing separately:

- **Component**: They hold specific HTML components and functionality that can be reused multiple times. Bootstrap and Material UI were leveraged to build the web application as they provide already styled components and layouts.

- **Styling**: Each component has an associated CSS file with the styling and the application implemented a Material UI theme to encapsulate the colour palette of the design
- **Pages**: They set the layout and combine multiple components to create a page available at a given URL. Components are available for the 6 pages described in the Design section. The pages render content conditionally based on the state of the application.
- **State Management**: React has one-way data binding that follows the hierarchical structure of the virtual DOM by design, i.e., application state is only passed from parent to children in the DOM tree. Redux, a state container for JavaScript, was adopted to centralise the state management and make it available to any node in the DOM. In essence, Redux is the "single source of truth" of the application state and sits in parallel to the virtual DOM to provide global state for the application, thereby avoiding having to pass it from a parent to a child in the Virtual DOM (Three Principles | Redux. 2021). This directly contributes to achieving the non-functional requirement of keeping some of the pages on the FE private.

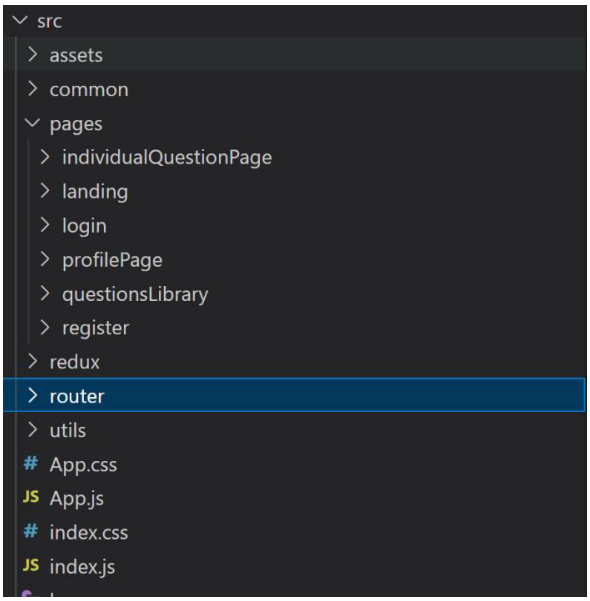


Figure 31- React project structure

- **Routing**: React applications are single page and have a single HTML file that loads the React application. React router mimics a multipage application by selecting the page component that should be rendered for each URL

Figure 32 shows the entry point of the application in index.js. Figure 33 show the React application that is loaded into the single HTML and how the main services are added:

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
);
```

Figure 32. React Application loaded in HTML, index.js

```
function App() {
  return (
    <Provider store={store}>
      <CacheProvider value={createEmotionCache()}>
        <ThemeProvider theme={theme}>
          <div className="App">
            <BrowserRouter>
              <Routes></Routes>
              <FixedTopContainer></FixedTopContainer>
            </BrowserRouter>
          </div>
        </ThemeProvider>
      </CacheProvider>
    </Provider>
  );
}

export default App;
```

Figure 33. React application from the web application source code, App.js

- Provider: Redux container to hold the global state
- CacheProvider: Maintains the Material UI theme consistently in the DOM
- ThemeProvider: Customised Material UI global theme
- BrowserRouter and Routes: Provide URL navigation and page rendering
- FixedTopContainer: This is the application navigation bar

This way of annotating allows to use these services at any point of the application as the highest in hierarchy component is App.js.

Finally, Axios library, an HTTP client for node.js was used to make asynchronous API calls to the backend and retrieve the data, thus minimising response times (Getting Started | Axios Docs. [no date]).

Visual Implementation

The final pages correspond to the UX design and are a good example of consistency in the colours, shapes and typography. Furthermore, they implement visual hierarchy,

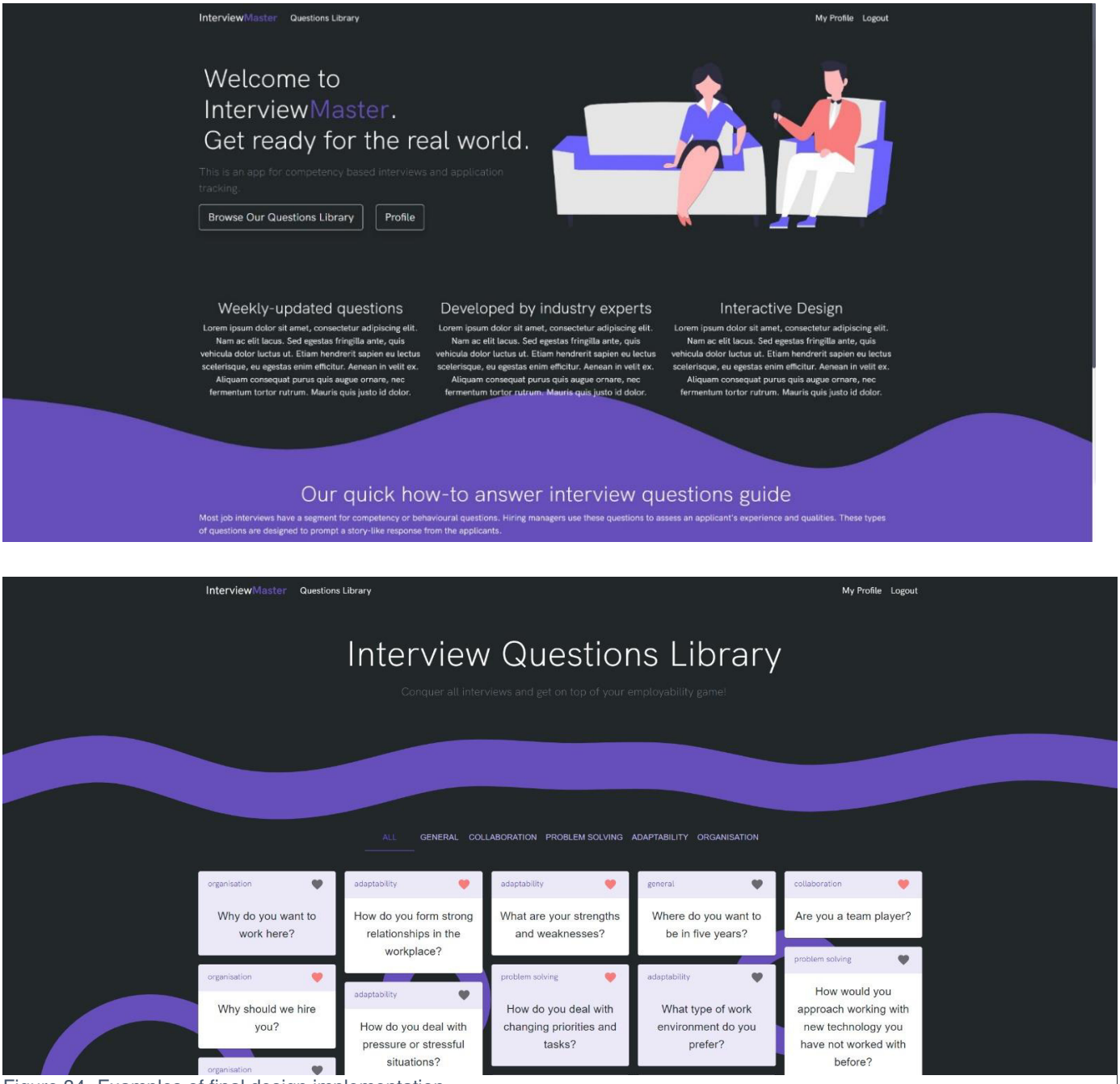


Figure 34- Examples of final design implementation

for example the main titles in Figure 34 are significantly more noticeable than the small paragraphs' titles. All of the pages are available in the demo video.

7.2.3. Testing

The MVP project implements unit testing and end to end manual testing.

Unit testing

XUnit, the most common open-source testing framework for .NET projects, was utilised for the Unit Tests. This framework was chosen as it provides a variety of learning materials online and because it is very lightweight, making it ideal for performing unit testing during development, which needs to be fast by definition (Khorikov 2020).

Unit testing was integrated in the Jira workflow and were developed with each ticket, so that all features were tested extensively at unit level.

Unit of testing in the context of the application is defined as a unit of business logic and focuses on behaviours. The name standard follows the structure of {Domain Object/Process}Should, for example InterviewQuestionShould and is follow by the name of an individual test, which is an action e.g. "Interview Question Should Fetch All Questions If Topic Is All". This directly contributes to the readability of the code as it brings clarity to the implementation of features and what the test scenarios are. There are a total of 18 test case scenarios that are executed in less than 30 seconds and available to view on Figure 35.

Figure 35 - Unit test run, all test passing, executed in 26.3 sec

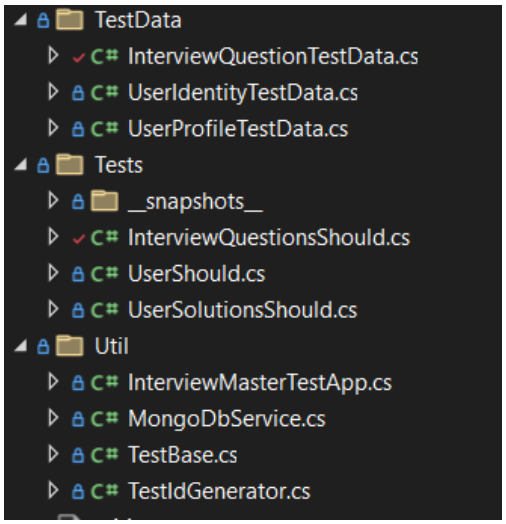


Figure 36 - Unit test structure

The testing project was structured in 3 main sections: Tests, Utilities and TestData (Figure 36). The InterviewMasterTestApp is created as a substitute that contains a dependency injection container. It allows to register replacements for the injected services similar to the Startup.cs in the main app project. This enhances testability, as discussed in the “Application, Controller and

Persistence layers” section and allows to register a MongoDB service that initialises a local Mongo Database. Therefore, the test cover whole features not isolated methods and classes. Furthermore, this eases development as it allows the developer to use the ready DTO models for MongoDB to initialise test data (Figure 37).

Finally, Snapshooter NuGet package is used to create snapshots of data that was received and simplify the result validation of the tests. Using a snapshooting tool reduces development time and allows for testing longer payloads and validate the test results in MongoDB. (Figure 38)

```
namespace InterviewMaster.Test.TestData
{
    2 references
    public class UserProfileTestData
    {
        2 references
        public static UserProfileDTO GenerateValidTestUserProfileOne()
        {
            return new UserProfileDTO()
            {
                Id = "6291dea19b4ee53850386abc",
                FirstName = "John",
                LastName = "Doe",
                FavouriteQuestions = new List<string>(),
                UserSolutions = new List<string>(),
            };
        }
    }
}
```

Figure 37 - Generating fake test data for unit tests by using the MongoDB DTOs

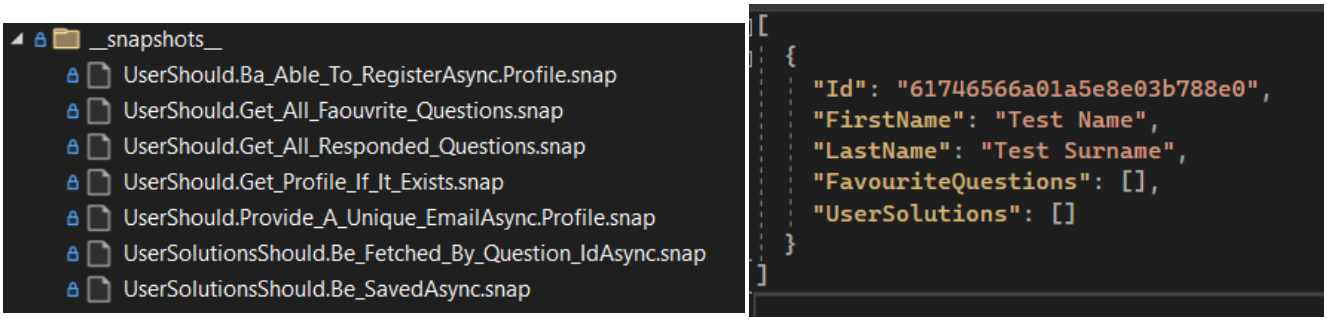
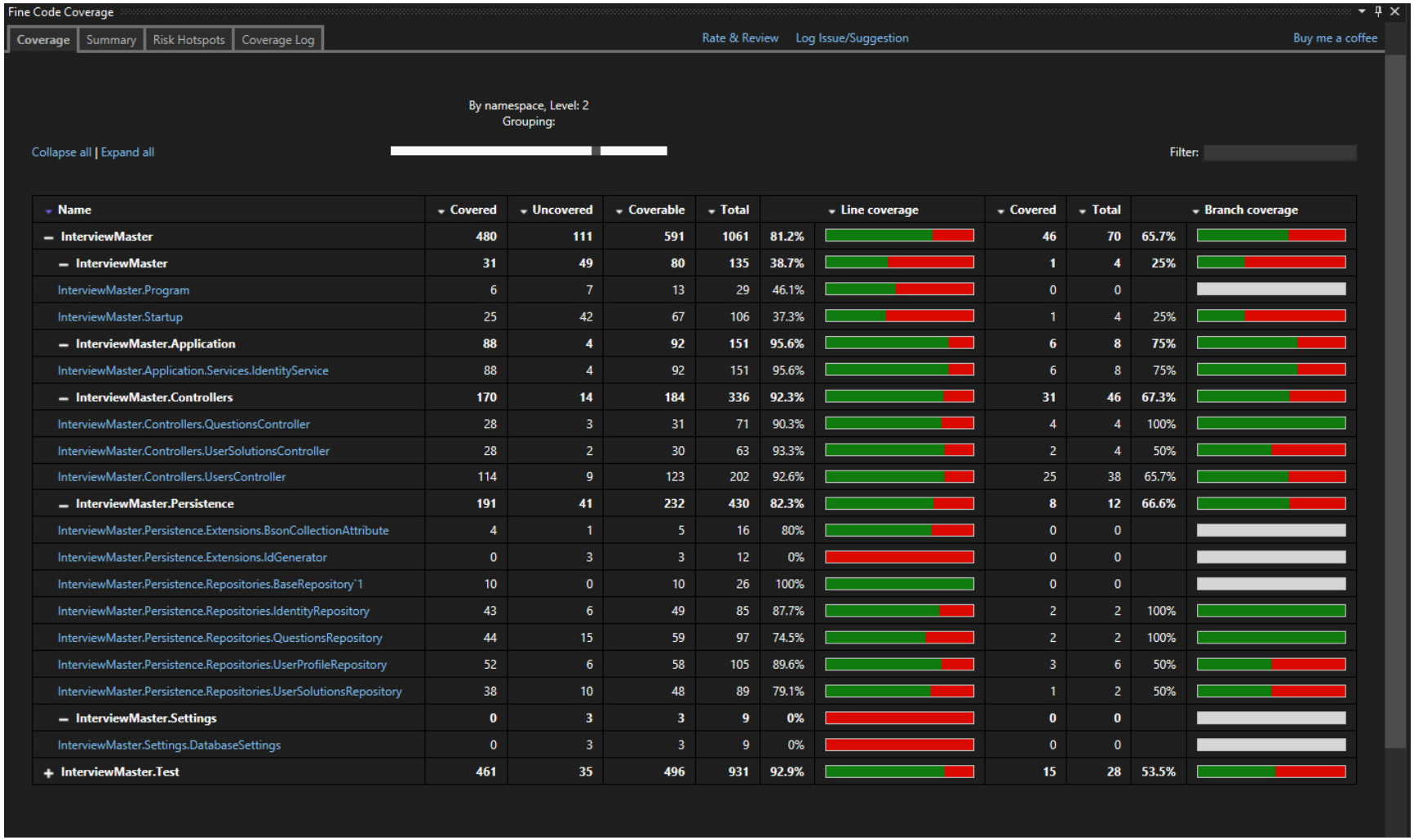


Figure 38 - Test snapshots (left) and example of a snapshot (right)

The unit test coverage was testes with “Fine Code Coverage” tool. This tool allows to see line coverage and branch coverage. This is useful when estimating what else should be tested , however, according to Khorikov (2020) only aiming to achieve 100% coverage does not guarantee that the system is resilient . This is another reason why the Unit Tests are focused on behaviours as they would only change should the business logic changes, not when implementation details change. Having this in mind, the application achieved 81% line coverage and 70% branch coverage. It is advisable to increase the branch coverage as it allows for multiple flows to be tested, nevertheless the line coverage is considered sufficient for MVP. The domain model is excluded from test coverage as any errors within it will cause an error with the outer layers of the application, thus this is an example where it is not necessary to add more tests to only increase the line/branch coverage. Furthermore, the Test Coverage is increased by the implementation of end to end manual testing.



Name	Covered	Uncovered	Coverable	Total	Line coverage	Covered	Total	Branch coverage
InterviewMaster	**480**	**111**	**591**	**1061**	**81.2%**	**46**	**70**	**65.7%**
InterviewMaster	**31**	**49**	**80**	**135**	**38.7%**	**1**	**4**	**25%**
InterviewMaster.Program	6	7	13	29	46.1%	0	0	
InterviewMaster.Startup	25	42	67	106	37.3%	1	4	25%
InterviewMaster.Application	**88**	**4**	**92**	**151**	**95.6%**	**6**	**8**	**75%**
InterviewMaster.Application.Services.IdentityService	88	4	92	151	95.6%	6	8	75%
InterviewMaster.Controllers	**170**	**14**	**184**	**336**	**92.3%**	**31**	**46**	**67.3%**
InterviewMaster.Controllers.QuestionsController	28	3	31	71	90.3%	4	4	100%
InterviewMaster.Controllers.UserSolutionsController	28	2	30	63	93.3%	2	4	50%
InterviewMaster.Controllers.UsersController	114	9	123	202	92.6%	25	38	65.7%
InterviewMaster.Persistence	**191**	**41**	**232**	**430**	**82.3%**	**8**	**12**	**66.6%**
InterviewMaster.Persistence.Extensions.BsonCollectionAttribute	4	1	5	16	80%	0	0	
InterviewMaster.Persistence.Extensions.IdGenerator	0	3	3	12	0%	0	0	
InterviewMaster.Persistence.Repositories.BaseRepository`1	10	0	10	26	100%	0	0	
InterviewMaster.Persistence.Repositories.IdentityRepository	43	6	49	85	87.7%	2	2	100%
InterviewMaster.Persistence.Repositories.QuestionsRepository	44	15	59	97	74.5%	2	2	100%
InterviewMaster.Persistence.Repositories.UserProfileRepository	52	6	58	105	89.6%	3	6	50%
InterviewMaster.Persistence.Repositories.UserSolutionsRepository	38	10	48	89	79.1%	1	2	50%
InterviewMaster.Settings	**0**	**3**	**3**	**9**	**0%**	**0**	**0**	
InterviewMaster.Settings.DatabaseSettings	0	3	3	9	0%	0	0	
InterviewMaster.Test	**461**	**35**	**496**	**931**	**92.9%**	**15**	**28**	**53.5%**

End to End Testing

End to end manual testing is implemented, so that the final application is tested as a black box from the perspective of the final user. This ensures that the frontend of the application is also tested and the integration between the three tiers is working effectively.

By definition end to end testing is the last level of the testing pyramid and should be minimal, thus 5 flows were identified and tested (Appendix 4). Those flows directly relate to the functional requirements as defined in section 5.1Functional Requirements :

- Main Flow / Response Flow

  This flow verifies that the user is able to access, save and answer interview questions.

- Favourite Flow

  Verifies that the user can favourite questions and that this state is persisted in both the “Questions library” and “My profile” pages. A bug was uncovered and raised as a result of this test, which proves its effectiveness in uncovering different scenarios.

- Topic Flow

  Verifies all the functionality related to topics.

- Login Flow and Register Flow

  Those two flows relate to the requirement that a user should be able to create and have access to their own profile and that the state of that profile is persisted.

The final results of the tests can be found in Appendix 4.

7.2.4. Version Control

“Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later” (Atlassian [no date]). Git is the most popular version control for software projects making arguably the most reliable as well (Atlassian [no date]). For the project a trunk-based development workflow was leveraged. This includes a remote and local version of the code hosted on GitHub and the researcher’s personal machine which ensures that there is sufficient back-up to complete the project and for future work. There is only one branch that is frequently committed to the remote repository, using meaningful names and the Jira ticket numbers. This helps to track which features are delivered in each commit and supports rolling back in case of regression bugs, see Figure 39 .

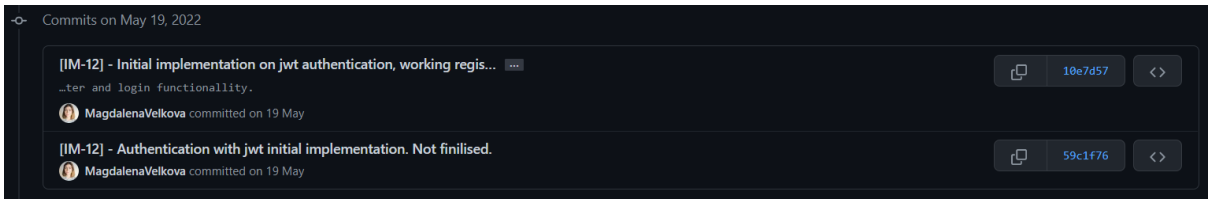


Figure 39 - Example of commit naming

7.2.5. Data Seeding

Meaningful data is identified by the “Best approaches for learning applications” as one of the most important aspects of creating successful learning applications. Therefore, a data seeding tool was developed to insert meaningful, curated interview questions and responses in the database. The tool is developed in JavaScript using Mongoose language to connect to MongoDB and seed the data.

8. Analysis

The study provides comprehensive research exploring the unemployment gap of Computer Science students. A variety of sources are used to build a discussion, identify the main reasons and explore them, making the study robust and reliable. The reason for unemployment in graduates that this study tackles is the “Lack of soft skills in graduates”. Therefore, that is explored in detail in conjunction with all the topics that relate to it, for example how are soft skills defined and learnt. Exploring a variety of topics and angles contributes to seeing the “Lack of soft skills in graduates” in a multi-dimensional way, thus enhancing the correctness of the study. In sections 2.5. – 2.7. the successful approaches for creating e-learning applications are explored and later applied in the design discussions, as per the objectives. Furthermore, the primary research generates data that shows students self-reported needs for the application and the desired functionality. Primary and secondary research findings are later applied in devising the functional requirements of the application, thus bringing the project as close as possible to achieving the project aim.

Moreover, the final MVP implementation is closely following the requirements and design, that are an outcome of the successful research. It satisfies the objectives of the project:

- As demonstrated in the Testing section all functional requirements of the project have been achieved and tested, at least through manual tests and for the Backend on Unit level as well.
- Software engineering best practices were followed and critically analysed before being implemented. This means that the complexity of the software corresponds to its maturity, but also it allows for capacity and scalability,

maintainability and testability. The latter is achieved also by implementing the suggestions in the non-functional requirements section.

- The Authentication section discusses in depth how personal information is protected through the hashing and salting of passwords as well as creating separate collections to maintain the information. This is also achieved by implementing the non-functional security requirements of keeping pages and API endpoints private. This is considered sufficient for the completion of this project and MVP, however in the next section it is discussed how this could be improved.
- The design is repetitive and consistent, and the users have multiple ways of reaching different points in the application. The psychological impact of shapes and colours is considered as well when creating the final theme.

8.1. Suggested improvements

Naturally, the research has its shortcomings. This section highlights them and gives improvement suggestions.

Research

The primary and secondary research provides extensive information for creating the MVP and for mapping the future of the application. Nevertheless, the success of the application is currently only assessed based on how it satisfies the requirements. It would be beneficial to extend the Primary research by testing the application with real users and revising the project objectives and product requirements.

In addition to this, the initial survey could have been revised to provide less more concisely defined options, so that it reduces the students' cognitive load and creates more accurate results. Furthermore, having a bigger sample group and extending the

survey to different universities would have created a more rounded opinion about the Computer Science graduates in the UK.

Implementation

While the project implementation satisfies the MVP, it mostly only considered the “happy path”, when using the application and therefore is prone to errors. This could be improved in a series of steps:

- **Exceptions:** Meaningful messages on exceptions and exception handling are important steps of application development as they exist to give information about bugs and crashes. Currently the application has no exception handling, thus it is harder to troubleshoot and less reliable as it could crash should the use encounter an unhandled exception.
- **Logging:** The current implementation of the application does not provide any logging. This creates a “black box” application after deployment and is very hard to monitor for potential issues. Logging should be implemented for important events in the application, and it should go beyond Error handling only. As a standard the log levels of an application are “Information”, “Warning” and “Error”. These are defined as following:
    - Information: A typical behaviour in the application that is beneficial when investigating a bug but should not cause concerns
    - Warning: A non-typical behaviour that is a reason for investigation but not serious concerns
    - Error: Atypical behaviour that should raise concerns and should be fixed immediately

In this way event-driven monitoring of the logs can be implemented and alerts sent if any unusual behaviour appears. For example, an information log of connecting to the database performed many times can alert of memory leaks.

- **Validation:** Input validation is important for the correct functioning of the application, preventing user error and for increasing security by preventing XSS attacks. The Entities and ValueObjects in the backend should be updated to reflect proper validation and simple validation rules such as verifying the validity of an email field or ensuring the password is at least 8 symbols should be implemented in the React application.
- **Additional testing:** More automated tests should be added to both the backend and the frontend. It is recommended to do performance and load testing as this will explore the limits of the application and will diagnose implementation weaknesses and optimise them. One such weakness has already been identified, when a user wants to change from one individual question to another by pressing the arrow buttons it takes the application 3+ seconds to identify the next question and fetch it from the database. Load test on the other hand make the system more reliable and can help in estimating costs.
- **Implementing transaction in Backend:** When saving the User aggregate (combination of the UserAuth and UserProfile classses) to the database it was decided that the personal data is going to be saved in a separate collection and later on Database to improve security. However, this creates a scenario where a UserAuth class could be created or changed and due to a network error, for example, a UserProfile might never be initiated. To rectify this the Unit of Work pattern should be implemented, meaning that a set of operation, in this case

the creation of the two domain objects and saving them into MongoDB, should be grouped by using a transaction. This ensures that if there is an error the transaction will roll back and there will not be any inconsistencies in the data that might cause errors at a later stage. There is an argument against this solution as Vernon (2013) recommends to use transactions and UoW pattern minimally when implementing DDD as they increase the complexity and readability of code, nevertheless, in this case protecting the user's personal information comes as a priority.

Another recommendation for the project is improving the accessibility of the application by following the recommendations of the Web Accessibility Guidelines (Initiative (WAI) 2020). In addition to that, loading states could be added to make the application more user-friendly and interactive. This could be achieved by creating a loading skeletons for each page that are displayed, whilst the data is fetched from the backend.

Finally, enhancing security by creating a separate password-protected Vault is a must for the future stages of the project, especially before deployment. This vault should hold MondoDB secrets and keys for hashing the passwords as they are currently exposed and create a security vulnerability.

9. Conclusions

9.1. Future Work

Based on the Literature Review and Primary research, to be able to achieve the full benefits of this solution and to attract students, the application should implement the additional functionality, requested in the initial survey (Primary research) and described in the optional requirements for MVP:

6. As a **user** I want to receive trophies and prises for completing questions **so that** I am motivated to continue my learning.
7. As a **user** I want to be able to relate interview questions to companies and filter by company **so that** my learning is targeted to the specific interview I am preparing for.
8. As a **user** I want to track my job applications **so that** I can maintain good organisation while applying for placement/graduate opportunities.

This could be achieved by continuing to implement DDD and creating new microservices for Application Tracking and Company Intelligence, similarly to the Interview Preparation application, created for MVP and adding the trophy and prises to the Interview Preparation Domain. The Application tracking and Company Intelligence give the users to free access to more functionality, and brings them onto the website, whereas the instant gratification motivates them to continue using it. Prises, like Duolingo's streak, for using the application in consecutive days is also recommended. In addition, implementing Merill's principles of instruction is going to increase the product's effectiveness. Furthermore, leveraging a microservice architecture contributes towards looser coupling between the services and better scalability.

Another suggestion is to expand the application on the mobile platform. This will allow the students to use their preferred medium and give the ability to send direct notifications to their phones and prompt them to continue their learning. This should create more engagement from the users onto the mobile and website platforms and will give them flexibility to use their preferred way of learning. In terms of technological solution this could be achieved by using the API gateway design pattern and again Microservice architecture.

For the future stages of this project it is suggested to extend the Authentication application. This is because storing personal data comes with additional costs and risk of security and GDPR breaches. A solution for this is to implement Google authentication. This will guarantee a smooth experience when logging in and will relieve the project of having to maintain any personal data, such as names, emails and passwords, thus transferring the responsibility for the security of those data to Google.

10. Reflection on learning

This section aims to reflect on the project completion process and on the learning achievements. Bain reflective model is considered most appropriate for completing this section because it provides a systematic approach to reflective learning (Bain 2002; The University of Edinburgh 2018). Furthermore, it is more recent opposing to other approaches such as Gibbs' reflective cycle and Kolb's Learing cycle (Kolb 1984; Gibbs et al. 1988). In summary, Bain's model contains five steps:

- **Reporting** of the context of the experience
- **Responding** to the experience (observations, feelings, thoughts, etc.)
- **Relating** the experience to knowledge and skills you already have
- **Reasoning** about the significant factors/theory to explain the experience
- **Reconstructing** your practice by planning future actions for a similar experience

The full work is available in the Reference section (Bain 2002).

To complete the project, it was required to research non-technical topics such as employability and softs skills. Researching "the gap in employability" and completing the Literature Review has given me knowledge that I can apply in my own career journey and development. I was able to reflect on my personal soft skills and educate myself beyond the scope of the project on how to articulate them better. This has helped me in my day-to-day job and I strongly believe it will continue to in the future.

The application and report development process could be generally split into two phases: before September 2021 and after January 2022. The idea generation, choosing technologies and developing the project with FastAPI, React and MongoDB

took place in the summer of 2021 with the completion of the taught modules at university. I was aiming to use the project as an opportunity to learn new technologies and strengthen my knowledge from the MSc Computing taught modules. Furthermore, to manage the workload for the original deadline in September 2021 I had to base the project on technologies I know as well as new ones. Therefore, I have selected React for the Frontend as one of the most popular JavaScript frameworks. Having learnt JavaScript at university helped me speed up the learning process. Python and FastAPI were chosen for similar reasons as Python were the language that I felt most knowledgeable during the degree but I wanted a framework that exposes more implementation details and is more lightweight and flexible in comparison to the one taught during the degree (Flask). In addition to the degree, I have started a new position in September, thus my workload became hard to manage and I had to move to a part-time education mode. Nevertheless, this experience taught me how to effectively manage my time and allow for more time to absorb fresh knowledge and learn new technologies. Upon reflection, I believe moving to part-time education has also allowed me to benefit more from the dissertation project, explore and learn new technologies and approaches. Alternatively, I would have allowed more time to learn and reduce the scope of the MVP to match my abilities or I would choose technologies I am more familiar with to complete the project on time. This, however, would not have achieved my goal of learning during the course of the project and gaining additional time management skills.

Between September 2021 and January 2022, I have been exploring other languages and frameworks and have gained an interest in C# and .NET Core framework. Furthermore, I was working as a software engineer, which helped me see how projects are handled in practice, in a professional environment. I learned new concepts from

my job and used my project to practice them. For example, I was not aware of Domain Driven Design and architecture processes beforehand so researching those has enhanced my abilities to make decisions and create more maintainable, readable and scalable implementations. This has aided me to redesign my application, using those new concepts and create a better solution.

Furthermore, in this timeframe I had sufficient time to practice my React knowledge and do additional courses.

In January 2022, as I felt more confident with React, I refactored my web client application and to explored in more depth the libraries I am using such as React Redux. I started doing C# and .NET Core courses for Web development and was learning different design patterns and strategies for implementation and re-made the whole Backend. Using MVC and two different applications for the API and the Frontend, opposing to a monolith made it easier to do this and I saw the benefits of loose coupling dependencies first-hand. Having this experience, in conjunction with the architectural knowledge from my job will benefit me in future projects as I now consider what effect such changes have and ways to avoid them.

Moreover, I have put higher priority onto testing and read a book about it that helped me understand it better (in particular unit testing). The project gave me the opportunity to practice my learning and to set up my own Testing project. During the process I experimented with an In memory data storge that slowed down the implementation sufficiently. This lead me to explore a solution, where a MongoDB is available in memory for the Unit tests, so that the CRUD operations do not need to be mocked for a different implementation. In retrospect, this has taught me the time cost of overcomplicating Unit Tests and the importance of good architectural decisions even

when setting a testing project up. If I am to complete this project again I will use Test Driven Development as now I can clearly see how it reduces the timeframes and increases the system's reliability.

I have updated my Jira board with my tasks and estimated my tasks. This was a direct result of my previous project management experience and the knowledge I gained for those three months. In retrospective, there was a big improvement in this area, however for future projects I will also implement Sprints for more structured and definitive completion timeframes. This is because I often found myself adding more stories and the work was not sufficiently prioritised until the later stages of the SDLC. Furthermore, I believe I achieved the purpose of this project by learning how to use React and C# .NET Core from the very basics.

11. References

9 Rules For UX Design. 2018. Available at: https://usabilitygeek.com/rules-for-ux-principles/ [Accessed: 31 July 2022].

Accessibility for Teams. 2022. Available at: https://accessibility.digital.gov/ux/inclusive-design/ [Accessed: 31 July 2022].

Alexandra, J. 2022. Programmers & software developers in the UK 2020. Available at: https://www.statista.com/statistics/318818/numbers-of-programmers-and-software-development-professionals-in-the-uk/ [Accessed: 3 September 2021].

Almonte, R. 2022. A practical guide to soft skills: communication, psychology, and ethics for your professional life. New York: Routledge, Taylor & Francis Group.

Al-Saqqa, S. et al. 2020. Agile Software Development Approaches And Trends.

Atlassian [no date][a]. What are story points and how do you estimate them? Available at: https://www.atlassian.com/agile/project-management/estimation [Accessed: 25 July 2022].

Atlassian [no date][b]. What is Git: become a pro at Git with this guide | Atlassian Git Tutorial. Available at: https://www.atlassian.com/git/tutorials/what-is-git [Accessed: 1 September 2022].

Auth0 2022. JSON Web Tokens. Available at: https://auth0.com/docs/ [Accessed: 1 September 2022].

Author, H. 2021. Understanding the Phases of the Software Development Life Cycle (SDLC). Available at: https://harness.io/blog/devops/software-development-life-cycle/ [Accessed: 21 July 2022].

Avram, A. and Marinescu, F. 2006. Domain-Driven Design Quickly: A Summary of Eric *Evans' Domain*-Driven Design. C4Media.

Babich, N. 2022. Information Architecture Guide for UX Architects & Designers | Adobe XD Ideas. Ideas . Available at: https://xd.adobe.com/ideas/process/information-architecture/information-ux-architect/ [Accessed: 31 July 2022].

baeldung 2021. The DTO Pattern (Data Transfer Object) | Baeldung. Available at: https://www.baeldung.com/java-dto-pattern [Accessed: 2 September 2022].

Bain, J.D. 2002. *Reflecting on practice: student teachers' perspectives*. Flaxton, Qld.: Post Pressed.

BCS, T.C.I. for I. 2022. Record numbers have applied for UK computer science degrees this year. Available at: https://www.bcs.org/articles-opinion-and-research/record-numbers-have-applied-for-uk-computer-science-degrees-this-year/ [Accessed: 4 July 2022].

Boyde, J. 2014. A Down-To-Earth Guide To SDLC Project Management (2nd Edition): Getting your system / software development life cycle project successfully across the line using PMBOK adaptively. Joshua Boyde.

Bridgstock, R. 2009. The graduate attributes we've overlooked: Enhancing graduate employability through career management skills. Higher Education Research and Development 28. doi: 10.1080/07294360802444347.

C# vs Python: Head to Head Comparison [Updated]. 2022. Available at: https://hackr.io/blog/c-sharp-vs-python [Accessed: 30 August 2022].

Careers and Employability. [no date]. Available at: https://www.cardiff.ac.uk/study/student-life/student-support/careers-and-employability [Accessed: 17 May 2021].

CBI 2019. *Getting young people 'work ready'*. Available at: https://www.cbi.org.uk/media/2960/cbi_work-readiness.pdf.

CHERI 2002. UK graduates and the impact of work experience.

Chou, W. 2013. Fast-tracking your career: soft skills for engineering & IT professionals. Hoboken, New Jersey: IEEE Press. doi: 10.1002/9781118662144.

Clarke, M. 2009. Plodders, Pragmatists, visionaries and opportunists: Career patterns and employability. The Career Development International 14(1), pp. 8–28. doi: 10.1108/13620430910933556.

Coach, G. 2020. Why is it so hard to get a job after university? Available at: https://graduatecoach.co.uk/so-hard-to-get-a-job/ [Accessed: 2 May 2021].

Cubet 2015. Introduction to Repository Design Pattern. Cubettech 24 July. Available at: https://cubettech.com/resources/blog/introduction-to-repository-design-pattern/ [Accessed: 15 May 2021].

DDD Part 2: Tactical Domain-Driven Design | Vaadin. [no date]. Available at: https://vaadin.com/blog/ddd-part-2-tactical-domain-driven-design [Accessed: 22 August 2022].

Department for Business, Innovation and Skills 2016. Computer science graduate employability: qualitative interviews with graduates., p. 64.

Devins, D. and Hogarth, T. 2005. Employing the Unemployed: Some Case Study Evidence on the Role and Practice of Employers. Urban Studies 42(2), pp. 245–256. doi: 10.1080/0042098042000316128.

Disadvantages of Agile. [no date]. Available at: https://www.planview.com/resources/articles/disadvantages-agile/ [Accessed: 24 July 2022].

Drunk Tank Pink. [no date]. Available at: https://www.colormatters.com/color-and-the-body/drunk-tank-pink [Accessed: 31 July 2022].

Duhigg, C. 2013. The power of habit: why we do what we do and how to change. New York: Random House Books.

Dwyer, G. 2021. Stateful vs Stateless Architecture: Why Stateless Won | Virtasant. Available at: https://www.virtasant.com/blog/stateful-vs-stateless-architecture-why-stateless-won [Accessed: 2 September 2022].

Elliott, G. 2004. Global Business Information Technology: An Integrated Systems Approach. Pearson Education.

Evans, E. 2003. Domain-Driven Design Tackling Complexity in the Heart of Software. Sydney: Pearson Education, Limited. Available at: https://public.ebookcentral.proquest.com/choice/PublicFullRecord.aspx?p=7054140 [Accessed: 22 August 2022].

Falls, I. et al. 2014. Factors Influencing Students' Perceptions of Online Teamwork. SAGE Open 4(1), p. 2158244014525415. doi: 10.1177/2158244014525415.

FARM Stack Course - FastAPI, React, MongoDB [no date]. Available at: https://www.youtube.com/watch?v=OzUzrs8uJl8 [Accessed: 1 August 2021].

FastAPI. [no date]. Available at: https://fastapi.tiangolo.com/ [Accessed: 23 May 2021].

Fowler, M. 2020. bliki: DomainDrivenDesign. Available at: https://martinfowler.com/bliki/DomainDrivenDesign.html [Accessed: 29 August 2022].

Friedman, V. 2018. Responsive Web Design - What It Is And How To Use It — Smashing Magazine. Available at: https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/ [Accessed: 31 July 2022].

Getting Started | Axios Docs. [no date]. Available at: https://axios-http.com/docs/intro [Accessed: 15 September 2022].

Gibbs, G. et al. 1988. Learning by doing: a guide to teaching and learning methods. London: FEU.

Higginbotham, D. 2021. Graduate schemes 2021 | Prospects.ac.uk. Available at: https://www.prospects.ac.uk/careers-advice/getting-a-job/graduate-schemes [Accessed: 2 May 2021].

History: The Agile Manifesto. [no date]. Available at: https://agilemanifesto.org/history.html [Accessed: 23 July 2022].

IBM [no date]. What is Software Testing and How Does it Work? | IBM. Available at: https://www.ibm.com/topics/software-testing [Accessed: 15 September 2022].

Initiative (WAI), W.W.A. 2020. Web Content Accessibility Guidelines (WCAG) Overview. Available at: https://www.w3.org/WAI/standards-guidelines/wcag/ [Accessed: 14 February 2021].

Jaiswal, S. [no date]. Pros and Cons of RectJS. Available at: https://www.javatpoint.com/pros-and-cons-of-react [Accessed: 7 September 2022].

Janssen, T. 2018. Design Patterns Explained – Dependency Injection with Code Examples. Available at: https://stackify.com/dependency-injection/ [Accessed: 6 September 2022].

Joseph, D. et al. 2010. Practical intelligence in IT: assessing soft skills of IT professionals. Commun. ACM 53, pp. 149–154.

Khorikov, V. 2020. Unit testing: principles, practices, and patterns. Shelter Island, NY: Manning.

Kinsbruner, E. [no date]. The Testing Pyramid & How to Use It. Available at: https://www.perfecto.io/blog/testing-pyramid [Accessed: 15 September 2022].

Kolb 1984. Kolb's Learning Styles and Experiential Learning Cycle | Simply Psychology. Available at: https://www.simplypsychology.org/learning-kolb.html [Accessed: 4 September 2022].

Krug, S. 2006. *Don't make me think! a common sen*se approach to Web usability. 2nd ed. Berkeley, Calif: New Riders Pub.

LePage, J. 2022. Svelte vs React: Which is the best library in 2022? - Isotropic. https://isotropic.co/ . Available at: https://isotropic.co/svelte-vs-react/ [Accessed: 7 September 2022].

Mansour, B.E. and Dean, J.C. 2016. Employability Skills as Perceived by Employers and University Faculty in the Fields of Human Resource Development (HRD) for Entry Level Graduate Jobs. Journal of Human Resource and Sustainability Studies 4(1), pp. 39–49. doi: 10.4236/jhrss.2016.41005.

Martin, R. 2000. Design Principles and Patterns.

Mason, G. et al. [no date]. How Much Does Higher Education Enhance the Employability of Graduates?, p. 52.

McCall's Quality Model. 2020. GeeksforGeeks 4 July. Available at: https://www.geeksforgeeks.org/mccalls-quality-model/ [Accessed: 9 August 2022].

McQuaid, R.W. et al. 2005. Introducing Employability. Urban studies (Edinburgh, Scotland) 42(2), pp. 191–195. doi: 10.1080/0042098042000316092.

McQuaid, R.W. and Lindsay, C.D. 2005. The concept of employability. Urban Studies , pp. 197–219.

Microservices Pattern: API gateway pattern. [no date]. Available at: http://microservices.io/patterns/apigateway.html [Accessed: 5 September 2022].

Microsoft 2022. NuGet Gallery | Packages. Available at: https://www.nuget.org/packages [Accessed: 4 September 2022].

Misra, R.K. and Khurana, K. 2017. Employability Skills among Information Technology Professionals: A Literature Review. Procedia Computer Science 122, pp. 63–70. doi: 10.1016/j.procs.2017.11.342.

Mitzi, W. [no date]. Improving Student Employability.

Nare, K. 2021. Analyzing Duolingo from Product Design Perspective after 400 Days Of Non-Stop Practice | by Nare K. | UX Planet. Available at: https://uxplanet.org/analyzing-duolingo-from-product-design-perspective-after-400-days-of-non-stop-practice-c4d4809bdb37 [Accessed: 17 July 2022].

Nation, T. 2021. Tech hiring at its highest level for five years. Available at: https://technation.io/news/tech-hiring-at-its-highest-level-for-five-years/ [Accessed: 5 July 2022].

Newell, S. [no date]. What is the Software Development Lifecycle? Available at: https://www.productplan.com/learn/software-development-lifecycle/ [Accessed: 21 July 2022].

NoSQL Vs SQL Databases | MongoDB. [no date]. Available at: https://www.mongodb.com/nosql-explained/nosql-vs-sql [Accessed: 1 September 2022].

Oloruntoba, S. 2021. SOLID: The First 5 Principles of Object Oriented Design | DigitalOcean. Available at: https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design, https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design [Accessed: 2 September 2022].

Oxford Advanced Learner's Dictionary 2022. Definition of employability. Available at: https://www.oxfordlearnersdictionaries.com/definition/english/employability [Accessed: 1 September 2022].

Patel, S. 2019. How To Design And Build Successful eLearning Websites: Key Tips And Principles To Follow. Available at: https://elearningindustry.com/build-successful-elearning-websites-key-tips-principles-follow [Accessed: 19 July 2022].

Ponsukcharoen, T. 2017. Technical report - Skills analysis - April 2017. Available at: https://docs.google.com/document/d/1h8OOkz1L7AZQvOYFbY3jgzh8yZMVKlZifSzdDEY07kU/edit?usp=embed_facebook [Accessed: 8 July 2022].

Professional Web Design Process Explained in 5 minutes 2020. Available at: https://www.youtube.com/watch?v=IbOyBIS57C0 [Accessed: 30 July 2022].

Rainbow Table Attack. 2022. Available at: https://www.beyondidentity.com/glossary/rainbow-table-attack [Accessed: 15 September 2022].

Rauser, A. 2021. The Psychology of Lines and Shapes in Web Design. Available at: https://prototype.net/blog/web-design-shapes [Accessed: 23 July 2022].

Raza, M. 2020. Agile vs Waterfall SDLCs: What's The Difference? Available at: https://www.bmc.com/blogs/agile-vs-waterfall/ [Accessed: 23 July 2022].

Reactjs 2022. React. Available at: https://reactjs.org/ [Accessed: 15 September 2022].

Rosenberg, M.J. 2001. E-learning: strategies for delivering knowledge in the digital age. New York: McGraw-Hill.

Schipper, M. and van der Stappen, E. 2018. Motivation and attitude of computer engineering students toward soft skills. In: 2018 IEEE Global Engineering Education Conference (EDUCON)., pp. 217–222. doi: 10.1109/EDUCON.2018.8363231.

Shadbolt, S.N. 2016. Shadbolt review of computer sciences degree accreditation and graduate employability., p. 91.

Sharma, A. 2020. 10 Tips For Creating The Perfect eLearning Website. Available at: https://elearningindustry.com/10-tips-creating-perfect-elearning-website [Accessed: 14 July 2022].

Sharma, L. 2022. Why is Testing Necessary and Important? | ISTQB. Available at: https://www.toolsqa.com/software-testing/istqb/why-is-testing-necessary/ [Accessed: 15 September 2022].

Smith, S. et al. 2018. The impact of work placement on graduate employment in computing: Outcomes from a UK-based study. Available at: https://files.eric.ed.gov/fulltext/EJ1199461.pdf.

Sobral, S. 2019. 30 YEARS OF CS1: PROGRAMMING LANGUAGES EVOLUTION. doi: 10.21125/iceri.2019.2214.

Soft skills – how to assess them. [no date]. Available at: https://www.pagepersonnel.co.uk/advice/management-advice/attracting-candidates/soft-skills-how-assess-them [Accessed: 14 July 2022].

The 6 Key Principles of UI Design. 2021. Available at: https://maze.co/collections/ux-ui-design/ui-design-principles/ [Accessed: 17 July 2022].

The Disadvantages of Agile Methodology. 2019. Available at: https://www.lucidchart.com/blog/3-disadvantages-of-agile-methodology [Accessed: 24 July 2022].

The University of Edinburgh 2018. The 5R framework for reflection. Available at: https://www.ed.ac.uk/reflection/reflectors-toolkit/reflecting-on-experience/5r-framework [Accessed: 4 September 2022].

Three Principles | Redux. 2021. Available at: https://redux.js.org/understanding/thinking-in-redux/three-principles [Accessed: 7 September 2022].

TIOBE Index. [no date]. Available at: https://www.tiobe.com/tiobe-index/python/ [Accessed: 12 July 2022].

Todd, B. 2017. These skills make you most employable. Why isn’t coding in the top 10? Available at: https://80000hours.org/articles/skills-most-employable/ [Accessed: 8 July 2022].

Tuning 2008. Tuning Project. Available at: https://www.unideusto.org/tuningeu/tuning-methodology.html [Accessed: 14 September 2022].

Twardowska, B. 2022. Why Svelte is the Next Big Thing in JavaScript Development. Available at: https://naturaily.com/blog/why-svelte-is-next-big-thing-javascript-development [Accessed: 7 September 2022].

Vernon, V. 2013. Implementing domain-driven design. Upper Saddle River, NJ: Addision-Wesley.

Vlahovic, N. [no date]. Implications of Domain-driven Design in Complex Software Value Estimation and Maintenance using DSL Platform. Available at: https://www.semanticscholar.org/paper/Implications-of-Domain-driven-Design-in-Complex-and-Vlahovic/41b403e4d61f8de7dc7364bf6186d5fc5584efe7 [Accessed: 24 August 2022].

What is User Interface Design? [no date]. Available at: https://www.interaction-design.org/literature/topics/ui-design [Accessed: 30 July 2022].

When to Use Waterfall vs. Agile - Macadamian. [no date]. Available at: https://www.macadamian.com/learn/when-to-use-waterfall-vs-agile/ [Accessed: 23 July 2022].

11. Appendices

11.1. Appendix 1 - Ethical Review Approval

Re: Ethics Review - Magdalena Velkova - MSc Computing

COMSC Ethics <comsc-ethics@cardiff.ac.uk>
Wed 7/21/2021 3:31 PM
To: Magdalena Velkova <VelkovaMS@cardiff.ac.uk>

Dear Magdalena,

Research project title: Application to help students prepare for placement and graduate interviews
SREC reference: COMSC/Ethics/2021/087

The SCHOOL OF COMPUTER SCIENCE & INFORMATICS RESEARCH ETHICS COMMITTEE ('Committee') reviewed the above application at the meeting held electronically on 21st July 2021.

Ethical Opinion: FAVOURABLE

The Committee gave a favourable ethical opinion of the above application on the basis described in the application form, protocol and supporting documentation.

Additional approvals:

This letter provides an ethical opinion only. You must not start your research project until all appropriate approvals are in place. This is not a matter for the committee to advise on. For student projects, you should contact your supervisor. For staff projects/supervisors, contact the School Research Office comsc-research@cardiff.ac.uk.

Amendments:

Any substantial amendments to documents previously reviewed by the Committee must be submitted to the Committee for consideration and cannot be implemented until the Committee has confirmed it is satisfied with the proposed amendments.

You are permitted to implement non-substantial amendments to the documents previously reviewed by the Committee but you must provide a copy of any updated documents to the Committee via comsc-ethics@cardiff.ac.uk for its records.

Monitoring requirements:

The Committee must be informed of any unexpected ethical issues or unexpected adverse events that arise during the research project. This notification should be made via email to us. The Committee must be informed when your research project has ended. This notification should be made within three months of research project completion.

Complaints/Appeals:

If you are dissatisfied with the decision made by the Committee, please contact the school's Ethics Officer, Dr Liam Turner in the first instance to discuss your complaint. If this discussion does not resolve the issue, you are entitled to refer the matter to the Head of School for further consideration. The Head of School may refer the matter to the Open Research Integrity and Ethics Committee (ORIEC), where this is appropriate. Please be advised that ORIEC will not normally interfere with a decision of the Committee and is concerned only with the general principles of natural justice, reasonableness and fairness of the decision.

Please use the Committee reference number (SREC reference) on all future correspondence.

The Committee reminds you that it is your responsibility to conduct your research project to the highest ethical standards and to keep all ethical issues arising from your research project under regular review.

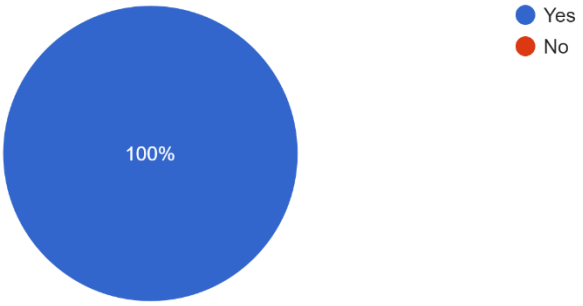
You are expected to comply with Cardiff University's policies, procedures and guidance at all times, including, but not limited to, its Policy on the Ethical Conduct of Research involving Human Participants, Human Material or Human Data and our Research Integrity and Governance Code of Practice.

Yours sincerely,
COMSC SREC

11.2. Appendix 2 – Full Survey Results

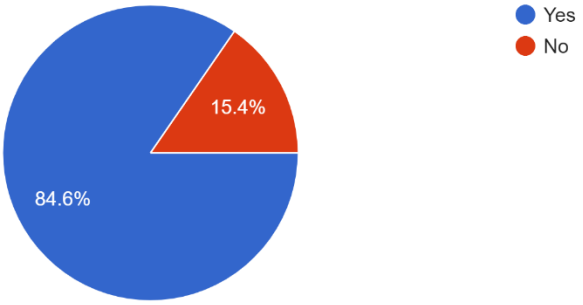
Are you currently enrolled at university or a recent graduate?

26 responses



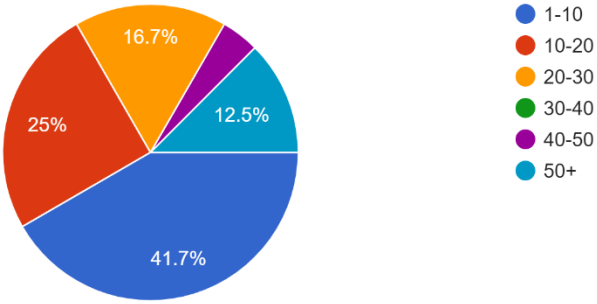
Have you started to look for job opportunities?

26 responses



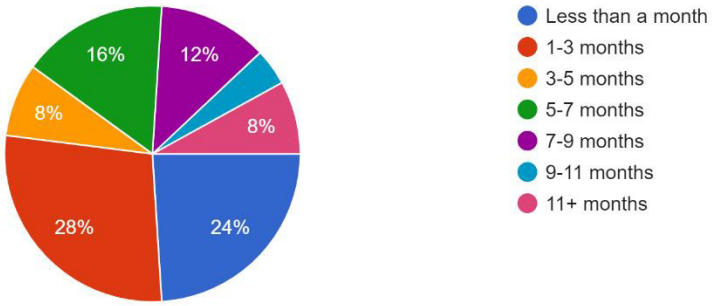
How many graduate/placement positions have you applied for in total?

24 responses



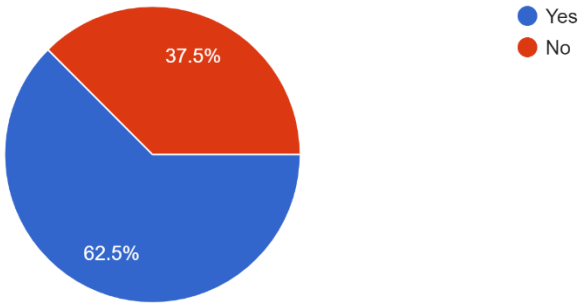
For how long have you been applying for graduate/placement positions?

25 responses



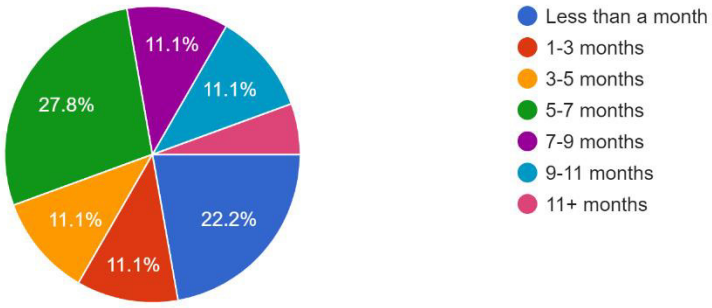
Have you received a job offer for a graduate/placement programme?

24 responses

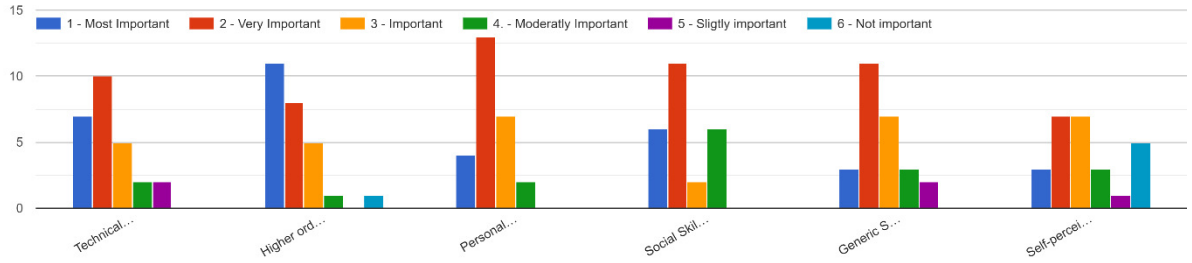


If yes, how long did it take between your first application and receiving a job offer?

18 responses

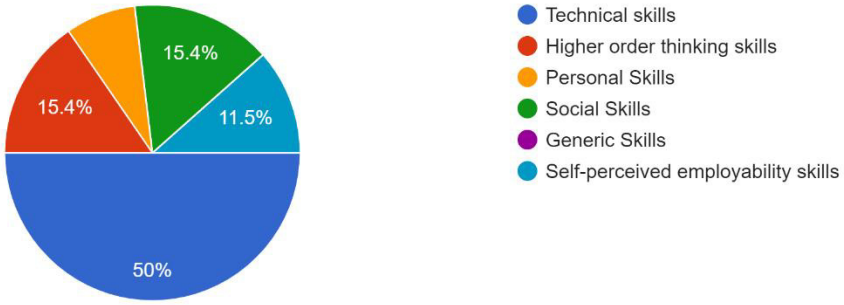


Rate the following skills in order of priority (1 being the highest) that you consider most important in finding a job:



Which of those areas do you think that you can benefit from improving?

26 responses



Please tell us why?

22 responses

I don't feel that I have learned enough coding skills from the MSc - I am still a beginner despite doing very well in assignments and passing the modules.

I am not a confident coder.

Specifically occupational skills - having solid knowledge in CS theory and practical knowledge in a programming knowledge we're coming themes in my interviews/applications. Having more knowledge/confidence in these areas would make the interview process less daunting and ultimately improve your work in related roles.

I have felt as a woman in STEM that I do not belong and need to work on my confidence

---

I am not very good with people

I feel like I've got a good grasp on everything else

I believe I need more practice in creating projects and completing them to a good standard

---

Actually it is not higher order thinking skills important, the most important thing is that you can express your thinking in free. Most people know technical skills but they do not know how to share. They just know do do do, but do not know promote their ideas. Like writing code, you should not be silence, be ready to tell a good story of what you are writing and be ready to communicate with partners telling them what they need do. Speaking out idea in a short time, at least in the interview, is very important, and it needs practice. I do not think Cardiff University does this well, especially in the pandemic.

I feel confident in the soft skills I've gained in other industries - as a conversion masters student, I think I have a diverse skillset in that regard - but I am competing with technically proficient applicants from STEM backgrounds. As such, becoming more technically skilled would make me a more well-rounded candidate.

Many jobs require particular technical skills

Having a genuine conversation with your interviewer is generally far more beneficial then behaving like somebody who is being interrogated

---

These skills can be trained even during the study of our MSc Course

---

Little experience in the technical skills aside from Master's program. Likely to develop more when in a work environment.

Criteria employers are looking for is often unclear

I have Asperger's Syndrome and find networking and small talk difficult

---

creative but rational solutions to worldly problems are always in the highest demand

---

It would help pass tests that the graduate schemes put across.

I sometimes am too rash in decisions which I need to improve on

---

Technical skills are highly prioritized by employers in China since they usually want well-equipped graduates.

Graduate jobs feel like a new experience

Technical skills in the Cyber security realm are very important

What type of preparation tools have you used through the application process?

21 responses

Linkedin, CodeCademy, Google, etc.

---

none

Leetcode, AlgoExpert

Practice interviews mainly. I have been through the process a lot by this point so am well practiced in other things like CV creation and cover letters etc

---

I have used various interview techniques practising quistions on social interactions

---

Leetcode, hackerrank, CTCI

I haven't

A blank paper, recall what I have done.

Self-study courses (eg. online free bootcamps and tutorials) and personal programming projects.

Doing extremely thorough research into the company, practicing aptitude tests, preparing answers to commonly asked questions

---

Google, Uni Career Services

Cardiff Careers centre resources

CV advice

Mock interviews

Articles about CV and Cover letter writing, LinkedIn, numeracy tests

---

Meeting with university careers staff, the CV checker

Practice behavioural tests

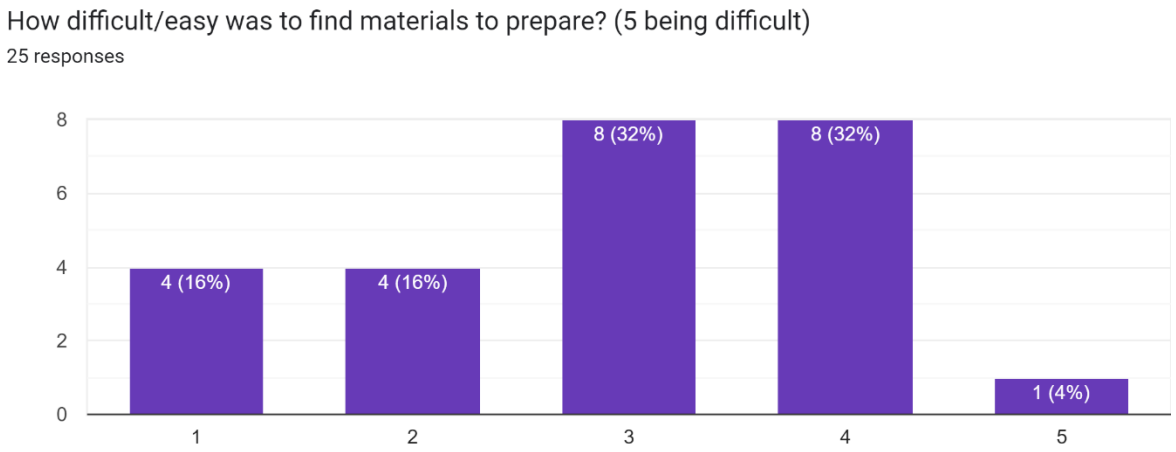
CV support

Guides for CV writing online for free, for interview skill I got better as I done real interviews with various companies.

Notes apps.

None

Mostly determination. The ability to pick yourself up after every rejection and just keep applying



11.3. Appendix 3 – Jira Issues

Note: the issues were exported as CSV and some of the fields such as description and acceptance criteria were omitted for simplicity

Summary	Issue key	Issue Type	Status	Created	Updated	Descriptio	Story point
Interview questions grouped in topics	IM-1	Task	Done	04/10/2021 21:36	10/04/2022 22:54	As a user I	8
Individual question page	IM-2	Task	Done	04/10/2021 21:38	10/04/2022 22:56	As a user I want to ac	
Saving answers	IM-3	Task	Done	04/10/2021 21:40	10/04/2022 11:10	As a user I	5
Edit answers	IM-4	Task	Done	04/10/2021 21:41	10/04/2022 11:12	As a user I	3
Status functionality of questions	IM-5	Task	Done	04/10/2021 21:42	02/06/2022 14:47	As a user	3
Job application creation and logging	IM-6	Task	To Do	04/10/2021 21:46	10/04/2022 11:13	As a user I	8
Add Application categories	IM-8	Task	To Do	04/10/2021 21:48	10/04/2022 11:13	As a user I	3
Application email notifications	IM-9	Task	To Do	04/10/2021 21:49	10/04/2022 11:13		8
Timestammping of events in Applications	IM-10	Task	To Do	04/10/2021 21:49	10/04/2022 11:13	As a user I	2
Calender option in job applications	IM-11	Task	To Do	04/10/2021 21:49	10/04/2022 11:13	As a user I	8
Authentication	IM-12	Task	Done	04/10/2021 21:50	15/06/2022 10:44	As a user I	8
Application tracking dashboard	IM-15	Task	To Do	04/10/2021 21:51	10/04/2022 11:14		5
Create model for a single question	IM-18	Task	Done	17/10/2021 12:30	26/04/2022 08:51		
Create an API request to fetch all questions	IM-19	Task	Done	17/10/2021 12:35	29/03/2022 23:45		
Create an API request to fetch a single question	IM-20	Task	Done	17/10/2021 12:35	10/04/2022 11:11		
Create an API request to fetch all questions by topic	IM-21	Task	Done	17/10/2021 12:35	21/11/2021 13:13		
Design all questions page	IM-22	Task	Done	17/10/2021 12:39	10/04/2022 11:09		2
Implement design of all questions page	IM-23	Task	Done	17/10/2021 12:39	10/04/2022 11:09		5
Display all questions dynamically from DB on 'All Questions	IM-24	Task	Done	17/10/2021 12:40	29/03/2022 23:45		
Create an individual Question component to be displayed o	IM-25	Task	Done	17/10/2021 12:43	10/04/2022 10:59		
Create a 'User Response' schema	IM-27	Task	Done	17/10/2021 12:49	10/04/2022 22:56		
Design Individual Question page	IM-28	Task	Done	17/10/2021 12:50	10/04/2022 11:11		2
Implement design for Individual Question Page	IM-29	Task	Done	17/10/2021 12:50	10/04/2022 11:13		5
Individual Question	IM-30	Epic	Done	17/10/2021 12:52	03/06/2022 23:22	As a user	
Questions group features	IM-31	Epic	To Do	17/10/2021 12:58	28/05/2022 10:12	As a user I want to ha	
Profile Features	IM-32	Epic	Done	17/10/2021 13:08	15/06/2022 10:44	As a user I want my da	
Set up error handling with redux	IM-33	Task	To Do	17/10/2021 13:09	03/06/2022 19:32		5
Job Application Tracking functionality	IM-35	Epic	To Do	17/10/2021 13:13	17/10/2021 13:13	As a user I want my jo	
Add topics tabs	IM-37	Subtask	Done	24/10/2021 11:03	10/04/2022 22:53		
Create topics attribute in Question schema	IM-38	Subtask	Done	24/10/2021 11:03	13/03/2022 21:31		
Pass filtering by topics to frontend	IM-39	Subtask	Done	24/10/2021 11:04	10/04/2022 22:53		
Create an API to request questions by topic	IM-40	Subtask	Done	24/10/2021 11:05	26/03/2022 14:06		
Connect to MongoDB	IM-42	Task	Done	13/03/2022 12:11	13/03/2022 21:30		
Create git repository	IM-43	Task	Done	13/03/2022 21:29	13/03/2022 21:30		
Make individual question page responsive on mobile	IM-44	Task	Done	10/04/2022 11:01	03/06/2022 23:23		1
Design and implement profile page	IM-46	Task	Done	10/04/2022 11:26	03/06/2022 18:36		5
Dynamically get the user profile from DB	IM-47	Task	Done	10/04/2022 11:27	24/05/2022 15:27		2
Seed meaningful data	IM-48	Task	Done	10/04/2022 11:28	03/06/2022 10:50		5
Create data seeding project for all collections	IM-49	Subtask	Done	10/04/2022 11:29	02/06/2022 14:47		
Generate meaningful data	IM-50	Subtask	To Do	10/04/2022 11:29	10/04/2022 11:29		
Refactor tabs component on Question Library page	IM-51	Task	Done	10/04/2022 22:54	28/05/2022 10:05		3
Add informative logging	IM-52	Task	To Do	15/05/2022 20:37	15/06/2022 10:45		5
JWT token authentication	IM-53	Subtask	Done	15/05/2022 22:06	22/05/2022 23:06		
Refactors FE	IM-55	Task	Done	19/05/2022 22:56	28/05/2022 10:09		2
Redirect to home page from register and login page	IM-56	Subtask	Done	19/05/2022 22:56	28/05/2022 10:03		
Set up Logout functionality	IM-58	Subtask	Done	19/05/2022 22:57	22/05/2022 23:06		
Add favourite/unfavourite question functionality to questic	IM-59	Task	Done	19/05/2022 23:03	28/05/2022 10:05		3
UserSolution not being saved with correct userId and Quest	IM-60	Task	Done	22/05/2022 22:47	28/05/2022 10:09		
Password hash, encode, salt and refactor authentication to	IM-61	Task	Done	23/05/2022 19:26	03/06/2022 19:40		
Set up unit testing project BE	IM-62	Task	Done	24/05/2022 15:29	28/05/2022 21:16	Acceptan	8
Create Unit Testing scenarios for the BE	IM-63	Task	Done	24/05/2022 15:29	03/06/2022 19:40		2
Testing	IM-64	Epic	To Do	24/05/2022 15:31	28/05/2022 10:10		
Implement transaction when creating a userprofile and use	IM-65	Task	To Do	28/05/2022 09:56	15/06/2022 10:45		
Non-functional	IM-67	Epic	To Do	28/05/2022 10:08	28/05/2022 10:18		
Optional	IM-68	Epic	To Do	28/05/2022 10:16	28/05/2022 10:17		
Set up unit tests FE	IM-69	Task	To Do	28/05/2022 10:31	15/06/2022 10:46		8
Add Unit tests for questions	IM-70	Task	Done	28/05/2022 21:16	03/06/2022 19:40		5
Add unit tests for User	IM-71	Task	Done	28/05/2022 21:16	01/06/2022 19:10		5
Add unit tests for UserResponse	IM-72	Task	Done	28/05/2022 21:17	03/06/2022 19:40		5
Refactor unit tests to use an instance of MongoDb so repos	IM-73	Task	Done	30/05/2022 17:00	03/06/2022 19:40		
Add previous/next functionality to individual question page	IM-74	Task	Done	03/06/2022 10:49	03/06/2022 23:00		
Refactor individual question page and menu	IM-75	Task	Done	03/06/2022 10:50	03/06/2022 19:40		
Delete leftover pages from refactoring questions library	IM-76	Task	Done	03/06/2022 10:50	03/06/2022 19:40		
WriteUp ReadMe file	IM-77	Task	To Do	03/06/2022 19:46	15/06/2022 10:47		
Delete unused node modules	IM-78	Task	Done	03/06/2022 19:46	19/07/2022 08:10		1
Seed more questions	IM-79	Task	To Do	03/06/2022 19:47	15/06/2022 10:47		
Add references	IM-80	Task	To Do	04/06/2022 00:18	15/06/2022 10:47		
Add FE form validation	IM-81	Task	To Do	15/06/2022 10:46	15/06/2022 10:47		
Add meaningful errors on Exceptions	IM-82	Task	To Do	15/06/2022 10:46	15/06/2022 10:47		

11.4. Appendix 4 – Manual Test Results

Test Case Id: 1	Test Purpose: Verify functional requirement that a user should be able to access interview questions, answer them and view them in their profile		
Environment: **Microsoft Windows v.10.0.18363 Build 18363 Google Chrome**			
Preconditions: Be logged in the application with the following account: Email: magic@email.me Password: 123456 and be on the application homepage. AND have the Questions library open.			
Test Case Steps: Main Flow / Response Flow			
Step No	Procedure	Expected Response	**Pass/Fail**
1	Click on the “Tell me a bit about yourself” card	Be redirected to the individual question page with 3 panes as shown in sketch 1.	**Pass**
2	In the right pane write “Lorem Ipsum”	Text should appear on screen when typing.	**Pass**
3	Select “Lorem” and press the bold button 	The text should appear bold	**Pass**
4	Click the “Submit Response” button	No visible changes.	**Pass**
5	Click the  questions library icon in the nav bar	Be redirected to the “Questions Library " page	**Pass**
6	Click on the “Tell me a bit about yourself” card	Be redirected to the individual question page AND the text “Lorem Ipsum” with “Lorem” in bold should appear in the editor box on the right	**Pass**

7	Click on the profile icon  in the navigation bar	Be redirected to “My profile” page AND “Tell me a bit about yourself” should appear under “My Responses”	**Pass**
8	Click on the “Tell me a bit about yourself” card	Be redirected to the individual question page AND the text “Lorem Ipsum” with “Lorem” in bold should appear in the editor box on the right	**Pass**
9	Edit the text to say “Hi, my name is John Doe.” AND Click submit response	The text should change as you type.	Pass
10	Click the  questions library icon in the nav bar AND Click on the “Tell me a bit about yourself” card	Be redirected to the individual question page AND the text “Hi, my name is John Doe.” should appear in the editor box on the right	Pass
Related Tests: **Favourite flow**			
Author: MV		Checker: MV	

Test Case Id: 2	Test Purpose: Verify functional requirement that a user should be able to access interview questions, favourite them and view them to their profile

Environment: **Microsoft Windows v.10.0.18363 Build 18363 Google Chrome**

Preconditions:

Be logged in the application with the following account:

Email: magic@email.me

Password: 123456

and be on the application homepage.

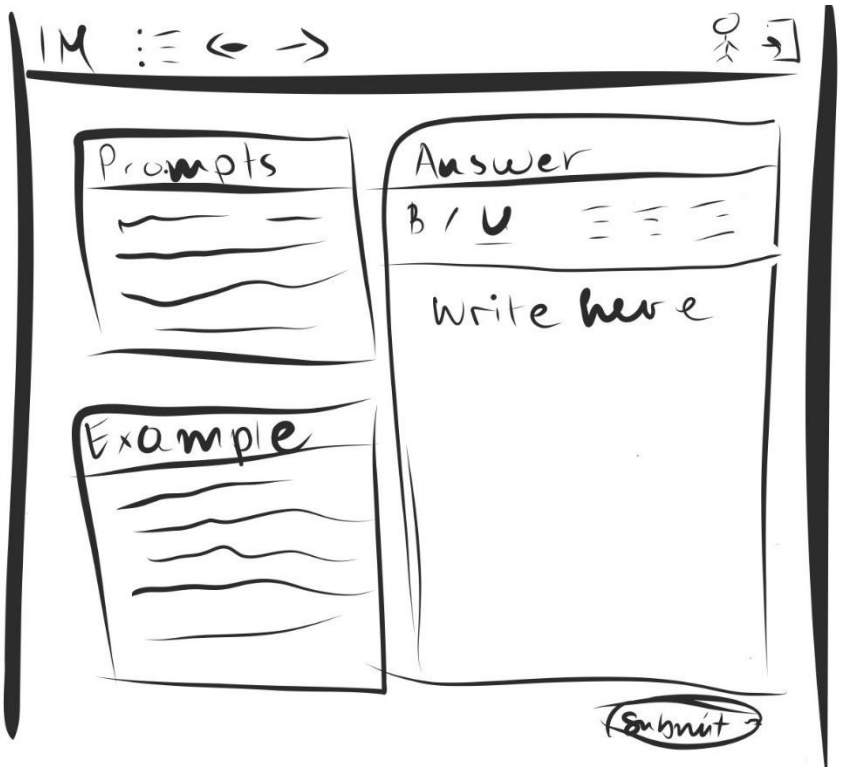
Test Case Steps: Favourite Flow

Step No	Procedure	Expected Response	**Pass/Fail**
1	Click on the 'Questions library' button	Be redirected to the "Questions Library " page	**Pass**
2	Click the ♡ icon in the right corner of the "Tell me a bit about yourself" question.	The icon is now coloured ♥	**Pass**
3	Click 'My Profile' button in the navigation bar	Under "Your Favourites" the "Tell me a bit about yourself" should appear AND the heart icon should be coloured ♥	**Fail**
4	Click on the "Tell me a bit about yourself" card	Be redirected to the individual question page with 3 panes as shown in sketch 1.	**Pass**

Comments: Step 3 failed – the icon colour is not persisted in the "My Profile" page. A bug is raised in Jira IM-108

Related Tests: **N/A**

Author: MV	Checker: MV



Sketch 1 – The individual page wireframe

Test Case Id: 3	Test Purpose: Verify functional requirement that questions should be split in topics		
Environment: **Microsoft Windows v.10.0.18363 Build 18363 Google Chrome**			
Preconditions: **Open the application as a guest and navigate to the "Question library" page**			
Test Case Steps: Topic Flow			
Step No	Procedure	Expected Response	**Pass/Fail**
1	Click on the 'General' tab	See only questions with topic 'general'	**Pass**
2	Click on the 'Collaboration' tab	See only questions with topic 'Collaboration'	**Pass**
3	Click on the 'Problem Solving' tab	See only questions with topic 'Problem Solving'	**Fail**
4	Click on the 'Adaptability' tab	See only questions with topic 'Adaptability'	**Pass**
5	Click on the 'Organisation' tab	See only questions with topic 'Organisation'	**Pass**
6	Click on the 'All' tab	See all questions.	**Pass**
Author: MV		Checker: MV	

Test Case Id: 4	Test Purpose: Verify functional requirement that a user should be able to log into a profile		
Environment: **Microsoft Windows v.10.0.18363 Build 18363 Google Chrome**			
Preconditions: Open the application homepage as a guest user.			
Test Case Steps: Login Flow			
Step No	Procedure	Expected Response	**Pass/Fail**
1	Click on the 'Login' button	Be redirected to the Login page	**Pass**
2	Fill the form: Email: magic@email.me Password: 123456	Be redirected to the home page. Navigation menu should display options "My Profile" and "Logout"	**Pass**
3	Click 'My Profile' button in the navigation bar	Be redirected to the "My Profile" page. The title should say *"Hi there, Mag!".*	**Pass**
4	Click on question "Are you a team *player?"*	Be redirected to "Individual Question" page. The navigation bar should have 5 icons.	
5	Click the exit icon at the end.	Be redirected to the homepage. In the navigation bar: 'My Profile' becomes 'Login' 'Logout' becomes 'Register' On the homepage 'Profile' button becomes Register	**Pass**
Related Tests:			
Author: MV		Checker: MV	

Test Case Id: 5	Test Purpose: Verify functional requirement that a user should be able to create a profile and access it.		
Environment: **Microsoft Windows v.10.0.18363 Build 18363 Google Chrome**			
Preconditions: Open the application homepage as a guest user.			
Test Case Steps: Register Flow			
Step No	Procedure	Expected Response	**Pass/Fail**
1	Click on the 'Register' button	Be redirected to the register page	**Pass**
2	Fill the form: Email: test@email.net Password: 123456 First Name: John Last Name: Doe	Be redirected to the home page. Navigation menu should display options "My Profile" and "Logout"	**Pass**
3	Click 'My Profile' button in the navigation bar	Be redirected to the "My Profile" page. The title should say *"Hi there, John!"*.	**Pass**
4	Click 'Logout' button in the navigation bar	Be redirected to the homepage. In the navigation bar: 'My Profile' becomes 'Login' 'Logout' becomes 'Register' On the homepage 'Profile' button becomes Register	**Pass**
5	Click 'Register' button in the navigation bar	Be redirected to the register page	**Pass**
Related Tests: **Login Flow**			
Author: MV		Checker: MV	