EMA: The Final Project Report

Developing an online catalogue and ordering system for a garment producer that can be managed from an administration portal

Graeme Fall

PI – H1725798

OUCU – ZX25798

Submitted on – 08/09/23

Due date – 11/09/23

Problem Description: Page 3 – 4

Account of Related Literature:  Page 4 – 7

Project Work – Requirements: Page 8 – 12

Project Work – Design: Page 13 – 15

Project Work – Increment 1: Page 16 – 21

Project Work – Increment 2: Page 22 – 29

Project Work – Increment 3: Page 30 – 33

Project Work – NF Requirements: Page 33 – 35

Acceptance testing & Next Steps: Page 35 – 37

Review of Project Work: Page 38 – 41

Review of Project Management: Page 42 – 45

Review of Personal Development: Page 46 – 47

References: Page 48 - 49

Appendix A – Initial Requirements: Page 50

Appendix B – Use Cases: Page 51 – 57

Appendix C – Webapp Overview: Page 58

Appendix D – Design Wireframes: Page 59

Appendix E – Gannt Chart Schedule: Page 60

Appendix F – Communication Log: Page 61 – 62

Appendix G – Selected Code: Page 63 – 65

Appendix H – Collated Testing: Page 66 – 76

Appendix I – Admin Portal Manuel: Page 77 – 84

Appendix J – Life Cycle Research: Page 85 – 86

Appendix K – Resources Table: Page 87

Developing an online catalogue and ordering system for a garment producer that can be managed from an administration portal.

**Problem Description**

*The Problem*

The client produces children’s clothing as a part-time hobby business, which mainly operates on Instagram. The client offers three garment types and over 40 fabric designs. Garments are made to order and have customisable elements which need to be specified. To order, the customer browses the clients Instagram posts to find fabrics and garments they like. The customer then messages the client to enquire about an order. If the fabric is in stock and a lead time agreed, the garment is produced and sent.

This process has several pitfalls. A customer must scroll through hundreds of Instagram posts to see all the available fabrics, resulting in most options being overlooked.

The outcome is stockpiles of fabric which are never ordered. Also, customer interactions with the client can be lengthy, as establishing what options are in stock and locating posts of alternatives is very time consuming.

The client currently preforms all administration through Instagram chat and a paper-based order system. The client has mentioned that occasionally orders are lost or sizes have been incorrect. There is no stock system and stock checks are made once an enquire is received. Again, this leads to lengthy interactions with customers when alternatives are needed.

The client takes payments through PayPal and is happy to continue this process rather than setting up ecommerce, therefore payment systems are out with the scope of this project.

The main ICT issues are there’s no centralised information online about available fabrics. The ordering system should be simpler than using social media posts and messaging. Also, having administration based online and in one place, stored securely and correctly is required to replace the unorganised paper-based system.

I have prior knowledge of these issues as the client is my sister. I know how the current system operates and have witnessed its pitfalls. Communication with the client will be vital while refining the requirements and ensuring the project is fit for purpose.

*The Solution*

The proposed solution is a browser-based web application. A customer facing section will be a catalogue of fabrics and premade products. Customers can add customised garments to a basket. When submitted, an order will be added to the orders management page and the client alerted via email. This will result in a smoother customer experience and less delay between ordering and production.

The apps administration side allows the client to manage orders and stock. The order management system will record placed orders, mark orders as paid and completed, edit orders, and manually add orders made through the old system. The client will also be able to add and remove fabrics from stock and edit their stock levels. The result being the client spends less time on administration and more time with their hobby of sewing.

The solution falls under my studies specialism route of software and web development. I have studied web development at Level 3 as well as databases and software project management. To complete this project, I will need to extend my skills and knowledge of each.

*The Impact*

I expect the project will reduce the clients’ levels of stress caused by administration and stock taking. It should make the ordering process much simpler and less reliant on the client. It is not expected that the client will take on more work due to the project, rather there will be more time to enjoy the hobby of producing garments. The client should also spend less money on unneeded fabric with the introduction of a stock system. The client should see an increase in customer satisfaction as their ordering experiences is simplified.

**Account of Related Literature**

To start work on the project solution, I first needed to develop several areas of my ICT knowledge beyond their current level. Topics such as storing and accessing the data required for the catalogue and how to display and edit admin data needed research before project work could begin. I also needed to study some non-functional requirements, like how to make a webapp useable and intuitive.

*Data Storage and Databases*

The first development decision was how to store order and stock data for the webapp. I had some prior knowledge of databases from studying TM351, although it focused on analysing established databases using database packages rather than transferring data to a website. Using databases in a web environment was new to me and needed research to ensure I implemented the right option.

Valentine (2021) discusses using databases as the focus for web design. He states the main advantage of storing web data in a database is it “gives you the flexibility to present your users dynamic content”. In the project context, dynamic content is making changes to stock in the admin portal and these being visible on the customer facing catalogue. Also, it would allow for customers to make orders and for the stock to be automatically updated. After reading Valentine, I was confident that databases were the correct technology for storing the data, compared to spreadsheets or data buckets, as databases and APIs allow the dynamic web content required for this project. Valentine is a creditable source as he’s published by Apress who specialise in IT development textbooks.

Early in the process, a major concern was storing images in a database. TM351 discussed the difficulties most databases have handling images. A relevant project constraint is zero cost, so specialist databases would be out of budget. Valentine discusses using a language called Perl. I was unfamiliar with Perl and on further investigation, learning it would have been beyond my capabilities within the project’s deadline. However, Valentine did mention data buckets as a storage method for images. I studied these during the Cloud Technologies course (TM352), so I started my research with the storage buckets user guide on AWS (2023). I learned a bucket could store any file and the file can be accessed via a URL, if the bucket is made public. In the project context, I could store an image and then add its URL to the relevant database field. As the images were of fabrics it didn’t matter if the bucket was made public. I believe this method saved me time compared to learning Perl and it fulfilled my image storage requirements. As AWS is a world leader in web services and data buckets are a service they supply, I can be confident the information is reliable.

Having overcome the image storing problem, the next step was deciding which database type best suits the project. I knew NoSQL are preferred for non-traditional data types but considering the storage bucket solution for images, a relational database would also work. I read a blog on Microsoft.com discussing the advantages of database types in a web development context. Toiba (2023) states relational databases have a better structure and “referential integrity” which makes searching databases easier, whereas NoSQL’s are unstructured. As retrieving and writing data to the project database would be a common task, referential integrity is an advantage. Toiba continues “developers will choose NoSQL databases if they needed massive scale for high throughput or very large database sizes”. The project has a very small database and little traffic, therefore the advantages of a relational database look to outweigh NoSQL. Toiba is a developer working for Microsoft and the blog is hosted by Microsoft. As they are a world leader in tech and the discussion isn’t ground-breaking, I would consider this a good source.

Next, I had to choose a specific relational database to use. With the projects “no cost” constraint, any services would need to be freeware. Therefore, using AWS for the bucket storage or databases was a non-starter but, having both services in one place would be useful for developing and maintaining the application. To supplement my Open University learning I take courses on the Udemy platform. I found a course by J. Schmedtmann (2022) called “Build a Full-Stack webapp in a Weekend”. I decided this course would refresh my web development skills and allow me to build a prototype with a creditable data storage solution. Schmedtmann uses a freeware relational database called Supabase. By working through the course and reading the Supabase (2023) user documentation, I was able to set up a working relational database connected to a webapp. Supabase also supplies storage buckets as part of their service, making it a great option for the project. An advantage of Supabase is its prewritten API’s for each database, which can be copy and pasted into the code. So, the alternative of setting up a PHP or node.js backend service is not required.

Jonas Schmedtmann is an Udemy instructor with over 1 million students. He has been teaching web development on Udemy since 2015. I believe he is a reliable source of information. Supabase user guides are clear and concise, a good source on using their services.

Overall, this database research has broadened my understand of using different databases in web development and has greatly expanded my knowledge of data storage solutions. After this research I could confidently select a database to efficiently store data and images for the project and was sure I could learn its processes within the deadline.

*Usability and Accessibility of the User Interface*

When designing the user interface for a webapp it’s important users with different needs or different sized devices find it usable. The design should be intuitive and everyone should have access to the same functionality. Although this topic has been discussed in previous courses, I’ve never designed anything that needs to meet these criteria. Therefore, I needed to research how to implement useability requirements and standards.

First, I investigated the argument for intuitive and accessible websites. Nielsen and Tahir (2002) discuss how poorly designed interfaces can make users leave a page and not return. They state average users will give a website 10 seconds and if their experience is not intuitive, they will move on. As the ordering system sends a request that includes customer specific customisation, rather than straightforward commerce, its important users are aware of the process and its intuitive. Nielson and Tahir emphasise methods to tackle this, mainly implementing a clear and uncluttered webpage and avoiding any ambiguous options. These guidelines will be used in the interface design. Nielsen and Tahir are leading researchers in web accessibility and usability, so their guidelines are an excellent source.

Legally and ethically, its important the webapp is accessible to all users. Brophy et al (2007) discuss “Accessibility for All”. They state accessibility can be achieved through new technologies, following guidelines and better practices when designing user interfaces. Again, with no budget, I will shy away from new software and technologies, focusing on better design and guideline implementation. One such design consideration is compatibility with screen readers for visually impaired users. Also, I ensured design decision helped those with partial sight. Brophy et al discuss the strategies of a less cluttered page, links being obvious and good accessibility to voice recognition systems to support these users. By following these guidelines, I will avoid breeching any accessibility laws and develop a more useable product. I believe this article to be reliable. It was found through an OU library search so will have been verified by their staff and has been used for citation 188 times.

Despite having strong guidelines to follow, I was worried their implementation could lead to compromises in the functionality of the webapp. Schmutz et al (2017) did a study into the effects of highly accessible websites versus sites with poor accessibility, where users don’t consider themselves disabled. The participants used both sites to complete tasks, and then filled out a usability questionnaire. The results showed the site with good accessibility had higher levels of task completion, quicker completion, and higher satisfaction ratings. This indicates a webapp designed to accessibility and inclusivity guidelines not only helps disabled users but benefits all users. Therefore, I felt confident implementing these guidelines wouldn’t hinder the functional requirements. This article is a good source as it’s been peer reviewed and is supported by serval past studies that came to the same conclusion.

Before interface design began, I wanted to find more usability guidelines for web development. Huber and Vitouch (2008) wrote a paper on this subject. They present a major guideline that states, “A useable system should be easy to learn, efficient to use, easy to remember and have a low error rate”. They say this is the key to higher user satisfaction when designing an interface.

By combining Huber and Vitouch’s usability guidelines with the W3C’s (2019) accessibility principals, I was better prepared to design an interface that delivers an excellent user experience for both able and disabled users.

Huber and Vitouch can be considered a good source as their paper is peer review and available on the OU Library website. W3C’s accessibility principals are also an excellent source as W3C are leaders in developing better accessibly for the internet.

This research was the basis of my user interface design. The webapp has an unfussy colour scheme and obvious buttons, links, and menu options. Anything that is not fully intuitive has been well explained. As the catalogue can be viewed on different devices, these pages have been made reactive to mobile screens and remain uncluttered and usable. By following these researched guidelines, I have delivered an application everyone can easily use.

*Client-Side Storage*

After completing the prototype system, I became aware I was going to need a solution to store customer basket data. A relational database wouldn’t deal with temporary data, so I tried using variables but when the page reloaded the basket was lost. I had to find a way to make the basket items persistent. After some research I found examples of using the browsers local storage to hold information. As this was a new concept, I wanted to check it was an accepted practice.

As the initial website wasn’t a solid source (although correct) I did some further research. I looked for possible issues and found a Mulligan and Schwartz (2000) paper on privacy concerns with client-side storage. They state the main risks are when personal information is being stored but for basic storage (such as a basket) there are no security concerns. Having read this, I was confident implementing client sided storage to my code. Mulligan and Schwartz’s paper is a creditable source as it comes from the ACM library.

**Project Work and Outcomes**

To develop the project solution, I have worked through a process of requirements gathering, design, implementation, and tests. I will discuss how each of these have been implemented and their contribution to the overall solution.

Requirements – Initial Discussions and Requirement Elicitation

During the initial client meetings, we discussed in detail what the current system did do, didn’t do, and what could be improved. It was decided that the web application solution should have three main tasks.

1. It should improve the customer experience via a catalogue and ordering system.
2. It should make admin simpler and less time consuming by implementing an admin portal to manage stock and existing orders.
3. It should have a shop front for premade (non-custom) items to allow instant orders of premade garments with no lead time.

Each of these were refined into smaller tasks which make up the initial requirements list in Appendix A. These were developed into the functional requirements in the Volere Template. The Volere template gives an overview of all the requirements and constraints for this project.

Requirements – Refined Requirements in a Volere Template

*Stakeholders*

The main stakeholder is the client. The client is the sole member of the small business so does all garment production and administration. The end users are customers who will use the service to order garments. It’s assumed both end users and client can use a web browser, but the application should be as intuitive as possible.

*Project Constraints*

The client has requested the web application is accessible to all users and implements a bright unfussy colour scheme. The client would like the application to be hosted remotely and only the client and developer have access to the administration portal. The business is a hobby, so it’s important software and services are freeware. The client requests the webapp works on all major computers-based browsers and orders can be made on mobile devices.

The client has indicated naming conventions to be used throughout the project:

Garment – this is the type of clothing. The client makes 3 types of garments: A hoody, a tank hoody (without sleeves) and a snood.

Fabric – This is the patterned fabric that the client has in stock. Fabric is stored on rolls and is measured by the meter (m).

Bands – These are the blocks of solid colour fabric which border the patterned fabric. These and the cuffs colours are the customisable options for hoodies and tanks.

Fleece – This is the lining on the inside of a snood. The colour of this is the customisable option for snoods.

Premade item – This is a garment that has already been made and is not customisable but is available for immediate dispatch.

A relevant fact is the turnaround for a snood is 1 week and 4 weeks for hoodies. The client can make 20 snoods in a week and 2 hoodies a month. Orders beyond this currently go on a waitlist. The waitlist is currently outside the project scope.

*Functional Requirements*

The clients’ garments business is a small hobby business that has very little baring on the clothing industry. The webapp will deliver a better customer experience and allow the client to organise administrative task easier. There shouldn’t be a requirement for more than basic performance usage as the site will not see user spikes.

The data required from the client will be details of stock. Each fabric requires: a photo, name, description, and a stock length. The application creates data about an order. Each order item requires: an order number, garment, fabric, size, custom colours, fleece colour, customer name, email and postcode. All personal details will be held behind passwords and polices. No payment data will be held by the client. Payments will be made by PayPal requests, out with the project scope.

There are several functional requirements agreed with the client that the webapp must meet to be successful:

FR1 - The application should display all garments and fabrics in a browsable catalogue. It should show all relevant details (pictures, name, description etc).

FR2 - A selected garment and fabric can be added to a basket via an order form which allows customers to customise their garment.

FR3 - The basket can be reviewed and then submitted to the client as an order with the customers details.

FR4 - The client will receive an email to alert them of a new order. The email will have details of the order.

FR5 - Items from the order will be added to the current order list in the order administration portal.

FR6 - The client can add manual orders or edit current orders in the administration portal. The orders can also be marked as paid or completed.

FR7 - The client can manage stock from the stock administration portal, viewing all fabrics and level of stock. New fabrics can be added, and stock numbers can be updated.

FR8 - When an order is made the length of fabric needed becomes reserved.

FR9 - When an order is marked as complete the length of fabric is removed from stock.

FR10 - Each fabric will show if it is in stock, low stock or out of stock in the catalogue.

FR11 - The application will have a catalogue of premade items.

FR12 - The stock of premade items can be managed from the administration page. New items can be added.

FR13 -When a premade item is ordered it is hidden from the catalogue. Once payment is made it will be removed from the catalogue.

When the client receives an order, the customer is emailed to confirm cost and timescale. The client also requests payment and a delivery address. This process is outside the scope of the project but added for context.

Each functional requirement has an accompanying Use Case found in Appendix B. The Use Case outlines the main design steps and constraints for each functional requirement. The Use Cases are also used to aid testing by applying Fit Case’s to each requirement during testing.

*Non-functional requirements*

The web application should have a clean and uncluttered design. The target customers are parent’s buying clothes from their children, so fonts and colours should reflect this. The application should be usable and intuitive to all users and no assumptions of technical knowledge should be made. Anyone that has used a web page to buy, or browse should be able to place an order.

The application will not require high performance. The client currently has a low volume of custom, which is not expected to rise beyond basic performance needs. It is important the administration portal is only accessible to the client and that customer details stored on the database are password protected. It would be ideal if availability was constant but short breaks for updates will not be an issue. It’s important that order details are accurate, and the application reliably sends email alerts to the client. The client should never need to directly access the database or the code, all changes should be possible using the webapp. The webapp needs good maintainability as changes to the catalogues and orders will be managed from the admin portal.

*Legal, Social, Ethical and Professional Issues*

There are four main project stakeholders that could be affected by LSEPI’s. The client (sole business owner), the developer (me), the users (customers) and the named fabric designers.

When the webapp is operational, the client will have a more efficient workload. This will provide a positive social impact for the client, due to having more time to spend with family and less time working. If the webapp becomes popular and more orders are made this could have a detrimental effect on the stress levels and mental health of the client. I have suggested implementing a method of closing orders if workload becomes too much. This could be a button in the administration portal to toggle accepting orders. The client has asked for this to be added after the core functionality.

The webapp should allow customers who can’t use social media due to accessibility issues to browse and order items. Ethically the client is very keen for products to be “Accessible by All”. By developing a webapp with good accessibility, we will be positively impacting these users.

The fabric designer may benefit from having their designs displayed in the catalogue. This could result in new work or sales independent of the webapp. A risk the designer faces is an association with any mistakes or unintended ethical/legal issues made by the client or developer. This shouldn’t be an issue as the client, and I have gone to great lengths to ensure the webapp is ethically and legally sound during our frequent meetings.

There are several legal requirements when developing a webapp. These are outlined by G. Harris (2021) on the Scope website. He mentions all webapp must adhere to the Equality act of 2010 and the disability discrimination act of 1995. He states a good way to implement this is adhering to the W3C guidelines and standards on accessibility. This has been actioned throughout planning and design with the goal of access for all and no discrimination.

Another legal requirement regards customers personal data. Personal data must be held in accordance with the GDPR (2018). To ensure this, the database must have suitable polices in place and personal details are only displayed in the administration portal which is password protected. The client is wary of holding any information about payments. Therefore, ecommerce has been removed from the scope of the project. I don’t believe there’s any concern that users can be tracked, as the only location information held is a postcode.

It was decided to have no images of models in the catalogue. The client did suggest garments worn by models, most of whom would be under 18. We decided for the online safety of minors, it would be better to have no models at all. This safeguards against any unwanted misuse and future embracement for a child model.

Professionally I have abided to the BCS code of Conduct by following the guidelines closely. A first example is avoiding conflict of interests with the client (3b). I have ensured that the clients’ interests are the project focus and my requirement for assessable work doesn’t detract from the clients needs.

The client is comfortable with my skill set and the project schedule (2a & 2b). They are aware I need to develop my knowledge and understanding of concepts during the project and the client is happy with the associated risks. I believe I haven’t misled the client with my skill set and as the project continues, I will develop all necessary skills required to be BCS compliant.

The client is also happy to use the old system until completion. If the project is not finished, the client will continue to use the old system until a way forward is achieved. I have ensured there is no cost to the client during development and deployment, which they are happy with (3c).

The BSC code of conduct highlights the importance of projects outcomes being fit for purpose(4a). I believe the requirement agreed ensure a suitable solution for the client. Also, in relation to the BCS code, all the resources I need are allocated, and I believe all tools selected are appropriate for the project, none of which are pirated(2d). If at any point some functionality looked too difficult to implement or couldn’t be done on time, an immediate conversation with the client would take place to find a solution (3c).

*Equality, Diversity and Inclusion Concerns*

It’s vital to make “Accessibility for All” the focus of the user interface design. Both legally and ethically it’s important users with disabilities can access the catalogue and place orders. The client is keen for equality of access for all users and that no user feel forgotten about in the applications development. To ensure this the W3C accessibility guidelines have been constantly referred to while making interface design decision. After implementation I tested the application with a screen reader and used a usability test to ensure it had been designed correctly.

It's also important to mitigate any unconscious bias that myself or the client might demonstrate during the project’s development. We have ensured no gender or ethnic background feel marginalised or underrepresented in the webapp. One strategy implemented was removing models for the catalogues, all images are of the products or samples of fabrics. Also, we have avoided grouping fabrics designs into stereotypical genders in the catalogue, so no bias is given to the intended wearer of each fabric.

Another strategy to minimise potential bias is to use the toolkits suggested in the OU course work. I have use information from the GenderMag Kit and the Beyond pink and blue checklists to ensure the webapp is as inclusive as possible.

Overall, I believe myself and the client have collaborated well to ensure that the application is inclusive and accessible.

*Project issues*

This is a new build, with no existing webapp or components in place. My research indicates several “off the shelf” solutions that can help in development. Packages like Supabase, a free to use database with built in API, will save time and effort by negating the need to building a new backend. Also, the JavaScript library React has been useful in structuring the webapp. Another example is the email client EmailJS which is used to alert the client of orders.

There should be no cost to the client in developing and implementing this project. All resources will be freeware and all hosting will be free. Therefore, budget over run will not be an issue.

The full risk assessment for the project can be seen in Table 9 on page 44.

**Design**

With requirements agreed, the first design aspect discussed was the structure and layout of the webapp. After the early discussions I developed an overview of the application (Appendix C) to map out what pages were required and how they would interact with one another. This was developed into a more refined designs as shown in the wireframes (Appendix D) which were the basis of the HTML for the user interface. CSS was implemented to meet the non-functional usability requirements and accessibility guidelines. JavaScript was used to make pages more dynamic and to ensure links and forms behave as expected.

A key design consideration was which Lifecyle model to use. After evaluating each type, I decided to use a hybrid of the incremental and Scrum approaches. This would group the requirements into three increments, with each requirement within approached like a scrum sprint. This sprint would be tested, and feedback would be gathered from the client. Using this feedback, the requirement would be iterated until the client was happy and accepted its implementation. Communication with the client was vital for this approach to work. It was agreed that I would update the client with demo videos of functionality updates, and we would have a zoom meeting every 2 weeks to discuss the work. These interactions have been collated from my project log and WhatsApp in Appendix F.

The implementation increments echo the main tasks discussed in the requirements phase:

Increment 1 – Catalogue and ordering system. Including set up of databases.

Increment 2 – Administration portal.

Increment 3 – Premade Items and User interface.

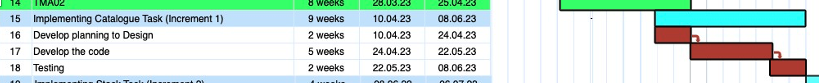
The structure of these increments is shown in the project schedule’s Gantt chart in Appendix E. The chart shows each increment split into planning, implementation, and testing. Below is an example for Increment 1. There is little coupling between these increments so they can be worked on in order without having to jump between tasks.

Figure 1. Increment 1 from Gantt Chart in Appendix E

Each functional requirement has an accompanying Use Case found in Appendix B. The Use Case outlines the main design steps and constraints for each functional requirement. The Use Cases bridge the gap between design and implementation. The Use Cases are also used to aid testing, as each functional requirement has Fit Cases derived from its Use Cases. These ensure the implementation matches the design requirements.

A screen shot of a computer

Description automatically generated

Figure 2. Example of Use Case from Appendix B

Although there is low coupling between functional requirements, many tasks do share the databases. Before implementing code, it was important to design the databases and their connections to the webapp.

To aid this process I developed the below class diagram (Figure 3). This helped me determine what data was needing held in which databases and what might be temporary data. The methods in the “backend” section correlate to the functional requirements of that page. With the planning and designs in place I was ready to start implementing the application.

Diagram

Description automatically generatedFigure 3. Data Flow/Class Diagram

**Implementation – Increment 1: Fabric Catalogue and Ordering System**

Before tackling the functional requirements, I first had to decide the webapps structure. Having run a prototype using methods from Schmedtmann’s course, I decided to use React Libraries to build the webapp. React uses a set structure to allow webapps to be more dynamic and interactive, which benefited both the catalogue and administration portal. Building a page by calling smaller components was new to me and a good skill to pick up. The top level React structure is in code snippet 1 in Appendix G. As React was completely new to me, I made extensive use of Reacts official documentation and examples from StackOverflow.com. This was combined with HTML, CSS and Java Script to code the website structure.

As discussed in the literature review, I decided to use Supabase to host my relational database and my images in a storage bucket. I implemented the necessary tables and folder to store the required data. To start with I manually entered test data, but as the project developed, I was able to add, edit and delete from the database using API’s and forms I designed in the webapp.

I also built the initial pages, the “home” and “products” page were coded. Their implementation used HTML, CSS and API calls to collect stored images. Next, I worked in smaller sprints to complete each of the functional requirements for this increment.

*FR1 - The application should display all garments and fabrics in a browsable catalogue. It should show all relevant details (pictures, name, description etc).*

A screenshot of a computer

Description automatically generatedThe Catalogue is populated by retrieving all data from the Stock database. If a fabric is added or removed this is reflected in the catalogue when the page is refreshed. The text, name and stock levels are stored in the Stock database along with the URL of the image in the public data bucket. Below is a screen shot of the “Stock” table from the database.

Figure 4. Stock Table from Database

A screenshot of a computer screen

Description automatically generatedThe order buttons are automatically hidden if there’s insufficient stock. The stock indicator shows one of three levels (In stock, Out of stock and Low stock) which inform the customer of availability of the fabrics. Together these display the full catalogue of fabrics found in the database as requested in the functional requirements (Figure 5).

Figure 5. Fabrics Catalogue Page

A screen shot of a black and white screen

Description automatically generatedTo ensure the functional requirements are satisfied, FR1 has met a Fit Case test and black box testing to ensure boundary data behaved as expected (Table 1). At the end of each sprint, I have employed client acceptance testing and feedback. Full records of these test are found in Appendix H.

Table 1. Black Box testing for Button Showing Boundary Behaviour

*FR2 - A selected garment and fabric can be added to a basket via an order form which allows customers to customise their garment.*

When the order button is pressed the “Garments order form” overlays the screen (Figure 6). The customer can then select customisation options from the drop-down menus. Then the customer can add the order to the basket.



Figure 6. Order Form

A black and white screen

Description automatically generated with medium confidenceThe basket is saved in local storage as the data is only needed temporarily (Figure 7). The customer can add as many items to the basket as they wish. These can be reviewed on the basket page of the app (Figure 8 in FR3).

Figure 7. Local storage showing basket.

Blackbox testing of sample orders were successfully completed and the Fit Case tests were met. The client’s feedback during acceptance testing requested the fabric name to be present on the form and the form to be bigger than originally sized (both implemented in Figure 6). Full testing results are in Appendix H.

*FR3 - The basket can be reviewed and then submitted to the client as an order with the customers details.*

The customer can access the basket page and check their basket items. The basket can be cleared if there are any errors. The basket and form only appear if there are items in the basket. This is achieved using React useStates and a conditional IF statement dependent on the Boolean variable in the useState. The customer needs to add their personal details to the form before sending the order request to the client (Figure 8).

A screenshot of a computer

Description automatically generated

Figure 8. Basket Page with Details of Order

The personal details are checked in the code to ensure valid data, then the order items are sent to the Orders database via an API. Each item in an order is given the same order number (Figure 9) so these items can be grouped in the Order admin.

A screenshot of a computer

Description automatically generated

Figure 9. Order Database After Order Sent (Bottom 2)

The Fit Case tests have been met and client feedback was used to remove unnecessary customer data (full address). The black box testing checked for valid data (Table 2). FR3’s full testing in Appendix H.

Note – Hiding the “Personal data form” when the basket was empty happened due to feedback from acceptance/handover testing. Therefore test 2 below is now defunct as no personal data can be entered.

A black and blue box with white text

Description automatically generatedTable 2. Black Box Testing for Valid Personal Details

*FR4 - The client will receive an email to alert them of a new order. The email will have details of the order.*

When a valid order is sent, an email is automatically generated and delivered to the client, alerting them of a new order. This email includes details of the order and the customers email address (Figure 10).

A screenshot of a computer

Description automatically generated

Figure 10. Generated Alert Email

A 3rd party email service called EmailJS was required. The order data is sent to the email client and an email is generated. This service is free up to 200 emails per month, which the client doesn’t anticipate reaching. The code can be seen in code snippet 2 of Appendix G.

The Fit Case Test below is consistently met and therefore works successfully. Full testing report in Appendix H.

Description:

UC4 Step 4 - The email is sent to the client, informing them of a new order

Fit Case:

Client received email of each order to the business email account with all relevant details.

**Implementation – Increment 2: Administration portal**

The second Increment focuses on the administration portal, aiding the client to maintain the webapp and managing orders and stock. The administration portal is made up of five pages. In this iteration three are developed: an admin home, a stock management section, and an order management section. The login and premade items management are delt with in iteration three.

First the home page for the administration portal was made. This consist of links to each of the other admin pages and can be returned to from each without logging out.

Again, each functional requirement was tackled in a sprint. Testing and client feedback was used to improve the code until the functional requirement was accepted.

*FR5 - Items from the order will be added to the current order list in the order administration portal*

A screenshot of a computer screen

Description automatically generatedThe main body of the order management page is split into two tables, one for open orders and the other completed orders. All order items are gathered from the Orders database (Figure 9) by API call and filtered into one of the tables, depending on the state of the “Complete” cell (yes or no). As new orders are stored in the Orders database they are automatically added. The tables are sorted by order number so items from the same orders are displayed together. Figure 11 shows the admin orders page.

Figure 11. Admin Order Management Page

The main Fit Case test was consistently met as all items are correctly filtered

Description:

UC5 Step 3 - The system will sort all order items into two tables, current or complete.

Fit Case:

All orders not marked as complete are in the current orders table, all marked complete are in the completed table.

FR5 has also had acceptance testing. A suggestion early was to keep orders grouped by order number which has been implemented. Full testing documentation is found in Appendix H.

*FR6 - The client can add manual orders or edit current orders in the administration portal. The orders can also be marked as paid or completed.*

A screenshot of a computer

Description automatically generatedEach order in the tables can be edited, marked as paid and complete or deleted from the database. To edit the client clicks the “Edit” button at the end of the item’s row. The edit form (Figure 12) is prepopulated with the items current values and the client can make changes which update the database via an API update call.

Figure 12. Edit Form with Changes to Order

These changes can be seen when the page is refreshed. In Figure 13 you can see the order in Figure 12 has been moved to the completed orders table and the top band colour has been changed to yellow from blue.

A screenshot of a computer

Description automatically generated

Figure 13. Edited Admin Order (Order #992400)

A screenshot of a computer screen

Description automatically generatedAdding a manual order is a similar process. The “new order” button is pressed, and a blank version of the edit form is presented. The client fills out the form with valid inputs and then submits. The new order item will be seen in the tables on the next refresh, in Figure 14 this is the order #510713 which has been added to the current orders list from Figure 13.

Figure 14. New Manual Order in Current Orders

The above screenshots show that each of the Fit Cases have been satisfied for FR6.

All non-drop-down inputs must be valid (length, not null etc) which is tested in Table 3 below.

A screenshot of a computer screen

Description automatically generated

Table 3. Black Box Testing for Valid Inputs of Edit and Manual Order Form

*FR7 - The client can manage stock from the stock administration portal, viewing all fabrics and level of stock. New fabrics can be added, and stock numbers can be updated.*

A screenshot of a computer

Description automatically generatedThe Stock Management pages data is retrieved form the “Stock” database and displayed in the table. Each fabric type has its name, picture, description, reserved stock, and stock amount viewable. (Figure 15).

Figure 15. Stock Management Page

New items of stock can be added using the Add Stock form (Figure 16). This asks for the name, description (max 120 characters), initial stock and an image file for the new fabric. The image can be uploaded from the client’s computer. When submitted the image is added to the Supabase Storage bucket via API. The webapp then uses an API to requests the new URL of the image from the storage bucket. The webapp then sends the URL and other fabric details to create a new fabric item in the “Stock” database. This new stock item will be seen in the stock list and the catalogue.

A screenshot of a computer

Description automatically generatedFigure 16. Add Stock Form

A screenshot of a computer

Description automatically generatedEach stock item has an edit button on the right-hand side of the table. This will present a prepopulated form which can be used to update a current fabric. In Figure 17 I have updated the newly added Blue Lions fabric (Figure 16) to have 12 meters of stock and 4 in reserve. Figure 18 shows Blue Lions in the table with its updated figures. These screenshots prove that the Fit Case testing has been satisfied. Full testing in Appendix H.

A screenshot of a computer

Description automatically generatedFigure 17. Edit Stock Form

Figure 18. New Fabric Added and Visible Edits

*FR8 - When an order is made the length of fabric needed becomes reserved.*

*FR9 - When an order is marked as complete the length of fabric is removed from stock.*

When a customer submits an order the length of fabric for the item will become reserved and added to the total in the “reserved” column in stock management (Figure 15).

Depending on the garment this will add either 2m (Hoody/Tank) or 0.5m (Snood) to the reserved total. This is implemented using Java Script and different function calls depending on an IF condition. This will affect the total available stock (stock minus reserved) which can make a difference to the stock indicator in the catalogue (FR10).

The “Complete” status can be changed with the Edit form (figure 12) on the Order Management page. When this happens, the order will move to the complete table (like in figure 13) and the amount reserved is removed from both the “Reserved” and “Stock” columns on the Stock Management page for that fabric.

This black box testing checks the stock and reserved totals when orders made or complete state changes.

A black and blue chart with white text

Description automatically generated

Table 4. Black box Testing of Changes to Stock and Reserved Totals

Black box testing was also done on the movement of completed items in the order management. After client feedback the initial design of tick boxes to show state of items displayed in a single table was replaced with no tick boxes and two tables to manage orders. The full testing report for FR8 and FR9 are in Appendix H.

*FR10 - Each fabric will show if it is in stock, low stock or out of stock in the catalogue.*

The stock indicator allows customers to see if a fabric is low in stock and will remove the option to order a fabric with insufficient available stock (stock minus reserved). Figure 5 (FR1) shows items that are out of stock, have more that 5 meters left (in Stock) and low stock (less than 5 available meters).

A screen shot of a computer code

Description automatically generatedFigure 16. Code Snippet from fabricItems.js for Stock indicator

The code snippet above shows the conditions for each for the indicators. The colours are added in CSS by associating colours to the class names of each <p> section.

A black and blue table with white text

Description automatically generatedFR10 was tested using Fit Cases tests and black box testing on the boundaries of fabrics stock (Table 5). Full testing report in Appendix H.

Table 5. Black box Testing of Stock Indicator

**Implementation – Increment 3: Premade Items and Non-Functional Requirements**

The last increment focuses on adding premade catalogue (no customisation) and a premade stock management page in the admin portal. I will also implement usability strategies to ensure the application is “Usable by All” and implement a login page with some basic security for the admin portal.

*FR11 - The application will have a catalogue of premade items*

A screenshot of a computer

Description automatically generatedThe implantation of FR11 is very similar to the implementation of FR1 (catalogue of fabrics). The page displays a picture of each premade item, a description and a button that will add the item to the basket. Figure 17 shows the premade stock database and Figure 18 shows the corresponding items on the live webapp (called by API).

A screenshot of a website

Description automatically generatedFigure 17. Premade Stock Table in Database

Figure 18. Premade Items Catalogue

The screenshots show the Fit Case testing (derived from UC11 Step 2 in appendix B) of “The system will retrieve all details from the pre-made items database and display items” has been successful and therefore FR11 is functional. After client feedback the “order item” button was moved into the row, where before it was below the item.

*FR12 - The stock of premade items can be managed from the administration page. New items can be added.*

A screen shot of a computer

Description automatically generatedAgain, FR12 is very similar in design and code to FR5, FR6 and FR7. A lot of the code from these outcomes has been reused and tweaked to meet the needs of FR12. In Figure 19 each item from the database in Figure 18 is shown in the table and each item can be edited. Adding new stock is again done via a form found at the bottom of the page.

Figure 19. Premade Stock Admin Page

Although the fundamental code and design is similar to earlier functional requirements, it is good practice to be able to reuse code and make small adjustments to make new functionality work. In this context reusing the image storing and calling code from FR7 saved a lot of time and effort.

The below Fit Case testing ensured that all the details were stored in the database and that the item showed in the catalogue.

Description:

UC12 step 6 & 7 - The details of the new item (including URL) are sent to the Premade Database. When refreshed the new item is in the premade table and is currently displayed on the premade items page.

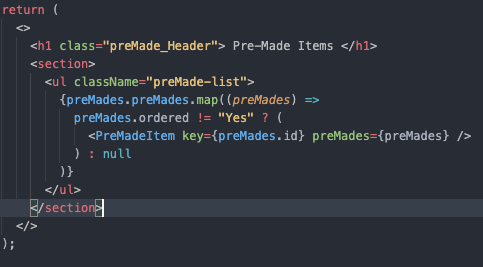
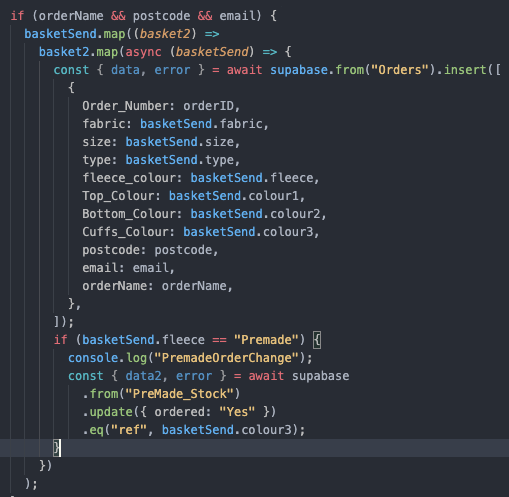
Fit Case:

The new Premade Stock item is shown in the admin page and is also shown in the catalogue.

*FR13 -When a premade item is ordered it is hidden from the catalogue. Once payment is made it will be removed from the catalogue.*

This last functional requirement ensures that a premade item is not ordered twice. When a premade item is summitted with an order its hidden in the catalogue. This is achieved by giving each premade item an “ordered” tag. Once an item is ordered the tag changes to “Yes” and when the premade catalogue is being rendered it maps every item but only displays those that have “No” in their tag. The two main code snippets are in figure 20.

Figure 20. Code Snippets from Basket Sending (right) and Premade Catalogue (left)



These code snippets demonstrate new skills I have developed in API’s, Mapping and React during the project.

A black and blue rectangular box with white text

Description automatically generatedAfter testing I was able to confirm that this met the Fit Case for the requirement and the black box testing (Table 6) shows that the functionality has been successful.

Table 6. Black Box Testing of Hiding Premade Items

**Non-functional Requirements – User Interface and Reactive Design for Mobile.**

A key non-functional requirement is ensuring the webapp is intuitive to use and that customers with disabilities can browses and order items using the app. The research behind this is covered in the Literature Accounts section but I will demonstrate its implementation in the user interface.

An important useability guideline is the interface should be uncluttered and intuitive to navigate. In the screenshots of the fabric catalogue (Figure 5) and the premade items (figure 18) each item is given is own defined section. These sections are well spaced and the content of each is aligned to give a neat display of the items. Care was taken to ensure all text colour is contrasting with the background which makes the page easier to read. I also ensured that table colour schemes were usable using Pigment (2023), a webapp developed by Shapefactory to suggest usable colour combinations. I found this to be an excellent source of colour matching/contrasting.

To allow intuitive navigation, I have followed guidelines about clear labelling of links and buttons on the page. I have made an obvious navigation bar in an intuitive position and all buttons are distinct from their surroundings and appropriately named. On the order form, each drop down has been given an appropriate name so a customer will know exactly what they are customising (figure 21).

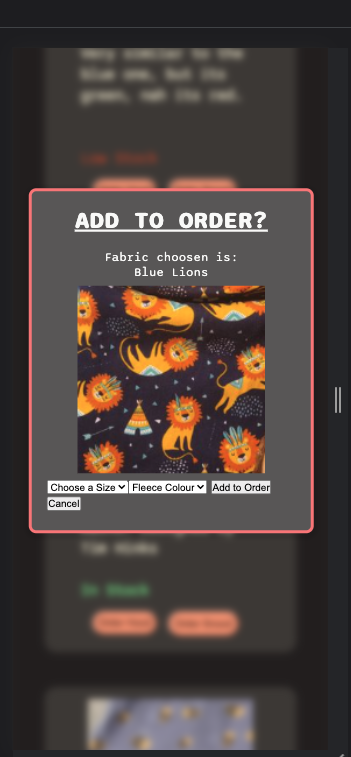
*A screen shot of a website

Description automatically generated*Figure 21. Order Form Customisation Dropdowns

To ensure the webapp is usable for blind customers, I have tagged images with good alt text descriptions. Also, I have discussed implementing a “Standard fabric discerption script” with the client, so when they add descriptions, they are useful to blind users. To test this, I have successfully used a screen reader to check for errors in the flow of the page design.

The client has requested the webapp works on mobile devices. I have adopted a reactive approach so when the screen size is reduced the pages features are rearranged using CSS. In Figure 22 the chrome developer tools has been used to mimic a mobile phone screen. At this size the navigation bar changes to a hamburger menu. Fabrics are displayed one to a row and the order form has been scaled down to fit the screen. The dropdown options are displayed using the mobiles interface which is more intuitive. The client decided they will not do admin on a mobile, so the admin section has not been made reactive.

Figure 22. 3 Screenshots at Mobile Phone Screen Size



*Non-Functional Requirement – Security - Admin Login*

Legally it is important to keep customers data safe. It is also important that the integrity of the data is maintained and no one maliciously gains access to the admin portal. To implement this, I designed a login page with a password and username.

I have stored the username and password in the database, so they are not stored in the code and are not visible if inspected. Once logged in the client can use the link buttons in the admin portal to navigate. If the page is refreshed the client is logged out and needs to re-verify their login details. There is a simple token system in place so if someone tries to access any admin page direct, they will be redirected to the login page (figure 23).

A screenshot of a computer

Description automatically generatedFigure 23. Admin Login Page

This system has worked well and will deter everyday users from trying to access the admin portal. I would like to build a proper backend with node.js. This would allow a more secure token system but learning to do this was not possible within the project deadline.

**Client Handover, Acceptance Testing and Tasks to be Completed**

The webapp has been hosted on a free hosting site so acceptance testing can take place. I tested the customer front end with 2 of the client’s customers and 2 family members. The admin section was only tested by the client. I produced the scenarios below (Tables 7 & 8) and asked for general feedback.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenario | Expected Result | Tester 1 | Tester 2 | Tester 3 | Tester 4 | Feedback |
| Order a Snood, Blue lining, Size Teen | Client receives an order email for teen snood with blue lining. Correct customer details to email response. | Client follow-up Email received | Client follow-up Email received | Client follow-up Email received | Client follow-up Email received | Process straight forward. Not immediately obvious to order in Fabrics section. Basket could do with more directions. |
| Check the lengths for a 12-14y hoodie | Customer finds the right size. | Size Found | Size Found | Size Found | Size Found | Easy to find sizes. Tables slightly overlap when browser window made smaller on laptops. Good on browser and phone. In-between needs work. |
| Order a tank hoodie, Size 2-3y, Bands all red. | Client receives an order email for Tank hood with red bands. Correct customer details so can give email response. | Client Email received | Couldn’t find tank. | Client Email received | Client Email received | Tester 2 could not find the tank option. The order buttons are for hood and snood. Tank option is in the hood custom options. Clear instructions needed. |
| Add an item to the basket and then remove it. | Customer can easily clear the basket. | Yes | Yes | Yes | Yes | Clear operation on the button. Whole basket emptied, would like option to remove item at a time. |
| Order a premade Item | Client receives an order email for item. Customer receives email from client. | Client Email received | Client Email received | Client Email received | Client Email received | Easy to add to order. More information about how a premade item order differs to a customer order. |

Table 7. Customer Feedback Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Scenario | Expected Result | Client Result | Feedback |
| Login to Admin Section and use buttons to navigate. | Details used are correct and all admin pages can be accessed | Login achieved. All pages accessed. | Simple to use but forgot to use buttons a few times (using browser back) and needed to log back in. |
| Change an order from not complete to complete | Order should change table in the orders admin page using the edit button and form. | Successful | Good functionality, happy with how it works. |
| Add a new fabric to the stock list. | Use new fabric button and add valid details. Item then seen in catalogue and admin. | Successful after changing picture type to .png | Client forgot the image type needed. Perhaps a small manual needed. |
| Edit a premade items price | Use edit button and change description on form. Change seen in catalogue | Successful | Client happy with functionality. |
| Delete a fabric | Use edit button and then delete. | Successful after finding button | Once found done simply. Manual would help. |

Table 8. Client Admin Portal Testing

Next Steps

From this testing these revisions and improvements should be implemented:

* More detail describing how to order and the differences between order types.
* Clear instructions on how to order a tank hoody.
* Check responsive borders between mobile and desktop and edit CSS code accordingly.
* Remove link to Admin Pages and access through URL.
* Better solution for security where token persists for a time rather than refresh/logout.
* Develop manual so client can read if any issues arise. (Actioned, see Appendix I)

Other discussed next steps

* Add a button to disable orders and/or display warning of lead time, if orders are currently being taken or a waitlist is in action. Both managed from admin page.
* Add item deletion to the Basket list rather than clearing the whole basket.
* Refine some of the Mobile Reactive pages. Although they are functional, a few things could be changed or added. (Size tables currently not shown)
* Decision to keep free hosting (with hosts name in URL) or pay for URL without hosts’ name.

Each of these steps will be completed after the submission of this report. Overall, the feedback indicates I thought the app was more intuitive to users than it is. This demonstrates the importance of acceptance testing.

**Review of Project Work and Skills developed**

Throughout this project, I have encountered challenges and frustrations. When working through these challenges, some solutions have been a success, and others have been difficult to overcome.

A successful action was building a prototype application using the supplementary Udemy course. This helped prepared me for the project and gave me an insight into concepts and hurdles I would face. The course supported me with step-by-step tutorials that helped me get to grips with techniques like sending and calling API’s to a database, using React hooks, and how to host the finished application. By completing this single page React application, I could dry run a database option. This gave me the confidence to continue with the database in the project. Below is a project log showing my initial issues.

A black and white text on a black background

Description automatically generated with low confidence

Figure 24. Project Log 13/3/23

A persistent issue throughout the project was understanding the complexities of React libraries. The first issue was turning the one-page prototype into a multipage dynamic app. This wasn’t as simple as calling complete pages, as React uses a more complex component-based system. I did eventually get my head around this but later I started to struggle with the way React renders and reloads the page using hooks. I often found changes made to variables during code events weren’t visible when I expected or were “one cycle” behind.

As I started to understand how these hooks and useStates should be used, I already had most of the webapp structure built. It’s something that should be ingrained into the initial design to work effectively, and I didn’t have the experience to implement it this way. As you can see from the project log in Figure 25, I eventually found ways round code triggering and made changes to plans for later increments.

Text

Description automatically generatedFigure 25. Project Log 12/04/23

React also posed me issues at the end of the project when I was deploying the webapp on Netlify, a free hosting service. When using the webapp the links wouldn’t work. It took a long time troubleshooting to eventually find a Netlify (2023) doc which describes how the indexing will not work with React and that an extra redirect file is required in the project files. An advantage from this process is I proved I’ve developed excellent strategies for dealing with difficult troubleshooting. These have developed throughout the project and I wouldn’t have solved this problem without them. Figure 26 shows my project log.

A black and white text on a black background

Description automatically generatedFigure 26. Project Log 20/08/23

To be honest, I don’t think I could have developed a full understanding of React before I started, as I was learning new concepts over different disciplines of web development. I am glad I have developed my knowledge of React during the project, as it will stand me in good stead when tackling my next project. I feel it was worthwhile sticking with React as its design advantages have helped me developed a good responsive webapp. The webapp could have been a bit better if I had designed the hooks into the initial plan, rather than adapting as I went but overall, the projects functionality is good.

After feedback from my tutor in an earlier TMA, I had been trying not to spend excessive time before moving on to another task. In the early stages of the project, I tended to get a section of code working, delivering the functional requirement but then I would spend several hours fiddling with HTML and CSS to get everything looking polished, despite it being early in development.

Since then, I had been more disciplined to continue with the schedule and return to incomplete or imperfect code later. An example is in FR7 where I spend a lot of time working on half the functionality and ran out of scheduled time. As below.

A black and white text on a black background

Description automatically generated with low confidence

Figure 27. Project Log 02/06/23

I was able to return to this code and finish it when another piece of functionality was completed much quicker than expected. Overall, I am please I was able to take the advice of my tutor and implement this improvement into my project management skills. It supported me to complete all the functional requirements agreed with the client and gave me time to work on some non-functional requirement I didn’t expect to tackle before the deadline (Security and Reactive design).

A skill I’m pleased I’ve developed is using strategies to adapt designs and plans when unexpected issues arise. This occurred when I was looking to send the order details to the client in an email alert. When planning I had assumed that JavaScript would have a built-in method to send a form via email. After some research I found an email service or server was required. The journal entry for the 15/04 (Figure 28) shows I researched the problem further and eventually found a third-party service which would bridge the gap and send the order details to the client. I was worried it would take a long time to learn and implement the service and considered shelving the functionality. After some research I was able to follow an EmailJS tutorial to set up the service and template in one evening. I was able to send an alert email to the client the next day.

I believe the continuing development of my research skills during the project gave me the confidence to attempt setting up this complex service and I was rewarded with an elegant solution to the client’s requirement rather than shelving or delaying is implementation.

Text

Description automatically generated

Figure 28. Project Log 15/04/23

The process of developing and implementing this project has given me new skills which are beyond what I anticipated I would be capable of at the start of the project. The feedback from my tutor and the completion of TM354, have given me better insight into how the project should be planned and produced, compared with my understanding at the start.

**Review of Project Management**

During TMA01 I was asked to research and decide on a life cycle model to use in the project management of the webapp (TMA01 Section in Appendix J). As descried in the report I opted for a hybrid of Scrum and Incremental life cycles. I believe this approach has worked well for me. The project has kept to schedule and the method allowed me to plan easier.

An aspect that has worked well is the short deadlines of each sprint. This helped indicate when it was time to move onto the next functional requirement and ensured that no requirements were missed before moving to the next increment. The shorter deadlines also make it easier to look ahead, to see where I could borrow time from to catch up on missing work. This resulted in a feeling of control over the project timeline. By the end of the project, I was able to keep to schedule (Appendix E) and added some extra functionality that was identified during the project, which I thought would be added after the deadline.

Another plus of this approach is the workload in each sprint seem to be the right size. The result being I never felt the tasks Infront of me were too daunting. There always seems to be a goal or check point in sight rather than fighting through all the work that must be done for the entire increment.

Another benefit is the way the requirements have been structured. I have found little dependences between each of the sprints. I seem to have planned the work in a way that doesn’t jump between functional requirements often to complete them. There has also been a good amount of code reuse, where I have been able to take similar code and either call it as a component elsewhere or copy the code and reuse most of it.

Overall, I would say this approach has really worked well for me. It broke the project down into manageable workloads with sensible deadlines. It also allowed me to have a lot of contact and feedback with the client (communication log in Appendix F). A real indicator of its successes is that I have kept to schedule and added a few extra functions (security and reactive design) and extra documentation for the client like the admin section user manual (Appendix I).

Project Resources

To complete this project, I had to consider a lot of different resources to make the project function effectively. The core resource of the database has been discussed in the related literature section.

I had to ensure I could easily develop and manage my code. I used Visual Studio Code writer and Google Browser, as VSC had an extension which allowed real time updates to the google browser when developing. This was a very helpful feature. These are both industry standards so little risk associated to using them.

I had to source and use a few third-party services to help develop the project. Each was a free service, as per the budget restrictions. There was little risk to using these are they are free and have good reputations.

* Supabase for databases and bucket storage.
* EmailJS for an email client to send automatic email updates to client
* Netlfiy hosting. Versions of the webapp have been added so it can be tested.

Personally, a resource which really helped me was investing in a second monitor for my desktop computer. This has made working on the project much more efficient, as I could open more windows and have an efficient workflow.

An excellent resource throughout the report was my tutor. He encouraged me to take on this workload and not to dilute it at the beginning. He has also given me excellent feedback after TMA’s and when emailing questions or asked for guidance. Communication with my tutor can be seen in the communication log (Appendix F)

The full Resources Table can be found in Appendix K.

Project Risk Assessment

There were many risks posed throughout this project. Some were mitigated as the project progressed, but some remained present right to the end of the project. Below is the projects risk assessment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk To project | Consequences | Likelihood | Impact | Risk Response |
| Finding an appropriate database in good time that will meet the requirements of the project. | Delaying the project and if incorrect database is implemented it could mean a redesign to meet requirements. | very low | High | I have a good understanding of the options, but research will be required to pick the best. Confident this will not be an issue. |
| Learning React Libraries to enhance the complexity of the website and make for a better overall project.  Zero prior knowledge of React. | If I am unable to implement React properly then I would have wasted a lot of time and it would probably mean losing functionality for the website.  Having no prior knowledge, learning my slow down the project. | Very Low | Very High | I will complete an early dry run of a react project to ensure I understand the principals of how it works.  As the language is new, I will need time to understand new parts. I will add 2 hours a week to any React code tasks. |
| Very High | Low |
| Learning how APIs work and how to implement them in this project. | If I don’t learn, then the database and website will not communicate, and all data will need to be stored locally which is not idea. | Very Low | Medium | After early research some databases have API’s built in. Using one of these options will negate a lot of this risk. |
| Sole Client is unavailable to discuss development or to provide required data. | Can delay development of project and could lead to requirements being implemented incorrectly. | Medium | Medium | To negate this, I have scheduled bi-weekly progress meetings with client and agreed deadlines for initial and full data collections. |
| Illness or other unforeseen delay | Project could be delayed, and some functionality not implemented as deadline approaches | High | Medium | Float time has been added to the schedule to cover such unexpected delays. |
| Users unavailable for acceptance testing | Project could be deployed without rigorous testing by users. Bugs could be present leading to late or missed orders | medium | High | To negate this, I am trying to black box test every component of code as I go. If I can’t get the client users to test, I will enlist family members to test the site. |
| Concurrently working on the software engineering OU course with its own deadlines and assessments.  Using concepts from software engineering in project despite the course not being complete. | Can be a drain on time and resources. Struggling on a TMA for that course can have negative impact on the project.  Could have miss understood concept and will not know until assessed therefore delays if implemented incorrectly. | High | Medium  Medium | Gantt chart has taken this into account and extra float time has been attached. Other TMAs will be completed at earliest opportunity.  Any key software engineering concepts to be used will be discussed with other tutor. |
| High |

Table 9. Risk Assessment

One risk that remained throughout was time management. I have always planned extra float time to mitigate this. I have given myself a deadline for this report of 4 days before the OU deadline to ensure I have handed in my best possible work without a frantic sprint to the deadline.

A risk which has been difficult to mitigate affected acceptance testing. The lack of test users to try the website posed a problem. Despite having ten users agree, only 4 were able to give me feedback before writing this report. Although I allowed plenty time, it was a risk that I knew could happen. Using this risk assessment is why I asked 15 users (10 saying they could) rather than a handful. Even so, the response was lower than I anticipated but the feedback I received was very helpful.

Overall, Managing the project risks has gone very well. The strategies to mitigate these risks have been effective and none of the high risk activates have had a large negative effect on the project.

**Review of Personal Development**

Throughout this project I have constantly been developing my web development and project management skills. Below is an account of skills I previously had, skills I have learned and skills I still need to develop.

*Relevant Skills Known Before Project*

Basic knowledge of web development – Before this project I had a basic knowledge of HTML, CSS and JavaScript. I also knew about basic structures like forms. This has been an excellent base to learn from, but I had to vastly develop my web development knowledge and skills.

Basics of database development – I have studied databases but mainly as a sorting and analysis tool rather than in a web development context. This still gave me a knowledge of the database types and their basic pros and cons.

Project management/Software design – I have done past courses at the OU on both subjects, and I have some good knowledge, but no practical experience of either.

*Skills Developed During the Project*

Using API’s – I have developed an understanding of how API’s work, allowing communication between databases and websites. I have successfully implemented “read from” and “write to” a database from the web application. I can also use APIs to update databases and to delete items. I have developed a basic understanding of database security policies.

React Libraries – I have learned how to use React Libraires to make an efficient and dynamic web application. I have used an array of different functionality that React offers, for example useStates, hooks and Links.

Using Email Server – I have always struggled with networking and servers. I was able to research how to use and set up a third-party email server. This research has developed my understand networks and why they are set up in certain ways.

Advanced Web Development – I have been developing my skills within HTML, CSS and JavaScript while adding React to make dynamic website. I have also been learning skills such was hiding elements and element overlays.

Using Data Storage Services - I had never used a service like Supabase or AWS despite reading about them in past courses. I now know how they work, how they interact with one another and how they can interact with a web application.

Using Client-Side Storage – Cookies were something I was vaguely aware off but now I have developed a working understanding of server-based storage and how it can be used.

Reactive Web Pages – I have been able to use advanced CSS to make a reactive page while using usability guidelines to ensure the page is as usable as possible.

Security – I was able to research and implement a login page which is secure. I was unable to do a full node.js version as there was not enough time but I used the fundamentals of that process to produce a secure login system.

Project Management and Software Development – I have used prior knowledge from past OU courses in a practical situation. This has developed my understanding of the processes involved in planning and developing a project and specifically a piece of software.

Report Writing – I feel this project has given me a more in-depth understanding of report writing and research. The EMA has provided a platform to push my skills and deliver a complete project in a report for the first time.

*Skills still to develop*

React – I would like to become more fluent in using react so I will continue to work through external courses on the subject.

Security – I would like to develop a proper Node.js back end to have a more secure login.

I believe the skills and knowledge I have developed during the project are well beyond my initial levels. I am very proud to have developed the skills to complete all the functional requirements outlined. I had hoped I could do it, but I was prepared to reduce the complexity if needed. After this project is submitted, I will continue to supplement this learning. I will try to extend my knowledge in widely used language like JavaScript, Python and SQL. I can see myself using the skills I’ve learned in this project when finding a job in web development.

Final Thoughts

I have really enjoyed all aspects of this project and I am very proud of the product that I have provided for my client. I believe the process of developing, researching, implementing and report writing has resulted in advanced skills and knowledge which will positively contribute to my post academic career.

Word Count - 12458

**References**

AWS (2023) *Methods for accessing a bucket.* Available at:<https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-bucket-intro.html> (Accessed: 20 March 2023)

Brophy, B. and Craven, J. (2007) ‘Web accessibility’: *Library Trends,* 55(4), pp. 950-972 Available at <https://muse.jhu.edu/article/216646> (Accessed: 3 April 2023)

Harris, G. (2021) *UK law and digital accessibility: What it means for your organisation.* Available at: <https://business.scope.org.uk/article/uk-law-and-digital-accessibility-what-it-means-for-your-organisation> (Accessed: 12 April 2023)

Huber, W. and Vitouch, P. (2008) ‘Usability and Accessibility on the internet: Effects of accessible web design on Usability’: Computers Helping People with Special Needs, pp. 482 -489. Available at: <https://link-springer-com.libezproxy.open.ac.uk/chapter/10.1007/978-3-540-70540-6_69> (Accessed: 28 March 2023)

Mulligan, D. and Schwartz, A. (2000) ‘Your place or mine?: privacy concerns and solutions for server and client-side storage of personal information’: Proceedings of the tenth conference on Computers, freedom and privacy. Pp 81 – 84. Available at: <https://dl.acm.org/doi/abs/10.1145/332186.332255> (Accessed: 12 April 2023)

Netlify (2023) *React on Netlify.* Available at:<https://docs.netlify.com/integrations/frameworks/react/#netlify-integration> (Accessed: 10 August 2023)

Nielsen, J and Tahir, M (2002) *Homepage Usability 50 websites deconstructed*. USA: New Riders Publishing.

React (2023) *React User Documents*. Available at: <https://react.dev> (Accessed: 20 May 2023)

Schmedtmann, J. (2022) *Crash Course: Build a Full-Stack Web App in a Weekend.* Available at:<https://www.udemy.com/course/full-stack-crash-course/learn/lecture/35198582#overview> (Accessed: 10 March 2023)

Schmutz, S. and Sonderegger, A. and Sauer, J. (2018) ‘Effects of accessible website design on nondisabled users: age and device as moderating factors’: Ergonomics, 61(5), pp.697-709 Available at: <https://www-tandfonline-com.libezproxy.open.ac.uk/doi/full/10.1080/00140139.2017.1405080> (Accessed: 16 April 2023).

Supabase (2023) *Open-Source SQL Database.* Available at: <https://supabase.com/database>

(Accessed: 20 March 2023)

The Chartered Institute For IT (2023) *BCS code of conduct.* Available at: <https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/> (Accessed: 21 June 2023)

Toiba, M. (2023) *NoSQL vs Relational: Which database should you use for your app?* Available at:<https://devblogs.microsoft.com/cosmosdb/nosql-vs-relational-which-database-should-you-use-for-your-app/> (Accessed: 4 April 2023)

UK Government (2018) *The Data Protection Act* Available at: <https://www.gov.uk/data-protection> (Accessed 5 May 2023)

Valentine, T. (2021) *Database-Driven Web Development: Learn to Operate at a Professional Level with PERL and MySQL* Berkeley, CA: Apress.

W3C (2023) *Accessibility Principles.* Available at: <https://www.w3.org/WAI/fundamentals/accessibility-principles/> (Accessed: 2 April 2023)

Appendix A – Initial Requirements

Incrementation 1

1. Client would like an online catalogue of all fabric designs and types of garments sold
2. All current fabrics pictured with Name, description and later (It 3) stock indicator.
3. All Garments will have description, picture, size/cost chart and customization options.
4. Garments of a chosen fabric can be added to an order form. Options about size and colour options will be captured.
5. Order form will show all Garments and fabrics chosen by the customer. An estimated total cost (with postage) will be displayed.
6. Order form will capture the customer’s name, email and address.
7. When an Order form is submitted it will be emailed to alert client and will be added to the Administration portal for client to manage.
8. Administration portal will be accessible to client via username and password. This portal should be hidden from public.
9. Order management page will list current orders by order number.
10. Orders can be marked as paid and they can be marked as complete.
11. Client can add or edit orders. This allows for changes or for non-website orders to be managed (i.e. through Instagram messaging).

Incrementation 2

1. Client would like a stock page to manage the stock of the fabrics.
2. Stock page shows current fabrics with total available length in meters.
3. When an order is paid for then the fabric required is reserved on the system and shown on the stock page.
4. The level of stock will be shown on the fabric catalogue page to give users an indication of time scale and availability (will the fabric need printed).
5. The system should be able to add new fabrics to the stock which will show on the catalogue page. This will also show the length in the stock page.
6. Fabrics in the stock page can be edited.

Incrementation 3

1. There should be a page in the catalogue showing premade items which are available immediately.
2. There will be a premade stock page in the admin portal.
3. New premade items can be added to the stock page, and these will show in the catalogue.
4. Premade items in the catalogue can be added to an order form.
5. When a Premade item is added to a submitted order it is hidden from the catalogue
6. When the item is paid it will be removed from the catalogue.

Appendix B – Use Cases For Functional Requirements

Table 1 Text Description of “Catalogue” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC1 Display catalogue |
| Initiator | Customer or Client |
| Goal | All fabrics are displayed with all their details from the Database. |
| Preconditions | None |
| Postconditions | All the fabric items stored in the database will be displayed. The display will adhere to Usability guidelines |
| Assumptions | There will be fabric Items, no reason for there not to be any. The client knows how to add new items. |

Main Success Scenario

1. The Fabrics Page of the web application has been accessed
2. The Application will use API to access all Fabric Items from the Stock Database
3. The system will display the fabric image with the correct name and description
4. The stock level will be displayed
5. If the stock is available, then an order button is displayed.

Extensions

* 1. If stock unavailable order button will be hidden.

Table 2 Text Description of “Order added to Basket” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC2 Order added to basket |
| Initiator | Customer |
| Goal | A selected and customized garment is added to the customers basket. |
| Preconditions | None |
| Postconditions | The selected item will correctly be added to the basket (with any previously added items) and stored in local storage. |
| Assumptions | The basket will be empty or have existing items that have been through this same process. There will be no invalid data as all data comes from drop down. |

Main Success Scenario

1. The Customer Clicks on the garment order button for the fabric they have chosen.
2. The System will show the item order form, it displays image and customisation options.
3. The customer will select customisation options from the drop-down menus and submit.
4. The garment item is saved to local storage.
5. Items added to the basket can be seen on the basket page.

Extensions

* 1. If Customer selects Hood garments, then the hood and tank top order form pops up.
  2. If the customer selects the Snood garment, then it Snood order form pops up.

Table 3 Text Description of “Order Submit” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC3 Order Review and Submit |
| Initiator | Customer |
| Goal | Items added to the basket can be reviewed and then an order can be submitted to the client. |
| Preconditions | Order will not submit if the basket is empty |
| Postconditions | The customer’s order is submitted, Order details are saved to the Order database and the Client is informed. The basket is emptied. |
| Assumptions | There will be an item in the basket before submitting. |

Main Success Scenario

1. The customer accesses the basket page of the application
2. The system displays all items stored in the basket from local storage
3. The customer will review the items, if happy they can submit order.
4. The customer can add their personal details in the text felids provided.
5. The customer then submits the order and the basket is empty
6. The order is stored in the Orders database.

Extensions

* 1. Customer can remove items from basket or empty the whole basket.

Table 4 Text Description of “Order Alert” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC4 Order Alert |
| Initiator | System |
| Goal | When an order is submitted an email with the orders details is sent to the clients business email. |
| Preconditions | An order has been successfully submitted by a customer |
| Postconditions | The client receives an email with the orders details. |
| Assumptions | The details sent to the client will have been inputted correctly. |

Main Success Scenario

1. The customer will submit an order as outlined in UC3
2. The system sends the details of the order to an email client.
3. The email client composes an email from the details sent.
4. The email is sent to the client, informing them of a new order

Table 5 Text Description of “Admin Orders table” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC5 Orders Admin Table |
| Initiator | User |
| Goal | The client can manage orders through tables in an admin portal. The orders will be automatically added. |
| Preconditions | There are Order items in the database |
| Postconditions | All current and completed orders can be seen on the Admin orders page. |
| Assumptions | The database Is up to data and running correctly. |

Main Success Scenario

1. The client accesses the order management page in the admin portal
2. The system will retrieve all Order items from the Orders database
3. The system will sort all order items into two tables, current or complete.
4. The Clint will be able to browse through this table using a scroll bar.

Table 6 Text Description of “Order Editing or Manual add” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC6 Adding manual order and editing existing orders |
| Initiator | Client |
| Goal | Client can add a new order in the portal. The client can edit all existing orders. |
| Preconditions | none |
| Postconditions | A new order is added to the database and therefore the order tables. Changes made to current orders are persisting. |
| Assumptions | none |

Main Success Scenario

1. The client accesses the order management page.
2. The client clicks on the row of the item they want to edit.
3. The Edit order form appears and is prepopulated with existing values.
4. The client makes changes from the drop downs or text boxes.
5. The form is submitted and any changes to the item are saved to the database.
6. The form is hidden
7. When refreshed the tables show the updated items

Extensions

2.1 The client clicks on new order.

2.2. The new order form appears and is not prepopulated.

2.3. Details are added from the drop down and text boxes.

2.4. The order is submitted, and the new order is added to the database.

2.5 The form is hidden

2.6 when refreshed the new order item is visible.

7.1 if items “complete” state changes then this will show in the relevant table

Table 7 Text Description of “Stock Management” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC7 Stock Management |
| Initiator | Client |
| Goal | The client can view, edit and add new stock items to the database which will be viewable in the Catalogue UC1. |
| Preconditions | There are fabric items in the database |
| Postconditions | All current stock is viewable, any new stock or changes to stock are seen once refreshed. |
| Assumptions | The client knows how to use the system. The photo used will be in the correct format. |

Main Success Scenario

1. The clint accesses the Admin Stock page.
2. The current stock is displayed in a table.
3. The client clicks to add a new stock item and the form appears.
4. The client fills in the relevant form and uploads the photo for the fabric.
5. The system sends the image to the storage bucket and retrieves the URL for the fabric.
6. The details of the new item (including URL) are sent to the stock Database
7. When refreshed the new item is in the stock table and is currently displayed on the fabrics page.

Extensions

3.1 The client clicks to edit an item and the edit item form appears

3.2 The current details are prepopulated

3.3 The client submits the changes

3.4 These are updated on the database and when refreshed are seen on the web app.

Table 8 Text Description of “Order Made” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC8 Order Made |
| Initiator | Client |
| Goal | To indicate that an order for an item has been made by the customer. The stock of selected item will be reserved. |
| Preconditions | The item has been added to the system |
| Postconditions | The items entry in the order management table will be updated to show that the item has been made. The amount of reserved stock of selected fabric will be increased by the amount required. |
| Assumptions | The order exists in the system. |

Main Success Scenario

1. The client will receive an order for item.
2. The client accesses the order management page from the admin portal
3. The client will click edit next to the item paid for.
4. The client will then select paid from the form and submit.
5. The system will update the paid status in the order management table.
6. The system will increase the reserved amount of that fabric.

Extensions

6.1 if the available stock falls below the thresholds the stock indicator in the catalogue (UC10) will change.

Table 9 Text Description of “Order Complete” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC9 Order Complete |
| Initiator | Client |
| Goal | To mark item as complete and remove the fabric from stock. |
| Preconditions | The item has been marked as paid for. |
| Postconditions | The items entry in the order management table will be moved from current to complete. The reserved stock and actual stock will be reduced by amount used. |
| Assumptions | The order exists in the system and has been marked as paid. |

Main Success Scenario

1. The client makes garment and sends to the customer.
2. The client accesses the order management page from the admin portal
3. The client will click edit next to the finished item.
4. The client will then select compete from the form and submit.
5. The system will update the item will move to the order complete table.
6. The system will decrease the reserved and actual stock amount of that fabric.

Extensions

6.1 if the available stock falls below the thresholds the stock indicator in the catalogue (UC10) will change.

Table 10 Text Description of “Stock Indicator” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC10 Stock Indicator |
| Initiator | Customer |
| Goal | Level of stock will be indicated and if there is adequate stock the order buttons will be displayed |
| Preconditions | The stock levels are up to date. |
| Postconditions | The customers will be able to see what stock levels are for each fabric and if levels allow, they can use the order button to make an order (UC2) |
| Assumptions | The stock levels are 0 or above |

Main Success Scenario

1. The customer will access the catalogue (UC1) from the Fabrics page.
2. The system will check the available stock level (Stock – Reserved).
3. The system will display the indicator of stock: In Stock, Low Stock or Out of Stock.
4. If there is enough stock the button for the garment will be shown.

Extensions

* 1. If stock is too low, then the order button is hidden.

Table 11 Text Description of “Pre-Made Items” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC11 Pre-Made Catalogue |
| Initiator | Customer |
| Goal | The customer will be able to see all premade items available for sale. These will show all details and can be added to an order. |
| Preconditions | The pre-made item has not be reserved. |
| Postconditions | A customer can browse all premade items and add to an order. |
| Assumptions | The order is nor reserved. |

Main Success Scenario

1. The customer accesses the Pre-Made items section in the products page
2. The system will retrieve all details from the pre-made items database and display items.
3. The customer can add premade item to their order. This is then added to the basket.

Table 12 Text Description of “Premade Management” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC12 Premade management |
| Initiator | Client |
| Goal | Allows the client to see all premade items and add new items to the website. |
| Preconditions | none |
| Postconditions | All premade items can be retrieved from the database. A new premade item can be added to the system |
| Assumptions | Photo for new item is in correct format. |

Main Success Scenario

1. The clint accesses the Admin Premade page.
2. The current stock is displayed in a table.
3. The client clicks to add a new stock item and the form appears.
4. The client fills in the relevant form and uploads the photo for the item.
5. The system sends the image to the storage bucket and retrieves the URL for the fabric.
6. The details of the new item (including URL) are sent to the Premade Database
7. When refreshed the new item is in the premade table and is currently displayed on the products page.

Table 13 Text Description of “Displaying Ordered Premade Stock” Use Case

|  |  |
| --- | --- |
| Identifier and Name | UC13 Displaying Ordered Premade Stock |
| Initiator | Customer and System |
| Goal | When a premade item is ordered it is hidden from the catalogue. An entry is made to the orders database. When the item is paid for it is removed from the catalogue completely. |
| Preconditions | Premade item is available |
| Postconditions | When an item is ordered it is unavailable to other customers. When paid for it is completely removed from sale. |
| Assumptions | none |

Main Success Scenario

1. The customer orders a premade item (UC11)
2. The premade item hidden from the catalogue
3. Once marked as paid the item is removed from catalogue

Diagram

Description automatically generatedAppendix C – WebApp Overview

Graphical user interface

Description automatically generatedAppendix D – Wireframes of Webapp Design

A screenshot of a project

Description automatically generated with low confidenceAppendix E – Gantt Chart For Project Schedule

Appendix F – Communication Log

Communication Logs

This log has been made up of interactions with other parties during the development of this project. I have made contact with two main outside soruces in helping me develop the project:

* The Client
* My Tutor

The document is a collection of Poject Logs, WhatsApp conversations, tutorials, video chats and emails.

Communicaiton with Client:

First Contact –02/02/23– Asking if the client would be interested in my making this project from them. Conversation was positive. Initial discussions of what Client would want the page to do.

05/02/23 – 2 hour video call to get initial ideas on paper and rough drawings of what the page may look like.

21/02/23 – Second long video call, initial functionality described. Rough overview diagram produced (Appendix C).

23/02/23 – Refined Overview document sent to client over Whatsapp and approved.

13/03/23 – discussed and refined requirements more.

29/03/23 – pictures of first implementation attempts sent to client. Happy with layouts. Video of refinements sent and aproved the next day.

01/04/23 – Test fabric pictures sent from the client so database can be populated.

15/04/23 – Video of functionality sent for aceptance testing from the client.

02/05/23 – video call to discus fuctionality so far. Some small changes requrired. Sizing guides and price guide given to me by clinet.

04/05/23 – Video sent to check layout of the home, fabric and product pages. Funcitonality already oked for these.

14/06/23 – Video update for order form functionality. Aceppted by the cilent.

20/06/23 – Two videos added with updates to the functionality. Messages from client asked for small changes which have been oked through screen shots.

28/06/23 – Checking Order Managmenet functionality with Videos and whatsapp messages to folow up.

14/07/23 – Video call to discuss Stock and premade stock functionality

18/07/23 – Follow up video form video call with updates discussed. Aceppted by the client.

27/07/23 – Video update of the first usablity edits.

17/08/23 – Video and whatsapp feedback to sign off on the Premade admin and catalogue.

20/08/23 – Page is now being hosted and Client has managed to get some people to test the site.

01/09/23 – Feedback from testers given back. Also client has completed the handover tasks I had ask they to do and provided me with feedback.

Communication with Tutor:

First Contact - 30/01/23 – Introductary email to give over view of my first idea for the project and to introduced myself.

03/02/23 – Email back and forth over a few days to tie down the what the project should be from my first 3 ideas and to get a better overview.

15/02/23 – Introductary Tutroial where project overview was given and questions about the process asked.

15/03/23 – Feedback recived for TMA01

19/03/23 – Email to update and get some clarification of going forward with the project.

18/04/23 – Email chain making sure that I had the right end of the stick before the submition of TMA02.

08/05/23 – Feedback recived for TMA02

14/05/23 – Couple of emails discussing how I can improve scores going forward in later assesments and giving an upate on progress to my tutor

19/06/23 – Email chain with update of the project progress. No major issues raised.

18/07/23 – feedback for TMA03

08/08/23 – Tutorial to discuss approch to the final report.

Appendix G – Selected Code

Below is the main structure of the APP file. Each calls another page which in turn call components

<>

<Title />

<Routes>

<Route path="/" element={<Layout />}>

<Route index element={<Home />} />

<Route path="products" element={<Products />} />

<Route

path="fabrics"

element={

<Fabrics

fabrics={fabrics}

setFabrics={setFabrics}

showFormHood={showFormHood}

setShowFormHood={setShowFormHood}

showFormSnood={showFormSnood}

setShowFormSnood={setShowFormSnood}

setBasket={setBasket}

basket={basket}

/>

}

/>

<Route path="preMade" element={<Premade preMades={preMades} />} />

<Route

path="basket"

element={

<Basket

basket={basket}

setBasket={setBasket}

fabrics={fabrics}

setBasket={setBasket}

/>

}

/>

<Route

path="adminLogin"

element={<AdminLogin setToken={setToken} />}

/>

<Route

path="adminHome"

element={<AdminHome token={token} setToken={setToken} />}

/>

<Route path="adminOrder" element={<AdminOrders token={token} />} />

<Route path="adminStock" element={<AdminStock token={token} />} />

<Route path="adminPreMade" element={<AdminPreMade token={token} />} />

<Route path="\*" element={<NoPage />} />

</Route>

</Routes>

<Footer />

</>

Code snippet 1

This code snippet shows the API sending data to the supabase Database and also show data being sent to the email server to send the email.

if (orderName && postcode && email) {

basketSend.map((*basket2*) =>

basket2.map(async (*basketSend*) => {

const { data, error } = await supabase.from("Orders").insert([

{

Order\_Number: orderID,

fabric: basketSend.fabric,

size: basketSend.size,

type: basketSend.type,

fleece\_colour: basketSend.colour,

postcode: postcode,

email: email,

orderName: orderName,

},

]);

})

);

function sendEmail(*e*) {

e.preventDefault();

var message = "";

basketSend.map((*basket3*) =>

basket3.map((*basketSend2*) => {

message += `Garment: ${basketSend2.type} .. Size: ${basketSend2.size} .. Fabric: ${basketSend2.fabric} .. Fleece\_colour: ${basketSend2.colour} / / / / / / / / / / / / `;

})

);

const values = {

order\_name: orderName,

message: message,

postcode: postcode,

email: email,

};

console.log(values);

emailjs

.send("service\_gl77v41", "basic", values, "Z\_xWZGcqk8z8-agm2")

.then(

(*result*) => {

window.location.reload();

console.log("email sent");

},

(*error*) => {

console.log(error.text);

}

Code snippet 2

Appendix H – Collated Testing

*For each Functional Requirement I have tested as the iterations have progressed. I have collated all the testing into this one document for convince. Each Functional requirement has an accompanying Use Case which can be found in Appendix D*

*I have used a mix of Fit Case testing, Black Box and Client Acceptance testing for each functional requirement.*

**Increment 1 – Catalogue and Order System**

***Testing Functional Requirement 1.***

FR1 - The application should display all garments and fabrics in a browsable catalogue. It should show all relevant details (pictures, name, description etc).

Description:

UC1 Step 3 - The system will display the fabric image with the correct name and description

UC1 Step 5 -If the stock is available, then an order button is displayed.

Fit Case:

The Fabrics page will display the picture, name, description, and stock level which match the items entry in the Stock Database. All items will be visible.

When stock is available the stock buttons will show.

Further Testing

It is expected that the order buttons will show if there is stock for the garment. Hoods need available 2m in stock. A snood needs 0.5 available stock. Available is stock minus the reserved.

Table 1 Black box Testing for Button Showing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fabric | Stock | Reserved | Test Reason | Expected | Actual |
| Unicorn | 0 | 0 | Normal both Out of stock | No Order Buttons | No Order Buttons |
| Tiger Green | 20 | 10 | Normal both In stock | Both Buttons | Both Buttons |
| Tractor Blue | 2 | 0 | Border both in stock | Both Buttons | Both Buttons |
| Space Navy | 1 | 0 | Normal Snood in Stock, Hood out of stock | Snood Button showing, Hood button hidden | Snood Button showing, Hood button hidden |
| Tractor Red | 2.5 | 2 | Border snood in stock and Reserved amount making stock low | Snood Button showing, Hood button hidden | Snood Button showing, Hood button hidden |
| Rainbow Navy | 5 | 5 | Normal all stock reserved | No order Buttons | No order Buttons |

Client Acceptance and Feedback

Client very happy with this functionality. Would like the buttons to be a little bit bigger and different colours. This will be addressed in increment 3.

***Testing Functional Requirement 2.***

FR2 - A selected garment and fabric can be added to a basket via an order form which allows customers to customise their garment.

Description:

UC2 Step 2 - The System will show the item order form, It displays image and customisation options.

Fit Case:

When the order form button is pressed the order form for the correct garment will appear.

Each Button tested to make sure the right order form appeared. This worked as expected.

Description:

UC2 Step 4 - The garment item is saved to local storage.

UC2 Step 5 - Items added to the basket can be seen on the basket page

Fit Case:

When the basket page is accessed, the items details match those in local storage. These items match those selected by the customer from the order form.

Further Testing

All options from the order forms are drop downs so there should be no invalid data, only normal data. Test to ensure local storage and Basket contain the correct data.

Table 2 Black box Testing for Basket/local storage after order form submit

|  |  |  |  |
| --- | --- | --- | --- |
| Entered Customisation | Test Reason | Expected | Actual |
| Snood, Teen, Red | Normal Snood Item selected from Rainbow Navy snood button | Both Show: Garment: Snood, Fabric: Rainbow Navy, Size: Teen, Fleece: Red, Top: None, Bottom: none, Cuffs : none. | Both Show: Garment: Snood, Fabric: Rainbow Navy, Size: Teen, Fleece: Red, Top: None, Bottom: none, Cuffs : none. |
| Hood, 3-4 years, red, blue, yellow | Normal Hood item selected from Tiger Green hood button | Both Show: Garment: Hood, Fabric: Tiger Green, Size: 3-4 years, Fleece: none, Top: red, Bottom: blue, Cuffs : yellow. | Both Show: Garment: Hood, Fabric: Tiger Green, Size: 3-4 years, Fleece: none, Top: red, Bottom: blue, Cuffs : yellow. |
| Tank, 6-7 years, Black, navy, purple | Normal Tank item selected from Dino Gray hood Button | Both Show: Garment: Tank, Fabric: Dino Gray, Size: 6-7 years, Fleece: none, Top: Black, Bottom: navy, Cuffs : purple. | Both Show: Garment: Tank, Fabric: Dino Gray, Size: 6-7 years, Fleece: none, Top: Black, Bottom: navy, Cuffs : purple. |

Client Acceptance and Feedback

Client very pleased with the basket functionality and the way the buttons/order forms work. Would like the order form to be bigger. Also wanted the Fabric name to be present on form (this has been implemented). Client would like the basket page to be set out as a table rather than a list. This will be implemented in the usability portion of Iteration 3.

***Testing Functional Requirement 3.***

FR3 - The basket can be reviewed and then submitted to the client as an order with the customers details.

Description:

UC3 Step 2 - The system displays all items stored in the basket from local storage.

Fit Case:

Will display the Basket page with each of the items added by the customer. Similar to testing for UC2 Step 5 but done multiple times.

Description:

UC3 Step 4 - The customer can add their personal details in the text felids provided.

Fit Case:

Customer adds personal details which are valid to the rules of the text fields.

Description:

UC3 Step 6- The order is stored in the Orders database.

Fit Case:

Oder details are successfully stored in the Order Database.

Further Testing

All options from the order forms are drop downs so there should be no invalid data, only normal data. Test to ensure local storage and Basket contain the correct data.

Table 3 Black box Testing for personal details and Order Submit

|  |  |  |  |
| --- | --- | --- | --- |
| Order Details | Test Reason | Expected | Actual |
| Item 1, 2, 3.  Name – Scott Tips  Email – [Test@test.com](mailto:Test@test.com)  Postcode – KY1 1SD | Normal Order send with valid personal details. | Successful Send,  Item 1,2,3 added to database with personal details.  Basket Emptied. | Successful Send,  Item 1,2,3 added to database with personal details.  Basket Emptied. |
| Name – Scott Tips  Email – [Test@test.com](mailto:Test@test.com)  Postcode – KY1 1SD | No items but correct personal details. | No Send, Basket remains | No Send, Basket remains |
| Item 1, 2, 3.  Name – Scott Tips  Email – [Test](mailto:Test@test.com)  Postcode – KY1 1SD | Normal order but invalid email | No Send, Warning about invalid email, Basket remains | No Send, Warning about invalid email, Basket remains |
| Item 1, 2, 3.  Name – Scott Tips  Email -  Postcode – | Normal Order but missing personal data | No Send, Warning about missing data, Basket Remains | No Send, Warning about missing data, Basket Remains |

Client Acceptance and Feedback

Client happy with this functionality. Nothing much happening on the interface but customer happy with warnings and what personal data take. Originally had a full address but client receives this during payment with PayPal and is relucent to hold unnecessary personal data. Postcode is not related to one person and will help identify orders.

***Testing Functional Requirement 4.***

FR4 - The client will receive an email to alert them of a new order. The email will have details of the order.

Description:

UC4 Step 4 - The email is sent to the client, informing them of a new order

Fit Case:

Client received email of each order to the business email account with all relevant details.

Further Testing

This is only sent when a successful order is sent to the database so the testing in Table 3 is relevant as only when the valid info is sent it should send the email.

The outcome of this testing is, yes, the email is only sent when a successful order is sent.

Client Acceptance and Feedback

Client happy with this functionality. Customer would like a better layout on the email sent from the client. This is something that I will need to work on as I have not found a way to do that yet.

**Increment 2 – Administration Portal**

***Testing Functional Requirement 5***

FR5 - Items from the order will be added to the current order list in the order administration portal

Description:

UC5 Step 2 - The system will retrieve all Order items from the Orders database

Fit Case:

All orders from the database will be visible on the Order Admin page.

Description:

UC5 Step 3 - The system will sort all order items into two tables, current or complete.

Fit Case:

All orders not marked as complete are in the current orders table, all completed orders in the completed table.

Further Testing

Testing to ensure that all orders are in the correct table. This is a case of checking the order ID’s that are complete and not complete and ensuring they are right. This has been implemented correctly.

Client Acceptance and Feedback

Originally there was to be only one list of orders with a complete marker on them. The client requested that the current orders and completed orders be kept apart. The client also asked for the tables to be ordered by order number to keep ordered grouped. The client is now very happy with the layout and functionality of viewing the made orders.

***Testing Functional Requirement 6***

FR6 - The client can add manual orders or edit current orders in the administration portal. The orders can also be marked as paid or completed.

Description:

UC6 Step 3 - The Edit order form appears and is prepopulated with existing values.

Fit Case:

The Edit form appears and has the correct prepopulated details of the selected value.

Description:

UC6 Step 5 -The form is submitted and any changes to the item are saved to the database.

Fit Case:

The changed details are updated in the database and visible after the page is refreshed.

Further testing

I had to ensure that any changed data was being recorded correctly and that invalid data was not added. Most of the options were drop down menus so could not be invalid.

Table 4 Black box Testing for Order Editing

|  |  |  |  |
| --- | --- | --- | --- |
| Updated Data submitted | Test Reason | Expected | Actual |
| Fabric – Hearts Blue  Top colour – Red  Name – Scott Pips  Email – [pips@test.com](mailto:pips@test.com) | Normal data to be updated for order with id 22. | Update Sent to database.  Database Row id 22 – Edits all made correctly.  Updates show on refresh | Update sent to database.  Database Row id 22 – Edits all made correctly.  Updates show on refresh |
| Email –pips.com  Postcode – KY1 1SD | Invalid email data. | No Send, System email warning | No Send, System email warning |
| No changes | No changes made but submit pressed | No send as no changes | No send as no changes |
| Name - | Data changes to be blank | No Send, System required input warning. | No Send, System required input warning. |

Description:

UC6 Step 2.2 - The new order form appears and is not prepopulated.

Fit Case:

A blank version of the edit form appears for client to populate.

Description:

UC6 Step 2.4 - The order is submitted, and the new order is added to the database.

Fit Case:

A new order row is added to the order management. Data is saved in the Orders Database.

Further testing

Table 5 Black box Testing for New order

|  |  |  |  |
| --- | --- | --- | --- |
| Updated Data submitted | Test Reason | Expected | Actual |
| All Valid data of new order | Normal data to be added to database. Random gen order number given | Send of data complete.  New order visible in database and order management | Send of data complete.  New order visible in database and order management |
| All valid data but one empty text cell. | Missing data | No Send, System required warning | No Send, System required warning |
| All vaild data but invalid email. | One instance of invalid data | No Send, System email warning | No Send, System email warning |

Client Acceptance and Feedback

Client happy with system. Originally did not have prepopulated data but changed after feedback. If possible, client would like the order number of the next new order to be pre populated with the last order manually submitted.

***Testing Functional Requirement 7***

FR7 - The client can manage stock from the stock administration portal, viewing all fabrics and level of stock. New fabrics can be added, and stock numbers can be updated.

Description:

UC7 Step 3 - The client clicks to add a new stock item and the form appears.

Fit Case:

The Add Stock form appears on the screen.

Description:

UC7 Step 5 - The system sends the image to the storage bucket and retrieves the URL for the fabric.

UC7 Step 6 - The details of the new item (including URL) are sent to the stock Database

Fit Case:

The uploaded image is then visible in the fabrics catalogue.

Further testing

To test this functionality, I needed to check that the new stock item appeared in the fabrics catalogue. It was found that description could be to long for the container on the user interface, but a maximum length restraint added to the input got round that issue.

Client Acceptance and Feedback

Client is pleased that they can add new stock without accessing the database or data stores. They had no feedback on this Functionality.

***Testing Functional Requirement 8 and 9***

FR8 - When an order is marked as paid the length of fabric needed becomes reserved.

FR9 - When an order is marked as complete the length of fabric is removed from stock.

Description:

UC8 Step 5 - The system will update the paid status in the order management table.

Fit Case:

The selected orders paid marker will now show paid.

Description:

UC8 Step 6 - The system will increase the reserved amount of that fabric

Fit Case:

The Stock database will update its reserved figure. This may change the stock indicator on the fabric catalogue page.

Description:

UC9 Step 5 -The system will update the item will move to the order complete table.

Fit Case:

The order row will move from the current to complete table.

Description:

UC9 Step 6 - The system will decrease the reserved and actual stock amount of that fabric.

Fit Case:

The Stock database will update its reserved and stock figures. This may change the stock indicator on the fabric catalogue page.

Further Testing

I needed to ensure that the paid marker was working correctly and that items were moving when completed.

Table 6 Black box Testing for Paid and Complete State changes

|  |  |  |  |
| --- | --- | --- | --- |
| Updated Data submitted | Test Reason | Expected | Actual |
| Paid from “No” to “Yes”  Complete Remains “No” | Normal data, Update of Paid | Paid is now Yes, Order details remain in Current order table. | Paid is now Yes, Order details remain in Current order table. |
| Paid from “No” to “Yes”  Complete from “no” “Yes” | Normal Data, update for paid and complete | Paid is now Yes, Order details now in complete order table. | Paid is now Yes, Order details now in complete order table. |
| No changes | No changes made to Paid or Complete but other changes made. | No change to Paid or which table the order is in. | No change to Paid or which table the order is in. |
| Paid remains “no” Complete from “no” to yes | Normal data, Garment complete but no payment yet | Paid remains No, Order details now in complete order table. | Paid remains No, Order details now in complete order table. |

Table 7 Black Box Testing for Reserved and Complete State Changes.

|  |  |  |  |
| --- | --- | --- | --- |
| Updated Data submitted | Test Reason | Expected | Actual |
| Fabric is ordered, Garment is a Snood. | Normal data for new order.  (Can’t access order form if no stock. ) | Fabrics Reserved Total rises by 0.5.  Available Total (Stock – Reserved) reduced by 0.5. | Fabrics Reserved Total rises by 0.5.  Available Total (Stock – Reserved) reduced by 0.5. |
| Fabric is ordered, Garment is a Hood or Tank. | Normal data for new order.  (Can’t access order form if no stock. ) | Fabrics Reserved Total rises by 2.  Available Total (Stock – Reserved) reduced by 2. | Fabrics Reserved Total rises by 2  Available Total (Stock – Reserved) reduced by 2 |
| Ordered Snood Item is marked as complete. | State change to complete | Fabrics Stock total and reserved total reduced by 0.5.  Available Stays same | Fabrics Stock total and reserved total reduced by 0.5.  Available Stays same |
| Ordered Hood or Tank Item is marked as complete. | State change to complete | Fabrics Stock total and reserved total reduced by 2.  Available Stays same | Fabrics Stock total and reserved total reduced by 2.  Available Stays same |

Client Acceptance and Feedback

Client accepts the functionality and is happy how it works. Feedback acted on was to change Paid to a yes or no rather than a tick box and for complete to be changed from tick box to a position in one of two tables.

***Testing Functional Requirement 10***

FR10 - Each fabric will show if it is in stock, low stock or out of stock in the catalogue.

Description:

UC10 Step 3 - The system will display the indicator of stock: In Stock, Low Stock or Out of Stock.

Fit Case:

The corresponding stock item will show its level of stock to the customer

Further Testing

Depending on the stock level there will be an indicator of the level of stock. An item shows low stock if its under 5 meters but more than 0.

Table 8 Black box Testing for Button Showing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fabric | Stock | Reserved | Test Reason | Expected | Actual |
| Unicorn | 0 | 0 | Normal out of stock | Out of Stock | Out of Stock |
| Tiger Green | 20 | 10 | Normal In stock | In Stock | In stock |
| Tractor Blue | 3 | 0 | Normal Low stock | Low Stock | Low Stock |
| Space Navy | 10 | 5 | Border line in stock and low stock | In Stock | In Stock |
| Tractor Red | 4.5 | 0 | Border line in stock and low stock | Low Stock | Low Stock |
| Rainbow Navy | 0.5 | 0 | Border line Low stock and out of stock | Low Stock | Low Stock |

Client Acceptance and Feedback

Client very happy with this functionality. Would like the stock number to be removed. Will be done after testing of parts of increment 3.

**Increment 3 – Premade and Usability**

***Testing Functional Requirement 11***

FR11 - The application will have a catalogue of premade items.

UC11 step 2 - The system will retrieve all details from the pre-made items database and display items.

Fit Case – The system will display all details stored in the Premade items database clearly in the catalogue.

Client Acceptance and Feedback

The client was unsure of the button being bellow the items so it was added to the end of each row of an item.

***Testing Functional Requirement12***

FR12 - The stock of premade items can be managed from the administration page. New items can be added.

Description:

UC12 step 6 & 7 - The details of the new item (including URL) are sent to the Premade Database. When refreshed the new item is in the premade table and is currently displayed on the products page.

Fit Case:

The new Premade Stock item is shown in the admin page and is also shown in the catalogue.

Client Acceptance and Feedback

The client is happy with the functionality. It operates very similar to earlier FR.

***Testing Functional Requirement 13***

FR13 - When a premade item is ordered it is hidden from the catalogue. Once payment is made it will be removed from the catalogue.

Further Testing

I needed to check that a premade item is hidden in all instances once it has been ordered.

Table 9 Black box Testing for Hidden Premade Item after Order

|  |  |  |  |
| --- | --- | --- | --- |
| Updated Data submitted | Test Reason | Expected | Actual |
| Premade Item added to basket and Order sent | Normal order process. | Premade Item should be hidden in the premade catalogue | Premade Item should be hidden in the Premade catalogue |
| Premade Item added to basket but order not complete | Order not sent to item should be visible | Item Still showing in premade catalogue | Item Still showing in Catalogue |
| Client edits item to Ordered Tag is yes | Manual order out with the Webapp. | Item will be hidden in premade catalogue | Item will be hidden in premade catalogue |
| Order of Premade item cancelled. | Premade item hidden but needs to be shown. | When cancelled Ordered Tag reverts to no and item shown | When cancelled Ordered Tag reverts to no and item shown |

Appendix I – User Manual for Administration Portal

**Snoods and Hoods Aministration Portal User Manual A logo of a green house with text

Description automatically generated**

**Contents**

Loging In and Navigation Page 2 - 3

Order Management Page 4 - 5

Stock Management Page 6-7

Premade Item Management Page 8

**Loging In**

*How do I login?*

The only way to acesse any of the portal is to log in. If you try to access pages of the portal without login in it will redriect to the home page.

A screenshot of a computer

Description automatically generated

The password and Admin ID will be suppied in a different document. If you require a ID or Password reset, contact the Developer.

You will need to login each time you visit the portal and you will need to login if the page is refresehed. There is a certain way to navigate the portal.

Navagation of Portal

*How do I navigate the portal without Logging out?*

The portal should be navigeted using the Button links on each page. On the portal home page you can accesss each management area or logout.

A screenshot of a website

Description automatically generated

Each of the management pages have a Admin Home button. This can be used to return to the home page and move to another management page.

A screen shot of a computer screen

Description automatically generated

If you were to use the browser back button or refresh the page then you will need to login again.

Order Management

The order management page is made up of two tables. One with open orders and the other with completed orders. When an order is made by a customer its added to current orders.

A screenshot of a computer

Description automatically generated*How do I edit or Mark as Complete?*

To change an order to complete or mark as paid you can use the “Edit” button at the end of the row. This will bring up a form with the details of the order. You can mark the order as complete or paid at the bottom. Any changes to the order can be made here too.

A screenshot of a computer

Description automatically generated

*How do I delete an order?*

At the bottom of the Edit for there is a button marked delete. Once pressed you will get an alert asking if you are sure. If deleted the order is wiped from the database so can’t be retreived.

A screenshot of a computer

Description automatically generated

*How do I add a manual order taken off the webapp?*

At the bottom of the compete table there is a button marked “New Order”. Pressing this will give you the New Order Form. You should add all the details of the order. Remember that if the order doesn’t have a colour in a category, select “None”.

Once all the detials are added, submit the form and the new item will be visible in the open orders table. The order will be given an order number automatically. If there it more than one item in the order, you should edit each subsiquent new items order number to match the first one.

Stock Management

A screenshot of a computer

Description automatically generatedStock management is made of a table with all items visible in the webapps catalouge of fabrics.

*How can I add a new farbic to stock?*

Pressing the “Add Stock” button on the top right will open the add stock form under the table.

A screenshot of a computer

Description automatically generated

Add all the new stocks details, the reserved amount will always start at 0. To add a photo, use the file slector to find the photo on your computer. Make sure the photo is a .png file. To make sure rename the picture the same as it is but add .png at the end.

Example: Tractor\_Blue.png.

New fabrics will be visible in the stock table and will show in the catalogue.

*How can I edit the Stock Numbers?*

A screenshot of a computer

Description automatically generatedTo edit stock numbers, press the “Edit” button of the row you want to change. A edit form will aprear at the bottom of the screen. This will show the current detials which can be chaged.

Once sumbited the changes will be seen in the stock table and the catalouge.

*How can I remove an stock item?*

Find the item you want to delete and press the “Edit” button. At the bottom of the edit for is a “Delete” button. Once pressed you will be asked to confirm deleting via an alert box.

A screenshot of a computer

Description automatically generated

Premade Item Management

A screenshot of a computer

Description automatically generatedThis page is very similar to the Stock management page. It is made up of a list of premade items. They have the fields which tell you if an item has been ordered which will change from No to Yes when a customer submits an order. The same item will be seen in the Order manangement page.

*How do I add a new premade item?*

Use the “Add Stock” button at the top of the table to open the add stock from. Then fill in the fields. The unique reference if for your own use and makes it easier to match to ordered items. The price will be in pounds but you don’t need to add the £.

The photo is the same as the stock photo. Make sure the file has .png at the end.

Once submitted the item will be in the premade items stock list and will show on the premade items catalogue on the website.

*Can I edit an item? How do I mark an order as sent?*

Yes, in the same way as before. Find the item you want to edit, press the “Edit” button and this will open the edit form. All the item detials will be there and you can change what you need to.

If someone orders the items manually, you can edit its “item ordered” status to “Yes” this will hide the item from the premade items catalouge.

Once you have sent the item you can again mark this in the edit form.

You can also remove an premade item in the same way as before. At the bottom of the edit form you can press “Delete” and then yes at the warning alert.

Appendix J – Life Cycle Research from TMA01

Selecting a Lifecycle model

There are several lifecycle models available and elements of some fit this project better than others. To narrow down the options, I created Table 1 below to compare the pros and cons of the main lifecycle models in relation to my project.

|  |  |  |
| --- | --- | --- |
| Lifecycle Model | Strengths | Weaknesses |
| Classic Waterfall | There are few stakeholders in my project so finding a common path should be simple.  The model should be easy to schedule as each section can be given a deadline and the project moves on to the next stage. | I don’t have a huge amount experience in the ICT area of this project and some concepts may need revisited or changed. Waterfall model is too ridged for this.  As this project has 3 distinct tasks, developing and implementing these at the same time could be very difficult to organise alone. |
| Iteration | Iterative allows for changes to be made following feedback. This means issues can be caught well before testing and changed. I plan to take advantage of client feedback at the end of each milestone, this approach allows for quicker adapting to errors and changes. | If I start to iterate several sub tasks at the same time, which is possible in a solo project, I could find myself in an endless loop of trying to get one small element correct before moving onto the next task. Again, working on a wide range of tasks at once could become unmanageable. |
| Incrementation | Incremental development lends itself well to this project as its broken into 3 clear tasks which each could be an increment. The dependences between the tasks are not strong and are therefore not a barrier, if some preplanning is implemented. I also have agreed the priority of the tasks with the client. | A purely incremental approach to complete the project would not be achievable in the time frame. Also, I think there might be better ways to approach the few dependencies that crop up, using an iteration of that stage rather than another incrementation of the task. |
| Agile Scrum | The Agile approach combines both increments and iteration based on feedback. I believe this is well suited to this project. As my client is the sole member of the business, I will receive very quick feedback on completed tasks. Any changes can be then iterated on before moving to the next task. Overall, this approach should result in a more personalised product for the client. Also, giving each sub task a “Sprint” style deadline will make scheduling easier. | The Agile approaches, particularly scrum, is focused on getting groups of developers to work in unison. As I will be completing this project alone, aspects like daily stand-ups and backlog boards sharing progress will be redundant.  If I follow the Scrum framework to the letter, I would be wasting valuable time.  As this is my first project, I feel having a slightly more ridged life cycle model will benefit my organisation of the project. |

Table 1. Strengths and Weaknesses of Life Cycle models

After analysing the main lifecycle models and for the reasons above, I don’t think one model will suit my project. I will approach the project using the core of the Agile Scrum model, without the redundant group work parts. I will achieve this by breaking the project into three main task increments. Each of these will break into sub-tasks with short deadlines, like a scrum sprint. I will consult the client regularly and use feedback to iterate over the task and make any changes required. This will require regular scheduled meetings with the client over zoom but this should be possible. I will use the tasks that make up a regular non-agile approach to keep structure to the project.

I am currently studying TM354 alongside this project. I would like to attempt the implementation of some Software engineering theory in the project. This would be in the form of requirements and class modelling. Although I have not studied the whole course, I would hope that future study on testing and implementation will be useful as the project progresses.

Appendix K - Project Resources Table

|  |  |  |
| --- | --- | --- |
| Resource | Why is it needed? | Possible Risk/issues Acquiring. |
| Visual Studio Code | Environment for coding the website. Used for Java Script, CSS and HTML. Has plugin for instant change update to browser. Also works with React Libraries. | Already familiar and using VS code on my device. A risk is an add on I need to cover an issue is not compatible but VS code is very up to data and well used. |
| Database | Required to store data and images about fabrics and garments. | Research needed to find best option for the project. Will need to be free use. \*Used database worked very well\* |
| Google Browser | Used to test the website. Requires inspector enabled to see developer options. | Simple to use. Goggle development tools use in OU course so already installed. No real risk as industry standard. |
| Client | Client is the sole member of the business so makes all decisions. Needed for feedback on completed work and testing as project develops. | Will use Zoom to talk, which I have download. With a sole person to talk to, if client is not available there is no one else to consult over decisions so prior planning and collaboration will be required. |
| User | Needed for testing | From the clients existing customer base. If unavailable, friends and family. |
| Stock list and Garment details. | Data needed to populate the databases and webpage. | Supplied by the client. Risk of missing data so we have agreed on a format and a deadline for initial data to aid development and a full list towards the project conclusion. |
| Email Client | To send information of an order to the client. | Research will be required to find a suitable way to send emails. |
| 2nd Monitor | To have a more efficient workstation while coding. More windows can be open at one time. | Bought and installed once I was finding flipping between windows very inefficient. |
| Email Address for business | The Client has asked for order details to be emailed. There is no business email, so I have made one. | Simply set up from Gmail with the company name. Client can change in future but needed for development. |

Table 1. Resource List