Men’s Shed Web Application Final Close-out Report

SOC09109 2022-3 TR2 001 - Group Project

Table of Contents

Introduction

Group Project

This is the final report for the Group Project. The Project chosen by this team was the Men's Shed Web Application. This report will focus on the overarching principles of managing a project, the dynamics of the team and the initial assessment, agreement, and delivery of the chosen project. The report will also discuss the methods used to deliver the project and there will be a stakeholder evaluation and analysis covering the project.

The Team

The Men's Shed group team is made up of 3 disciplines Computing Engineering, Software Developers and Security Specialists. The individuals who signed up to be part of the team and their roles are listed in Table 1 below:

	Name	Role
1	John Johnston	Project Manager (PM)
2	Jonathan Cloke	Security and testing (ST)
3	Rory Mackintosh	Principle Backend Developer (PBD)
4	Duncan Hastie	Secondary Backend Developer (SBD)
5	Joe Black	Principle UI Developer (PUD)
6	Daniel Beardmore	Secondary UI Developer (SUD)
7	Iain Donald	Sponsor

Table 1 Men's Shed Project Team and Roles

The Goal

The goal of this group project was to develop a piece of work for a live project proposed by a real client. The client being the Dunfermline Men's Shed Committee. Mr Ron Skirving (MS) is their spokesperson and the main contact for the project, all communications with the client are passed through the PM to the MS.

Project Description

Dunfermline Men's Shed objective.

The committee of the Dunfermline Men's Shed (MS) aspires to create a comprehensive and engaging resource for their community. The MS has access to some additional facilities and is keen to attract additional members and promote diverse activities. They already have selection of interest groups away from the typical MS Activities of woodwork or metalwork. As part of their aspirations, the intended application should provide a platform upon which they can build as the need arises.

The project

Following the initial meeting between the university Project Team and the MS representatives on the 2nd February 2023. The MS representatives highlighted their vision for the Dunfermline association, long-term objectives, and aspirations. The MS team want to build an exemplar of a modern adaptive association embracing the wider community and diverse interests. i.e., Computer club, guitar club and workshop facilities. But also, a lending library for the many tools, books, videos, and publications that the people of the community have donated at large.

They proposed the development of a web application as a tool for both members and the management of the organisation to track the donated equipment and manage their equipment, including the facility at large.

The PT highlighted to MS representatives this module's learning objectives and the available time available to produce a worthwhile and in-depth application proposal. The reality of the project is that the exercise is to be done over a trimester and that the PT members will have other learning and assignments to achieve during the same time allocation, so the time available is limited.

The negotiations cleared a way forward from the conversation, and the PT suggested as a team, the limited time would be better spent on creating a foundation piece of work that could be developed further by subsequent student project teams.

There were many different directions that the MS wished to develop the application; however, much of their aspirations were not feasible within the period allowed. Therefore, an agreement was confined to the following list.

1. Core application with:
    a. User interface
    b. Database
    c. Administrators User interface.
    d. Administrators access to maintain Data.

The MS agreed to provide the following items to assist with the development of the application:

1. Management personal information for the Administrator's access
2. A spreadsheet with tools and images to display as being available for use.

MoSCoW prioritisation (Proposed)

Label	Interpretation
M	• Constructing the backend and the database • Way method to restrict users from accessing dangerous tools that have or could have a severe impact on health and safety (Mental Health, Ability) • Admin check that items are still safe to use, • Health and safety requirements, PPE is needed for the item. • Admin functions • The ability to add/update/retire resources, User registration, • User Registration
S	• Hiding an item from view if currently booked out. • High amount of good documentation for people in the future to build on. • The static directory should be required to hold all images that will be used for the web app.
C	• Could have a feature to set up a delivery option for larger equipment for sharing amongst Men's shed locations
W	• The blog page application Men's Shed expressed interest in will not be involved in our project.

Table 2 MoSCoW Prioritisation (Proposed)

MoSCoW prioritisation (Achieved)

Label	Interpretation
M	• Constructing the backend and the database • Way method to restrict users from accessing dangerous tools that have or could have a severe impact on health and safety (Mental Health, Ability) • Admin check that items are still safe to use, • Health and safety requirements, PPE is needed for the item. • Admin functions • The ability to add/update/retire resources, User registration, • User Registration • Hiding an item from view if currently booked out. • High amount of good documentation for people in the future to build on. • The static directory should be required to hold all images that will be used for the web app. The only time this would not be required is if the web app did not have any images involved. •
S	• None
C	• Could have a feature to set up a delivery option for larger equipment for sharing amongst Men's shed locations
W	• The blog page application Men's Shed expressed interest in will not be involved in our project.

Table 3 MoSCoW Prioritisation (Achieved)

Deliverables Map

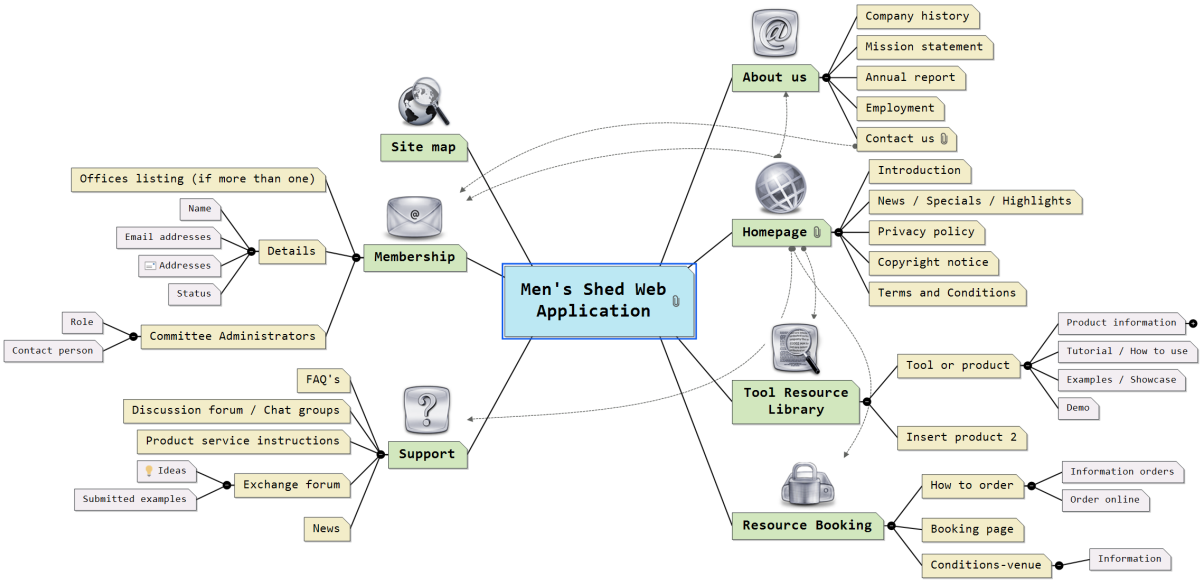

Figure 1 Deliverables Map

Project Management

The PT is happy to report that the project has been a valuable learning experience and that all members have actively participated in the development of the project. At the start of this learning experience, the team members were invited to share their preferences on what elements of the project (based on the initial documentation) they wished to contribute; 50% volunteered their roles, while the other 50% needed to be recruited. Where positions were appointed, those team members embraced their roles with enthusiasm. Each team member has been valued, and their opinions are sought at every stage of the project development.

With each role being clearly defined, i.e., Project manager, Principal Backend Developer, Principle UI Developer and finally, Security and Testing, The PM (Project Manager) used his experience to enable the team to take ownership of their area of work. They have been guided where necessary to collaborate with their counterparts for the benefit of the overall project. Each area of development has had the benefit of multidisciplinary team members from the BEng Computing, BEng (Hons) Cybersecurity and Forensics and BSc Software development courses. This worked well as team members collaborated and complimented each other's disciplines. The net result is a comprehensive cross-learning experience and collaboration on a live project.

All the project monitoring paperwork was collated, and where necessary, tables and spreadsheets joined to update each other, making recording information and time spent on the project simple and streamlined. At each scrum meeting, the members were encouraged to complete their contributions, maintaining the ownership of their individual contributions. However, the PM had to prompt to ensure these activities were completed from time to time.

In the absence of the PM, the ST, Mr Cloke, deputised and maintained the continuity of the team, hosting meetings and ensuring that the team met deadlines and completed all relevant paperwork.

The Project Manager

The project manager's role was taken by John Johnston, who by a considerable number of years, is the team's oldest member. John took the role in the absence of any other team member taking the initiative to bring a team together. Using his experience of team and collaborative working within the Military and Prison Service, John saw an opportunity to build an effective team that could deliver an acceptable product with endless possibilities for the Men's Shed organisation. While at the same time sharing his professional experiences with the team, empowering them to do well and work together for the benefit of both the team and the client.

While empowered to take ownership of their development areas, the team required some motivational and encouraging feedback from time to time. Still, most of the work completed was under their own initiative.

Principle Backend Developer

The Principal Backend Developer (PBD) of the Mens Shed web application, Rory Mackintosh (PBD), worked alongside (SBD) Duncan Hastie to design and develop the backend functionality. The overall experience of designing and developing the web application's backend has been a challenging yet rewarding experience that has allowed us to learn several new skills and techniques from a practical standpoint and within a group environment. The web application development followed the design plan previously created by the team. The main backend design element that has been implemented into our web application is the use of the Python Flask framework, which is a web development framework used alongside the Python programming language to allow for more efficient and sleeker web development by making use of built-in features such as the routes function for returning html templates and adding functionality to pages. The use of a packaged files structure can be noted in Appendix 5, and the use of an SQLite database implemented within the web application, which has been used to store given data provided by the Mens Shed team and the data created by users who use the website.

The Mens Shed team had requested some key features during the project's design stage. These features include admin functionality allowing admin users to have access to pages

regular users cannot, and a booking function where different branches of the Mens Shed organisation can request to book equipment such as tools for a certain period. The PBD, Rory Mackintosh, took it upon himself and SBD Duncan Hastie to decide what features to implement, given our limited time frame to complete the web application. The design and developed features implemented within the web app can be seen as follows:

Feature	**Description**
Database Creation & Population	The first feature implemented by the PBD and SBD was the creation of the SQLite database used to store and present Mens Shed data. The Duncan Hastie (SBD) created the initial SQL script to be used within the SQLite command line interface which was done by Rory Mackintosh (PBD) to create an empty database with the correct tables, columns and assigned types. Once created the next step was to populate the database with the given Mens Shed data. The data was transferred into a CSV file by Duncan Hastie (SBD) which was then used to be automatically implemented into the build database via SQLite command line interface by Rory Mackintosh (PBD). This was done with the tools data which was added to the Tools table within the database.
Signup Functionality	The next feature developed for the Mens Shed web application was a signup feature which allows users to create an account by taking details such as email, username, and password. These details are then stored in the SQLite database under the Users table which is later used for the login feature. The stored user data is also used during the signup function to check whether a new user is trying to input an already existing username or email. The python code for this functionality is stored under the signup route which returns the signup.html template

Login Functionality	The login feature takes users data from the Users table within the SQLite database to authenticate that a current user has an email and password to gain access to the Mens Shed web app. If they enter the incorrect email or password, a message will be flashed telling them that their details are wrong and to try again. If a user enters the correct details, then the current user will have access to more pages within the web app, such as the bookings page using Jinja2 within the HTML files. They will also get a flashing message telling them they have successfully logged in.
Tool Search Functionality	The Tool search function allows users to search for tools that the Mens Shed organisation currently owns. Using the given Tool data by the Mens Shed team, we used this data to populate the Tool table within the SQLite database, which is then used for the search function by using an SQLite query to search for tools that users enter. For example, if a user wants to search for a drill, they can enter the word drill and using the LIKE function within SQLite queries, it will return all results that contain the word drill in it, showing the user all the different types of drills currently in possession of the Mens Shed organisation.
Booking Functionality Prototype	The Booking function currently implemented within the Mens Shed web app it currently just a prototype to give the owners of the Dunfermline Mens Shed branch an example of how they could book equipment between branches. The Booking function allows users to enter their name, the item id of the item they want, the date they would like to book it, the start time they would like it for and the end time. These details are then stored in the SQLite database under the Bookings table. If a user tries to make a booking with an item already taken during

	that time slot, then the user will receive a warning message telling them that the item is invalid and to choose another time. Bookings are automatically deleted from the Bookings table once the date and end time of that booking has passed used the current time.
Admin Access Only Functionality	The admin access-only function uses a decorator to check whether a user has the admin access column within the Users table set to true. If the user has admin access, they can see and enter pages that other users are not permitted to enter. If a user does not have admin access, they will be unable to see admin pages within the navbar. If the user attempts to access an admin page via the search bar, they will be returned to the home page with a message stating they are not permitted to enter this page.

Overall, my experience as the principal backend developer has been a crucial chapter in my development as a programmer and a key building block in my career and achieving my goals. Designing and developing these features for the Mens Shed web application has been a challenging yet rewarding experience with many new skills learnt along the way. In summary I feel as though this has been one of the most important tasks set within my career so far and look forward to what comes next.

Secondary Backend Developer

As the secondary backend developer Duncan Hastie, a Software Engineer Student joined the team at the last chance not being the most confident programmer but looked forward to the team environment.

He focused on building the database system for the application and inputting the data into the database. If behind on a week's goal would sure to be sure to have it done by the next

meeting at the latest and was helped by the Principal Backend Developer when it asked for to try and make it easier for a smooth integration of the database into the website.

Over the course of the project, he gained experience in working with multiple people doing different his small part to the overall whole Application.

Principle UI Developer

Joe Black took the role of primary UI developer. Joe is a twenty-year-old student with no prior industry experience, so leading the UI development team was a new and challenging task. However, Joe was happy to take on this role as someone who has an interest in web development, believing this would give useful insight into real-world experience.

The UI development team was responsible for creating the interface with which the client and end users would ultimately interact. This role also involved organising weekly meetings with the UI sub-team, ensuring that week-by-week goals were achieved, and presenting interfaces and designs to the client and other team members. In this role, Joe was also responsible for collaborating with the principal back-end developer to discuss how the interfaces would connect to the coded application and discuss any improvements which could be made to streamline this process.

Throughout the project's run, Joe took these responsibilities and ensured that the UI development came together to create a professional-looking interface. They gained useful experience to prepare for industry work. This was overall a highly insightful experience for Joe, thanks to the role he took on.

Secondary UI Developer

As a Software Engineering student, Daniel Beardmore took the secondary UI developer role. Daniel has little experience designing and developing UI but was eager to collaborate with Joe and learn from him and his web development experience.

Daniel focused mostly on implementing the designs based on Daniel and Joe's initial designs and refinements. He had to research HTML coding and even more into

implementing Jinja2 from Python. Daniel received significant help from Rory Mackintosh and PM John Johnston.

Overall, this project has helped diversify Daniel's skills, knowledge, and experience.

Security and Testing

Jonathan Cloke fulfilled the role of security and testing on the project as a student on the BEng (Hons) Cybersecurity & Forensics programme. Jonathan was added to the team by the PM at the team finalisation stage of the module.

With such a different educational pathway from the other students, Jonathan's primary role was to assess the technical teams' development work, highlight security considerations in the project documentation, conduct product testing, and document it.

The main benefits gained were insights into web-application development and team-working in general. Cybersecurity as a subject area tends to favour and foster a certain type of mentality, and invaluable experience was gained to balance and integrate this way of tackling problems into a multi-disciplinary team with diverse ways of approaching problems.

During the project, the PM had to take a step back for a brief time, and he had instructed that the ST should act as deputy PM until his return. Jonathan ensured that the team maintained the weekly meeting and followed the in-place documentation procedures as much as possible during this time. The PM had already established an efficient working framework, and this short step-in only added value to and enhanced the learning experience for the ST.

Backend and Database Development

During the backend development, many different technologies were used to design and develop the backend of the Mens Shed web app. Some of the main technologies used were the Python programming language, the Flask framework, the Linux command line to set up virtual environments to work on the web application and Jinja2, which allows the backend functions to communicate with the front end to give users visual functionality.

The Flask framework was a vital technology to use during development as it allowed for easier web development for the backend team. As the Flask framework has many built-in functions for easy web development, which were used throughout the development of the Mens Shed web application.

Jinja2 was one of the most important technologies during development from both a practical and group perspective. Jinja2 allows the backend team to create function variables that can be used by the front-end team in their HTML templates. This allowed the backend team to collaborate with the front-end team, bringing us together instead of two separate teams. Not only does Jinja2 allow for the backend features to have user functionality via the front end, but it also allows both teams to collaborate and share skills, making it a valuable learning experience for both teams.

Overall, team collaboration between the front-end and back-end teams was a valuable experience as it gave both teams a distinct perspective on the development of the Mens Shed web application. Not only was the practical experience of collaborating with the front-end team a valuable use of time, but the experience has also added a new social skill set for both teams when moving forward to future projects and other group tasks. In summary, collaboration with the frontend team has been a valuable and rewarding task in developing the Mens Shed web application.

User Interface Development

For this project, the User Interface was developed by a sub-team consisting of Joe Black and Daniel Beardmore. The UI sub-teams' task was to develop the interface with which the client and end users would interact.

The team considered several key factors in the development of this interface. Firstly, the interface should appear clear and professional. It should be straightforward to navigate – minimising clutter, and only contain relevant information. The site should also be easy to navigate; developing an interface for a user base which may not be experienced in technology was something to pay close attention to. Developing an interface with which the backend team could connect was also important.

In the earliest stages of implementation, the team began to design pages with raw HTML code, which was connected to a style sheet. Upon early reviews, the backend team suggested that the UI development team consider Bootstrap templates for implementation. Joe had some experience with Bootstrap templates and decided that this would be beneficial for the site, as Bootstrap allows web pages to be created using already existing templates and allows for complex features to be included, features which otherwise would have required a great deal more work in raw HTML, such as removing the need for a separate file for styling the pages.. Both teams agreed that this change allowed the creation of an interface which aligned with the key factors of appearing professional and readable.

Following a meeting with the client, it was suggested by team leader John and principal backend developer Rory that the UI team could convert the bootstrap pages to Jinja2 format. The Jinja2 format further streamlines implementation, meaning that the same file is used for each page, with only the page contents differing. The UI team agreed that while both members were unfamiliar with the method, it would prove extremely beneficial to both teams for connecting the backend application. Secondary UI developer Daniel transferred each page from the Bootstrap template to the Jinja2 format, which heavily reduced the lines of code required for the site. Both teams agreed that this was a strong improvement that would assist towards the finished application.

Security and testing

During the initial stages of design planning and development, there was little demand placed on this role. Security considerations were communicated with the team in order to guide them as much as possible in developing a secure final product.

Any user account system must have security designs in place for storing the passwords of the registered user accounts. Bcrypt was suggested and approved for the storing and authentication of user passwords in a hashed format.

Members' personal information stored by the application was recommended to have encryption applied to it to keep compliance with the Data Protection Act 2018 and the UK GDPR implementations in mind.

Input validation was recommended to receive attention to reduce the risks from malformed and malicious entries affecting or reaching the underlying database.

A user account password policy was recommended to be enforced to reduce the risks of insecure account passwords during account creation.

The team worked hard to consider these considerations alongside their development approaches.

During the project initiation, it was agreed that with the limits on time and other competing time interests for the team, that the framework for the application would be developed with the intention of other teams in the future adding to the development process.

Due to this agreement, the testing of the application was challenging to undertake, a test build was created by the team using the most current version of the application at that time, and the initial framework for testing was planned out to include three primary areas.

- Functionality Testing – Core functionality that was agreed to be completed during the project was evaluated.
- User Experience – The application was evaluated from the perspectives of the potential end-users. Due to the application still being in development, this testing was unable to be fully completed and requires revisiting after future development cycles.
- Security Testing – This testing area had three planned objectives, firstly was to assess if the security considerations identified in the project initiation report had been implemented, secondly to scan the web application for potential vulnerabilities and weaknesses and thirdly to perform dynamic security testing on the application from the perspective of a threat actor.

  Security Testing had to be adjusted as the application required more development before being assessed in this way. The test build provided was deployed using the PythonAnywhere platform which would not be used to host the app in the future, due to the public nature of this web server only limited testing could be attempted and limited value was able to be gained with the current version of the app.

  The security testing needs to be completely redone once the application has reached the stage of being ready for live deployment and conducted on the privately owned intended host infrastructure.

Stakeholder Evaluation

The stakeholder evaluation was conducted to establish satisfaction of the project delivery 4 question were posed as to the effectiveness of the project progression. The questions were see table 4

1	The project documentation that you saw
2	Communications with the team
3	The quality of the work that you have seen
4	The pace of the project

Table 4 Stakeholder Evaluation Questions

Each question was scored out of 10 at the end of each month's cycle over the length of the project. The results show an overall 80% sense of satisfaction which is confidently a positive review the details of the evaluation can be seen at Appendix 6. The main recommendations for improvement are centred around effective communications and the recording of the communications.

These results highlight some weaknesses in individuals and the method of the communications used to share information. The teams Chat platform was used in this project as the entire timeline is available throughout the development cycle and transparent for all users. While it has not been mentioned directly, confidence amongst the younger team members may have also contributed to the overall scores in the evaluation.

Stakeholder List

The stakeholders for this project include the following personnel.

Role	Name	Organization
Project manager	John Johnston	Napier University PT
Main Client Contact	Ron Skirving	Men's Shed Committee
Sponsor	Iain Donald	Napier University Tutor
Project Team	Jonathan Cloke	Napier University PT
Project Team	Rory Mackintosh	Napier University PT
Project Team	Joe Black	Napier University PT
Project Team	Daniel Beardmore	Napier University PT
Project Team	Duncan Hastie	Napier University PT
Men's Shed	Committee members	Men's Shed Committee

Table 5 Stakeholder List

Appendix 1 Follow-Up Register

Type	Cause	Effect	Impact	Likelihood	Importance	Response	Response	Custodian
R	Coding	Inefficient application and failures. Incorrect programming, being under pressure	70	50	2	Avoid	Testing and Adjust	JJ, RM, JB, JC, DB, DH
R	Joining coding elements between team members	Inefficient application and failures. Incorrect programming, being under pressure	70	50	2	Avoid	Testing and Adjust	JJ, RM, JB, JC, DB, DH
R	SQL Scripting errors	Inefficient application and failures. Incorrect programming, being under pressure	70	50	2	Avoid	Testing and Adjust	JJ, RM, JB, JC, DB, DH
R	Broken Object Level Authorization	Integrity of the site being compromised	60	30	1	Avoid	Testing and Adjust	JJ, RM, JB, JC, DB, DH
R	Broken User Authentication	Unauthorised access to the site	50	30	1	Avoid	Testing and Adjust	JJ, RM, JB, JC, DB, DH
R	Broken Function Level Authorization	Application failure	40	30	1	Avoid	Testing and Adjust	JJ, RM, JB, JC, DB, DH
R	Security Misconfiguration	Application security compromise	40	30	1	Avoid	Testing and Adjust	JJ, RM, JB, JC, DB, DH

Appendix 2 Project Final Close-Out Peer Review

Reviewer: Luke Taft **Team: 49**

Reviewee: John Johnston **Team: 13**

Date of review: 28/4/2023

Project description

Reviewer's comments and recommendations

There are a few small grammatical errors in the project description: "Following the initial between the university…" It is assumed that the sentence is referring to the initial meeting held between the university team and the client. Another small misspelling in the 2nd paragraph. "…for both embers and the management..." Assumed to be "…members and…" These errors are just small typos though and do not detract much from a very detailed and highly descriptive paragraphs defining the project. Only recommendation is a thorough proof reading.

Response and actions taken

Following the comments above the document has been spell checked and the grammar checked to correct the mistake and grammatical errors

Deliverables map

Reviewer's comments and recommendations

The deliverables map is an excellent summation of the different deliverable elements of the proposed application. Highly professional. I cannot think of a single criticism. Well done.

Response and actions taken

None

Follow-up register

Reviewer's comments and recommendations

The follow-up register could use a little differentiation regarding the custodian for each item. Is every team member responsible for each possible problem encountered? It is possible that is the team doctrine, and in that case, the follow-up register says exactly what it intends to say.

Response and actions taken

The Register reflects the responsibility placed on all members of the team to communicate findings results changes and issues as they arose. The team accepted their commitment to deliver following this methodology.

Quality of document (clarity, presentation, etc.)

Reviewer's comments and recommendations

The document is incredibly detailed and well-written. No facet of the project is left unclear, and there are several extra steps the team has taken to stand out amongst their peers—a highly organised, visually appealing document detailing a successful project. Well executed.

Response and actions taken

Thank you! This was truly a team effort and all participants deserve the plaudits.

Appendix 3 Client Project Final Report Feedback

From: Delta Digital

To: Johnston, John

Subject: Re: Final Report Draft for Client Review

Date: 03 May 2023 23:04:19

Hi John

The Development Team has succeeded in delivering the Must Haves in the time allocated to this project. To demonstrate the possibility of going above and beyond it would have been useful to see some "Should haves" to understand the difference between them and the "Could haves" but also to document exceeding expectations in the tight timeline.

"The limited time would be better spent on creating a foundation piece of work that could be developed further by subsequent student project teams."

A "Should have" might be a commitment to produce a project handover to next year's students, perhaps following a commitment from Course Leaders to submit a development plan to next year's students based on the work carried out so far. I high mark would rubber stamp that commitment.

A "Could have" might be user training sessions on the user interface. These might be moved to must haves in the next round of development. These may be activities carried out by an L&D department in a bigger organisation but more likely to be carried out by developer to end user in this case.

I am very impressed with reports over the course of the project and it demonstrates what can be achieved when employing a project manager with some life experience in the real world. There is a level of ability that can only come from experience and not from higher education. It is not alway down to us all, as individuals, to choose which order we do things. John is a great ambassador for Life Long Learning and I am sure he will go on to achieve great things perhaps with some of the team he has worked with on this project. I already have a commercial project lined up! What could be a better reward for study than employment in the field of your subject.

It is clear that good project management is key to defining and delivering an agreed objective in a given timescale. If they are SMART objectives there is a greater opportunity to succeed.

Cheers

Ron

Appendix 4 Lessons Learned

After doing an evaluation of the report and the feedback from the stakeholders, there are some clear lessons learnt from the group project:

A. Communication is the key to managing a project.
B. Keeping comprehensive records of meetings, conversations and interactions is important to track arising issues and good practices alike.
C. Regular Meetings to discuss emerging technologies, opportunities and issues is important to the drive for success.
D. Each stakeholder involved in the development of a project must be accountable for their contributions. A form or some kind feedback mechanism should be introduced at the start of the project to record, provide feedback, and assess progress of their deliverables.
E. To maintain momentum and enthusiasm over a protracted time period it is important to celebrate the milestones and the successes along the life cycle of the project.
F. Maintaining a positive outlook regardless of issues encourages participation as apposed to criticism or any kind of “Stick “.

Appendix 5 Evaluations

Month end scores

	Mth 1	Mth 2	Mth 3	
John	30	32	32	
Rory	30	29	32	
Jonathan	32	34	34	
Joe	28	34	35	
Duncan	31	32	33	
Daniel	34	38	35	
Sponsor	29	31	31	
Client	30	34	31	
Totals	244	264	263	of a possible
				320
Percentage	76%	83%	82%	

Comments

John	The team has worked very well together. By
Rory	None
Jonathan	None
Joe	None
Duncan	None
Daniel	None
Sponsor	As a student project using University systems
Client	

Appendix 6 Contribution Spreadsheet

Week	Date	JJ Project manager	JB (PUD)	JC (ST)	RM (PBD)	DB (SUD)	DH (SBD)	Total
1	16/01/2023	71%	28.6%	0.0%	0.0%	0.0%	0.0%	100.0%
2	23/01/2023	73.2%	10.1%	13.4%	0.0%	3.4%	0.0%	100.0%
3	30/01/2023	46.4%	2.6%	42.6%	0.0%	8.5%	0.0%	100.0%
4	06/02/2023	21.1%	20.4%	15.7%	14.3%	19.0%	9.5%	100.0%
5	13/02/2023	19.2%	25.3%	16.2%	10.1%	25.3%	4.0%	100.0%
6	20/02/2023	11.7%	23.3%	3.9%	31.1%	24.3%	5.8%	100.0%
7	27/02/2023	5.0%	0.0%	30.0%	0.0%	40.0%	25.0%	100.0%
8	06/03/2023	19.1%	38.3%	17.0%	0.0%	0.0%	25.5%	100.0%
9	13/03/2023	55.6%	33.3%	11.1%	0.0%	0.0%	0.0%	100.0%
10	20/03/2023	9.8%	15.7%	3.9%	54.9%	0.0%	15.7%	100.0%
11	27/03/2023	9.4%	11.3%	3.8%	75.5%	0.0%	0.0%	100.0%
12	03/04/2023	2.4%	0.0%	0.0%	97.6%	0.0%	0.0%	100.0%
13	10/04/2023	0.0%	0.0%	0.0%	100.0%	0.0%	0.0%	100.0%
14	17/04/2023	11.0%	3.7%	12.1%	73.3%	0.0%	0.0%	100.0%
15	24/04/2023	6.8%	5.5%	30.1%	54.8%	0.0%	2.7%	100.0%
16	01/05/2023	100%	0%	0%	0%	0%	0%	100%
17	08/05/2023							
Contributions		28.9%	13.6%	12.5%	32.0%	7.5%	5.5%	100.0%

Appendix 7 Main Python Code. (views.py)

Code

```
#This is the file that contains all the logic for each page route


from flask import render_template, request, session, redirect, url_for, flash
from mens_shed import app, bcrypt
from mens_shed.forms import RegisterForm, LoginForm, ToolForm, BookingForm
from flask_login import login_required, login_manager, login_user, logout_user,
current_user
from mens_shed.modules import User
from datetime import datetime, time
import sqlite3
from mens_shed import app
from functools import wraps

#Set up database connection
def get_db_connection():
    conn = sqlite3.connect('database.db')
    c = conn.cursor()
    conn.row_factory = sqlite3.Row
    return conn

#This is the home page route
@app.route("/")
@app.route("/home")
def home():
        if current_user.is_authenticated:
            return redirect(url_for('about'))
        return render_template('home.html', title='Home')

#This is the about page route
@app.route("/about")
```

```
def about():
    return render_template('about.html', title='About')

#This is the booking page route
@app.route("/booking", methods=['GET', 'POST'])
def booking():
    conn = get_db_connection()
    cur = conn.cursor()
    form = BookingForm()

    #Get the curret time
    now = datetime.now()

    #Convert current time to a string
    now_str = now.strftime('%Y-%m-%d %H:%M:%S')

    # Print the current time string
    print('Current time:', now_str)

    # Execute delete query
    cur.execute('DELETE FROM Bookings WHERE booked_out_till < time(?)', (now_str,))

    # Print the number of rows affected by the delete query
    print(cur.rowcount, 'rows deleted')

    conn.commit()

    #Get the details on the new booking from the form
    if form.validate_on_submit():
        name = form.name.data
        item_id = form.item_id.data
```

```
        #Covert to String
        booking_date = datetime.strptime(request.form['booking_date'], '%Y-%m-%d').date()
        start_time = datetime.strptime(request.form['start_time'], '%H:%M').time()
        end_time = datetime.strptime(request.form['end_time'], '%H:%M').time()

        # Format the time objects as strings in the correct format
        start_time_str = start_time.strftime('%H:%M:%S')
        end_time_str = end_time.strftime('%H:%M:%S')

        # Format the datetime object as a string in the correct format
        booking_date_str = booking_date.strftime('%Y-%m-%d')

        # Check if there are any existing bookings that overlap with the new booking
        existing_bookings = cur.execute("SELECT * FROM Bookings WHERE itemID=? AND
Booked_for_day=? AND ((Booked_for_start>=? AND Booked_for_start<?) OR
(booked_out_till>? AND booked_out_till<=?) OR (Booked_for_start<=? AND
booked_out_till>=?))", (item_id, booking_date_str, start_time_str, end_time_str,
start_time_str, end_time_str, start_time_str, end_time_str)).fetchall()

        if existing_bookings:
            # If there are overlapping bookings, display an error message
            flash('Sorry, that item is already booked during that time slot. Please choose a
different time.', 'danger')
        else:
            # If there are no overlapping bookings, add the new booking to the database
            cur.execute('INSERT INTO Bookings (userName, itemID, Booked_for_day,
Booked_for_start, booked_out_till) VALUES (?, ?, ?, ?, ?)', (name, item_id,
booking_date_str, start_time_str, end_time_str))
            conn.commit()

            # Get the updated list of bookings
            booking = cur.execute('SELECT * FROM Bookings').fetchall()

            # Redirect to the bookings page to display the updated list of bookings
```

```
            return redirect(url_for('booking'))

    #If no new booking has been submitted, display the list of current bookings
    booking = cur.execute('SELECT * FROM Bookings').fetchall()
    return render_template('booking.html', title='Booking', booking=booking, form=form)

#This is the tools library page route
@app.route("/tools")
def tools():
    form = ToolForm()
    results = []

    if form.validate_on_submit():
        search = form.search.data

        conn = sqlite3.connect(app.config['DATABASE'])
        cursor = conn.cursor()

        cursor.execute("SELECT item, catagory, date_listed, picture FROM Tools WHERE
item LIKE ?", ('%' + search + '%',))
        results = cursor.fetchall()
    return render_template('tools.html', title='Tools', form=form, results=results)

#This is the resource page route
@app.route("/resource", methods=['GET', 'POST'])
def resource():
    return render_template('resource.html', title='Resources')

@app.route('/search', methods=['GET', 'POST'])
def search():
    form = ToolForm()
    results = []
```

```
    if form.validate_on_submit():
        search = form.search.data

        conn = sqlite3.connect(app.config['DATABASE'])
        cursor = conn.cursor()

        cursor.execute("SELECT item, catagory, date_listed, picture FROM Tools WHERE item LIKE ?", ('%' + search + '%',))
        results = cursor.fetchall()

    return render_template('search.html', form=form, results=results)

#This is the support page route
@app.route("/support")
def support():
    return render_template('support.html', title='Support')

#This is the signup page route
@app.route("/signup", methods=['POST', 'GET'])
def signup():
    form = RegisterForm()
    if form.validate_on_submit():
        conn = sqlite3.connect(app.config['DATABASE'])
        c = conn.cursor()

        # Check if the username already exists in the database
        existing_user = c.execute('SELECT username FROM Users WHERE username = ?', (form.username.data,)).fetchone()

        if existing_user:
            flash('Username already exists. Please choose a different one.', 'danger')
            conn.close()
```

```
            return redirect(url_for('signup'))

        # Check if the email already exists in the database
        existing_email = c.execute('SELECT email FROM Users WHERE email = ?',
(form.email.data,)).fetchone()

        if existing_email:
            flash('Email already exists. Please choose a different one.', 'danger')
            conn.close()
            return redirect(url_for('signup'))

        # If the username doesn't exist, insert the new user into the database
        else:
            c.execute('INSERT INTO Users (username, email, name, address, password)
VALUES (?, ?, ?, ?, ?)',
                      (form.username.data, form.email.data, form.name.data, form.address.data,
bcrypt.generate_password_hash(form.password.data).decode('utf-8')))
            conn.commit()
            conn.close()
            flash(f'Account created for {form.username.data}!', 'success')
            return redirect(url_for('login_url'))

    return render_template('signup.html', title='Signup', form=form)

def admin_required(func):
    def wrapper(*args, **kwargs):
        # Get a connection to the database
        conn = get_db_connection()

        # Check if the user is logged in and has admin access
        if 'email' in session:
            admin_access = session['admin_access']
            cursor = conn.cursor()
```

```
            cursor.execute('SELECT admin_access FROM Users WHERE admin_access = ?',
(admin_access,))
            user = cursor.fetchone()
            cursor.close()
            if user and user[0]:
                # User has admin access - allow access to admin page
                conn.close()
                return func(*args, **kwargs)

        # User doesn't have admin access or is not logged in - redirect to error page
        conn.close()
        flash('You do not have permission to access this page.', 'danger')
        return redirect(url_for('home'))

    # Set the name of the wrapper function to the name of the original function
    wrapper.__name__ = func.__name__
    return wrapper

#This is the login page route
@app.route("/login", methods=['POST', 'GET'])
def login_url():
    form = LoginForm()
    if form.validate_on_submit():
        conn = sqlite3.connect(app.config['DATABASE'])
        c = conn.cursor()
        c.execute('SELECT * FROM Users WHERE email=? AND password=?',
                  (form.email.data, form.password.data))
        user = c.fetchone()
        conn.close()
        # Function to Kepp user details in the session
        if user:
            session['email'] = user[0]  # store user email in session
```

```
            session['admin_access'] = user[6]
            flash('You have been logged in!', 'success')
            return redirect(url_for('home'))
        else:
            flash('Login unsuccessful. Please check email and password.', 'danger')
    return render_template('login.html', title='Login', form=form)

#This is the Admin page route
@app.route("/admin")
@admin_required
def admin():
    return render_template('admin.html', title='Admin')

@app.route("/logout")
def logout():
    session.pop('email', None)
    session.pop('admin_access', None)
    flash('You have been logged out.', 'success')
    return redirect(url_for('home'))
```