UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2022

Title: New Honours Project Archive

Author: Richard Paterson

Project Abbreviation: PROJECTS

Supervisor(s): Hussein Suleman

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	20
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	0
System Development and Implementation	*0*	*20*	20
Results, Findings and Conclusions	*10*	*20*	10
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	0
Total marks		**80**	**80**

New Honours Project Archive

Transition of the Honours Project Archive to the SimpleDL Toolkit

Richard Paterson
Computer Science Department
University of Cape Town
Cape Town
PTRRIC011@myuct.ac.za

ABSTRACT

The computer science department of the University of Cape Town has a digital archive of the Honours students' projects. The current archive has existed since 2003 and while it is still functional, it has become outdated. By creating a new archive, there is the possibility to include new and more sophisticated techniques, that both improve the archive and address needs that did not exist 20 years ago. The solution for the new archive is to utilize the Simple DL toolkit to provide a backbone for the archive and build on new features into it to provide functionality that caters directly to UCT Honours students that does not exist in Simple DL. The software was successfully developed and passed all of the unit testing and happy path testing to ensure it met the functional requirements and through usability testing showed that users found the new archive to be a viable solution that was an improvement compared to its predecessor.

CCS CONCEPTS

• Digital Libraries • Data Management

KEYWORDS

Digital Libraries, Data Management, Simple DL, Archiving

1 Introduction

The Department of Computer Science at the University of Cape Town has stored and hosted a collection of honours projects since 2003 [1]. These archives display the heritage of the Computer Science department and allow visitors to see the development of technology and formation of knowledge that has happened over the past 20 years. This archive is accessible to all members of staff, students (current and past), and even general members of the public who are interested in accessing such a rich collection of works. While this archive allows such access, the archive's main software solution was created at the same time the archive itself began. And over the past 20 years this software solution has largely remained the same. This software is still completely functional, however, new and more advanced solutions are available now that could improve upon the current archive. One such solution is using the Simple DL toolkit. This project utilizes the Simple DL toolkit to create a new Honours Archive that has the potential to provide a better archive solution that would be more scalable, simpler to use, more resistant to poor networks and also allowing for a more modern and user-friendly interface.

This project was split into two independent sections, with Richard Paterson developing on the functionality involved with the uploading and managing of student submissions into the archive, and Simangaliso developing a way to view the submissions made by the students.

2 Related Work

2.1 User Centered Design in Digital Archives

When designing digital archives, consideration needs to be given to the skill level and experience of the users. In many archives there is a large variety of users who will be using it so more simple and commonly used features should be displayed clearly and more advanced features for experienced users can be made with less ease of use [2]. Users will also have different goals for using the system, which would motivate finding user requirements for the project and including them to ensure user satisfaction [2]. While much effort is needed to enhance usability, human centered design (HCI) should not be the only consideration. Care needs to be taken to not detract from functionality in favor of simplicity and ease of use. This design methodology is activity centered design (ACD) and is often used when functionality of the software is more important than the user experience [3]. ACD requires a longer learning curve, but it works better in complex systems [3]. This could be used for more advanced querying in the new Honours Archive, for experienced users to utilize more advanced functionality where functionality is more important than simplicity.

2.2 Simple DL

The Simple DL toolkit is a digital library software toolkit that is designed to work in low resource environments [4]. It is able to be run offline or on a local network and has shifted functions such as searching from the server to a JavaScript script run on the client. This reduces the load placed on the server and also reduces the minimum requirements to run this toolkit. Simple DL is also able to customize the output HTML files that display the archive's contents, typically through the use of CSS. Simple DL also uses XML and Excel to store metadata for the objects instead of using a database.

2.3 Bleek and Lloyd Collection

An example of a digital library that was developed without a database is the Bleek and Lloyd Collection. This collection is comprised of a large collection of "paper-based artefacts" that record the language and culture of the |xam speakers [5]. The software system that this collection was built on was designed according to the following principles: that software system mediation was to be avoided, that a network connection may be unstable, and that it would be preferable to create static representations through the pre-processing of data. Suleman [2007] demonstrated how XML, XSLT and XHTML could be used to develop a useful static, and portable digital library. XML-centric solutions are advantageous for libraries that are expected to be used for long-term data preservation purposes [5]. Linked XHTML pages were used from the XML source data and were stylized using XSLT stylesheets [5]. These hyperlinked pages utilized an Ajax-based search system that allowed the collection to be browsed and items accessed completely within the browser. Thus, making this approach useful for projects that do not necessitate a server-side search engine.

3 Problem Statement

Currently, the software used for the honours project archive is outdated and there is potential to use new and more modern tools and techniques to style and improve the aesthetic appeal of the submission process to make it more in line with the standards of websites today. There is a lack of functionality, the only function is to add a new archive and view existing archives. This caused issues such as users being unable to modify submissions. There is also a lack of security on the website as outside users can upload projects to the archive.

4 Aims

The aim of this project is to improve the digital Honours archive by developing a Web application with advanced features and a better user interface that provides a more pleasant user experience. The new Honours archive will include all of the functionality of the previous Honours archive, provide new and modern features, utilize newer technologies and tools, meet users' needs that did not exist 20 years ago and provide a better user interface.

5 Requirements Gathering

For the requirement gathering two major approaches were used, user personas and analysis of the current system. For the user personas, two goal-directed perspective personas were created for the two major functional components of the software. Along with that there was an analysis of the current UCT Computer Science Honours Archive to discover the current functionality and areas to be improved.

5.1 Current System Analysis

The main source of information about the current system was gathered through weekly meetings with the supervisor of the project, and current manager of the UCT Honours Archive, Hussein Suleman. The other form of analysis was interacting and viewing the current system.

The current system has the following functionality: it is able to display all of the students' archives and a brief description and some information about each of them. Each archive is able to be viewed by running a Perl script to show a view of the webpage included in each archive. The administrator is able to set up an upload page for students to submit their projects; within the upload the student can upload their project and project details and view what their webpage will look like. The files are stored in a zip file and metadata is then stored in a XML file. Both are saved in the Honours Archive and are able to be viewed by the public.

Current issues with the system mentioned by the current administrator of the archive include not being able to change a submission that has already been submitted. Instead, the administrator would need to manually delete submission, then the student can upload a new zip file with the changes made. Another area to improve was the aesthetic design of the website, as currently the archive has little to no styling and just includes an HTML page, with a specific font style and colour.

Some of the proposed improvements recommended included creating a space for students to upload and modify their projects, so as to remove the problem of incorrect uploads to the archive. There should also be a dedicated upload section that does not need to be manually enabled each year. A potential feature mentioned was the inclusion of thumbnails for each project to display next to each project's name and description.

Through interacting with the system, the above issues were noticeably apparent, along with another issue of access control. Any person who views the site while the Upload page is enabled is able to upload a project. So any user on the website not even in UCT would be able to submit files to be displayed. A related issue was that any submission by a student or user would be added to the archive and is viewable to the public. This would create a situation where UCT could potentially publicly host material that could negatively reflect on them. A potential solution to this would be to include a moderation stage in between the student upload and publicly displaying the archive, where an administrator would approve each project for public viewing.

There was also the concern about only a single student being able to view the submission before it gets uploaded. As each submission includes multiple student's work, it would be preferable for each involved party to be able to ensure that their correct section and part of the project is working before the final submission. This means that the workspace for the project mentioned above should allow access to the multiple students in the group.

5.2 User Personas

The type of user personas that were chosen was a goal-directed perspective. This type of user persona is defined by their personal and practically oriented goals for the software and focuses on the context, attitudes and workflows using the software [7]. This design is considered an efficient psychological tool with which to guide the design of a system.

There were two goal-directed perspective user personas developed, each dealing with a specific aspect of the system: administration and student submissions. The personas were split due to the complete separation of functionality and permissions of these users. A student will never be managing all of the projects submitted by other students, and an administrator will never be submitting their own Honours project to the archive. Thus, motivating the need to develop individual personas for these workflows and subsystems.

The Administrator

Name: Adam
Occupation: Computer Science Lecturer – Honours Course Convener
Demographic: 34 years old, lives in Cape Town
Attitude: Wants to make the process of managing the archive as simple and efficient as possible
What He Needs: He needs to be able to allow students to upload their archives, he needs to be able to see what they have uploaded and choose which archives are to be added to the website.

The Student

Name: Steve
Occupation: UCT Computer Science Honours Student
Demographic: 21 years old, lives in Cape Town
Attitude: Wants to be ensure that their project is perfect and uploaded correctly
What He Needs: He needs to be able to upload his project. He needs to be able to view what he has uploaded so far. He needs to be able to modify anything that is wrong with his project. He needs to be able to collaborate with his project partners

5.3 Functional Requirements

After developing the user personas and analysing the current system the following functional requirements were developed:
Student Upload:

- Provide a workspace for the project
    - Upload a project file
    - Upload metadata for a project
    - Upload a thumbnail for the project
    - View the current display of the project
    - Allow multiple students access to the workspace
    - Submit the project
- Provide email notifications of the moderation outcome
- Resubmit the project if it is denied during moderation

Administrator Management:

- Create Projects
    - Create a project for students to upload their submissions to
    - Assign students to project
- Moderate Submissions
    - View metadata and the project students submit
    - Approve a project and add it to the archive
    - Deny a project and allow a student to make the required changes
    - Send feedback in the form of an email to the students

It should be noted that some of the functionality is inherently provided by Simple DL such as login and user creation. Only slight modifications were needed to be made to the login to link a system email address and a new, more efficient system was developed for user creation. As such, login functionality was not included in the functional requirements for the development of this project as it is already provided.

5.4 Non-Functional Requirements

The non-functional requirement of this software is that the software should provide an aesthetically pleasing user interface, and users should find that the software is easy to use. The software solution should be reliable, being able to handle file uploads of large sizes.
Design
For the design of the project, firstly a use case diagram was created to show the different functional requirements for the system. Then a class package diagram was constructed to show the overall architecture of the system. Finally, the workflows for the individual components of the system, namely the Student Upload, Administrator Project Creation and Administrator Moderation, were created in the form of Sequence Diagrams.
Use Case Diagram

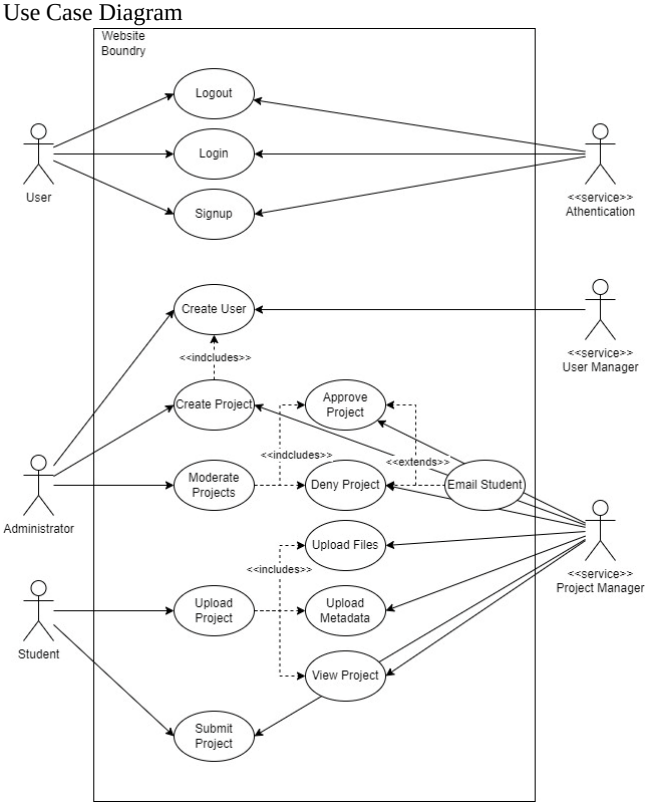


Figure 1: Use Case Diagram

Figure 1 shows the different actors in the system, with a general actor, user, which would include functionality that would be exposed to all of the different users just including basic functionality for login and account management. Then there is the

actor for an Administrator, which includes all of the different actions an administrator would perform: creating a project, adding students to a project and moderating submissions made by students. The last actor is the student who would be uploading a project or submitting the final version for moderation.

The use case diagram also includes which services would be responsible for each activity, with the authentication service handling the user account login and signup. The user manager would be responsible for any activities related to a user, such as creating a new student. The Project Manager would then be responsible for all of the activities related to the project, including submissions, uploads and moderation and many more.

Class Package Diagram

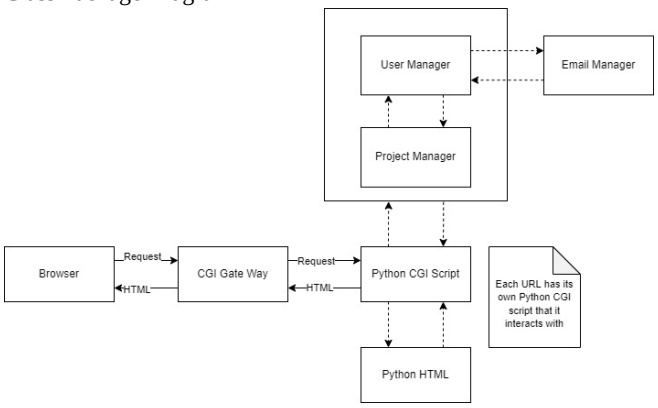


Figure 2: Class Package Diagram

This class package diagram (Figure 2) shows the basic architecture design of the system. A user's browser will make a request to the server, which will have Common Gateway Interface (CGI) enabled, an interface that enables Web servers to execute external scripts. Through the CGI gateway, individual python scripts will be executed to provide server-side functionality. These individual scripts will interact with the Project Manager and User Manager to make modifications and retrieve information, then with the information they return, the script will then utilize the pythonHTML package which will be responsible for converting and formatting the raw inputs into HTML output. The python script will then return all of the HTML back to the browser, allowing the browser to generate the resulting webpage. An Email Manager was included to allow the User Manager to email notifications to the users of the website.

5.3 Sequence Diagrams

An individual sequence diagram for each workflow was created showing the interaction and flow of information between the different classes. All these artifacts are included in Appendix 1, which show full size versions of each of the sequence diagrams produced. The sequence diagram for the Admin Project Moderation workflow has been included below to demonstrate and explain the system design.

Figure 3 shows the sequence diagram detailing the events of an administrator approving or denying a project. Firstly, the administrator will need to log in to their account. The account management is handled by a Perl script within the SimpleDL toolkit which will, on a successful login, create a cookie with the login details. When the cookie is populated, a Javascript script is run, which will populate the administrator menu for users who have administrator privileges. One such option in the menu is Moderate Projects. This option will run a python script to display all of the projects in the moderation queue. To achieve this functionality, the script will interact with the Project Manager and fetch all the pending projects.

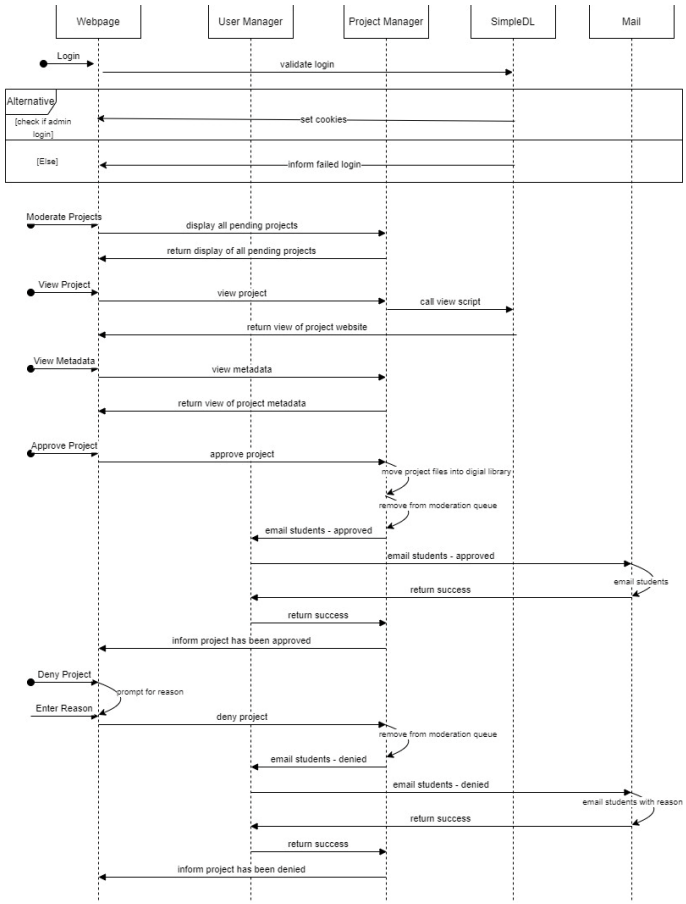


Figure 3: Sequence Diagram - Admin Moderation

For each project displayed there will be the option to view the webpage the student submitted along with the metadata. These requests get passed through to the Project Manager, which will call the view script from SimpleDL or will fetch and the projects metadata respectively. Each project will also allow the administrator to approve or deny the project, so once the submission has been viewed an action can be taken. If a project is approved the Project Manager will move the files into the correct directories as well as updating any indexing XML files; it will then remove the project from the moderation queue. Finally, it will email the students informing them of the positive outcome. In the case where a project is denied, the project will still be removed from the moderation queue, but the students will still be able to edit their files to make the requested changes. An email will be sent informing them of the negative outcome but will state the request the administrator has made for their project.

6 System Implementation

Many different tools and coding languages were used in this project to achieve the desired result. Python was predominantly used for the backend, along with some Perl scripts from the SimpleDL toolkit. For the frontend, HTML and CSS were used for the layout of the webpage and JavaScript for functionality. The individual implementation choices for each aspect of the project are listed below.

6.1.1 Backend Functionality

For this project the key software to be integrated with was SimpleDL. SimpleDL utilizes CGI to provide server-side functionality and renders HTML pages to display the archive to the user. To integrate with this software the decision was made to provide additional functionality through the use of CGI. However, it would move away from Perl scripts and rather use Python, which the developers were more familiar with. Thus, the backend is run with a variety of Python scripts.

6.1.2 Styling

SimpleDL also allowed for styling to be added in the form of CSS, so to make development more efficient and consistent, the decision was made to utilize the Bootstrap framework. By utilizing Bootstrap the website would also become responsive, allowing mobile devices to view it without necessitating the need for additional CSS styling.

6.1.3 Development Environment

Apache2 was used web as the server as this was recommended by the SimpleDL toolkit and the necessary setup instructions for how to run SimpleDL on Apache were included, thus, reducing the ramp up time for setting up the development environment.

6.1.4 SimpleDL modifications

Some small modification to the SimpleDL Perl scripts was necessary, such as changing the default password reset method as this required access to the UCT mail system. So minor changes were made in Perl to add on the functionality to send emails through a Gmail account, which is free to setup and use, as well as allowing SMTP access from an external application.

6.1.5 File Storage

The actual files in SimpleDL are stored as flat files and folders, with XML files created for indexing and information storage. These formats were used, for instance, when creating Python scripts to create new users. New XML files were made in the same format as SimpleDL was expecting, allowing SimpleDL to recognize and manage the new users that were created.

6.1.6 Version Control

For both version control and collaboration, GitHub was used to store the project files. A gitignore file was created to only upload the files created by the project and not SimpleDL code and a README was included, listing the installation and usage instructions. Regular commits were made to keep the project up to date for both members in the project as well as act as a backup in case of any unforeseen loss of data. This turned out to be instrumental as the hard drive containing the project got corrupted and had to be replaced, losing all of the data it contained. However, the project could still be recovered through the GitHub backup.

7 System Development

To develop the software, an iterative development cycle was used, in particular the agile methodology of feature driven development (FDD). An agile development process was chosen due to the core principles of agile including producing working software over comprehensive documentation as well as embodying adaptiveness as opposed to following a structured plan [8]. By doing this, the project could rapidly develop and test the core software, allowing the freedom to make changes over the development life cycle. It would also make the project more robust in the case of changing requirements uncovered during demos, supervisor meetings and usability testing.

Furthermore, the choice of FDD was made due to the nature of the project. FDD includes a five-step development cycle: Developing an Overall Model, Build a Features List, Plan by Feature, then iterating through Design Feature and Building the Feature [9]. The main advantages of FDD are that it provides clear tracking, making it easier to see how many features are developed and how many more still need to be completed. FDD also promotes individual code ownership as the main developer is responsible for designing, developing and testing each feature.

This aligned well with the project as each feature was largely independent and allowed rapid development of each new feature to quickly develop something that could be tested and modified. It also made it easier to clearly define individual scope within the project as each team member had a clear set of features to develop. It should be noted that FDD has less clearly defined iterations when compared to other agile methodologies. As such, iterations were rather a group of related features that achieved a specific overall functionality to the user.

The overall model building and feature planning are included in the requirements gathering and system design sections of the report. For the feature planning, the first to be developed were the features that other features depended on, for instance the uploading a file to a project required having created a project, and project creation required creating users to give permissions to. So, the development plan would be User Creation, Project Creation, Student Uploads, Project Submission, Archive Moderation and finally Archival of Projects.

Each iteration began with creating a lo-fi prototype, made in Paint.net a free photo editing software, of the user interface relating to the feature to be made. Then planning, through the use of flowcharts, how the system would behave. This was then formalized as how the information would flow between each part of the system in the sequence diagrams. Once all of the planning had been completed, the development of each feature followed. Upon completion, a manual integration test of the feature would be

done to ensure the correct changes were made in the system and the correct output was presented to the user. A more rigorous testing procedure was planned for testing all features when the entire system was completed; this was to allow the complete system to undergo user testing and have time to make the required changes afterwards.

7 Iterations

The iterations that were chosen were Student Uploading of Archives paired with Administrator Creation of Projects and Students as this was a dependency for the prior. The second iteration was the Moderation and Submission of Archives into the digital library. The third and final iteration consisted of styling and website layout.

7.1 Iteration 1

To upload a file to a project, there is the requirement of a project existing, and students assigned to that project to have permission to add to it. As such, the first task was creating a feature to create a project and students. A webpage for this was planned and a lo-fi prototype was created to show the information that would be included as well as the buttons and options that would be presented to the user; this is shown in Figure 4.

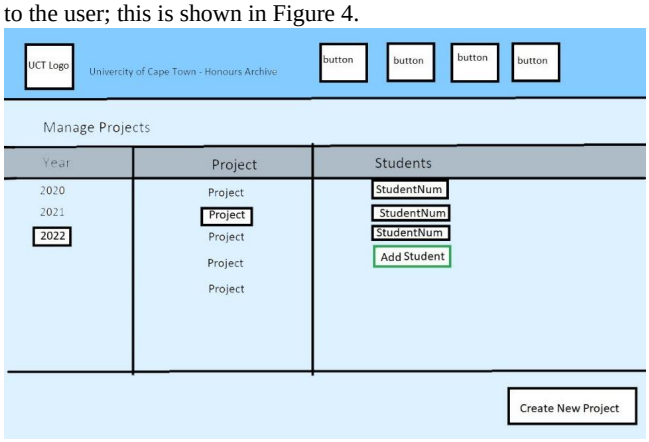


Figure 4: Manage Projects - Lo-fi Prototype

The expected behavior of the system is that when a user selects a year, the list of projects for the year would be displayed and selecting a project would in turn display a list of the students. A button to add a new student would be made available and clicking on a student number would delete a student from the project. When the user clicks on the Create New Project button, a new smaller webpage would be loaded containing a form to collect the project detail. As many of the webpages required individual forms with only different fields, a generic prototype template was created to be used for all of the smaller collection pages. This form is shown in Figure 5. When the form is submitted it would then show the outcome of the action.

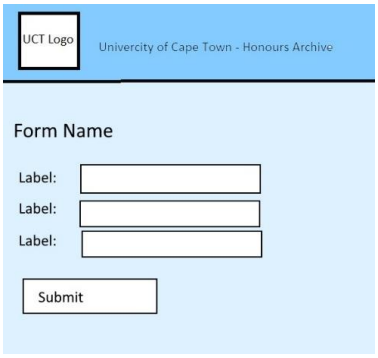


Figure 5: Upload Form - Lo-fi Prototype

The prototype, designed based of Figure 4, was developed and when presenting the demos, it was requested to make the deleting of users clearer, so the list of users was changed to be red outlined buttons that become filled with red when a user hovers over them. During the user testing one user commented that this behavior was both clear and aesthetically pleasing. Another change that was suggested was to include a CSV file upload that contains all of the projects for a year with all of the students. This would make the process more efficient as only a single upload would handle all of the project creations for the administrator and aligned with the view of the user persona Adam, the efficient administrator. It was also mentioned that UCT utilizes many Excel spreadsheets containing all of this data so it would not be difficult for an administrator to upload this particular document. As such, this feature was included in the system.

For the student project upload page, a similar process was followed with the prototype included below as Figure 6. When a user selects to upload metadata, the upload form page (Figure 5) is opened with the specified fields included.

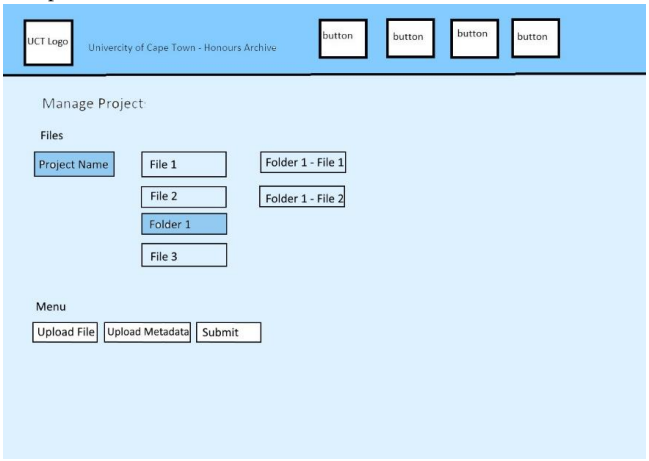


Figure 6: Student Upload Project - Lo-fi Prototype

To generate this webpage, the python script checks the cookies of the user to fetch the userID. It will then query with the Project Manager to fetch the project that the user is assigned to. It will then have the Project Manager cycle through the files in the specified projects directory to display them all to the user.

During the progress demo for the supervisor and second reader, it was mentioned that it was not clear how to utilize this webpage without it being explained so the page was modified to a tooltip that

when hovered over will display instructions on how to use the webpage and submit a project. It was also discovered that there needs to be a view webpage and view metadata so a student can see the result of their submission and that everything is correct before it goes to moderation. These features were then built into the system.

7.2 Iteration 2

Following the first iteration and changes made upon receiving feedback, the second major iteration of the software began focusing on the development of Moderation functionality. Firstly, a prototype was developed to plan the layout for the webpage, shown in Figure 7. Figure 7 shows how each project pending moderation will be displayed with an option for the administrator to view the metadata and the webpage to be shown. The administrator will then be able to approve or deny the project.

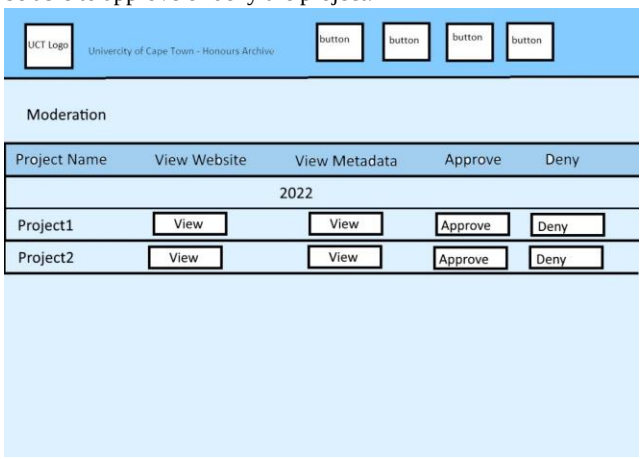


Figure 7: Moderate Projects - Lo-fi Prototype

The main difficulty of this iteration was creating the process to insert the files into the archive and updating all of the indexing. This involved discussions with the creator of the Simple DL software on how best to insert these files.

7.3 Iteration 3

For the final iteration, styling was done. This was by modifying the transformation script within SimpleDL on how it generates certain components on the website. Additionally, the pythonHTML script was modified to utilise the Bootstrap framework by inserting the additional class fields into the printed HTML statements. As all of the classes used the pythonHTML to handle most of their HTML generation, only slight modifications needed to be made to the existing python scripts for how they generated their webpages to use the new Bootstrap styling. The styling changes have been included in Appendix 2, showing how the use of Bootstrap was used to change the design of the webpage.

8 Testing

To perform testing on the system, both functional testing and non-functional testing were performed. The purpose of performing functional testing was to ensure that the behavior of the system was correct. This was done through integration testing, unit testing and system testing. Additionally, to ensure that the system was not only functional, usability testing and performance testing was done to test the system's ability to meet the non-functional requirements of being aesthetically pleasing, easy to use and reliable.

8.1 Functional Testing

8.1.1 Integration Testing

Integration testing is testing a group of logically connected modules within a system. Integration testing attempts to find issues in the interface, communication and data flow between modules. While integration testing does help identify if there are any errors within the group of tested modules, it is also more difficult to discover the cause of the error due to including a group of modules to test at once. To mitigate this effect, unit testing was also performed.

The process of integration testing that was used was manually testing the components through the webpage. All of the different functionality was testing providing different inputs to see how the system as a whole would process each input and how the data would flow through the system. Furthermore, the system was set to report any errors and exceptions to the webpage. This would allow easier tracing of any errors that individual modules encountered while interacting with each other. After the test was run, a manual check of the relevant storage locations within the server was done to ensure that all of the expected modifications were made such as saving an uploaded file. This was only used in the development stages of the project to ensure that the system was able to run properly, however, a more rigorous testing was done afterwards that was significantly more comprehensive.

8.1.2 Unit Testing

Unit testing involves testing the individual components of each of the functional modules. It provides very fine granularity testing to ensure the correct modifications and outputs are returned. For the unit testing, the python unittest module was used to test each method. Each method was tested for the expected output with normal data and relevant methods were tested with abnormal data to ensure correct graceful termination.

The modules used to convert and display HTML output were not tested using unittest, rather other testing practices in the system testing were used as these scripts result is HTML output and not actual data or modifications.

The unit tests were placed in a Test folder and some accompanying test resources were generated such as a demo testing project, csv file and other necessary testing items. A readme was included in the testing folder explaining how to run all of the tests and these tests could be automatically run to ensure that the entire system had the correct behavior at an individual method level.

8.1.3 System Testing

System testing is evaluating the system against the specified requirements. To run these tests the practices known as Happy Path Testing and Monkey Testing were used. Happy Path Testing is used to check if a system behaves correctly on a positive flow. To test this, the system is run with the expected valid inputs of a user and then checks are done to ensure that the outputs of the system meet

functional requirements. This process was manually done, and the checks were performed manually too.

For monkey testing, an independent honours student with a background in computer science was recruited and the different aspects of the system were tested with random inputs. The tester then looked at the resultant webpage to see if any errors occurred.

8.2 Non-Functional Testing

8.2.1 Usability Testing

To ensure that the system was not only meeting the functional requirements of the specification usability testing was performed to measure how users felt about the developed software. Usability testing consisted of gathering self-reported data on a 5-point Likert scale. Users were provided with a series of statements to which they reported to what extent they agreed to the provided statement with a 1 indicating "Strongly disagree" to a 5 indicating "Strongly agree" [10].

To conduct the testing, the tasks were split into two sections that related to two separate systems and managed by two separate users. The first section was student functionality and the second was administrator functionality. Each section had its own tasks and then questions for that given section. For both sections users were asked to perform a task then to respond to questions relating to the task that they just performed, then upon completing all of the tasks for a given section they would answer questions related to how they felt about that section of the system as a whole. By performing the testing in this manor, it would improve the quality of both task level data and overall usability data [11]. After answering the Likert scale questions, free response questions were asked to discover if there were any thoughts or opinions about the system that were not captured by the prior questions.

A total of 10 users were recruited for the usability testing. Each user signed a consent form, then had the instructions about how the test would proceed explained to them. A locally hosted version of the webpage was opened on a laptop and all of the necessary testing files were provided in a folder for testing submission and other tasks. Before each user attempted the test the prior data was removed from the system so as to maintain a consistent testing environment for each user.

For the testing questions roughly half of the questions were inverted to reduce positive bias, then after the testing was complete the negative question responses were inverted. This was done so as to make the analysis of the results clearer with higher scores indicating a more positive response.

8.2.2 Performance Testing

Performance testing was used to test the stability of the application under load. The focus of the performance testing performed was to ensure that the system was still reliable and consistent with large data flow. The specific form of performance testing to achieve this is called Volume testing, in which a large data size input is given to the system and the output is compared against the expected output at the end. To test this, a large file was uploaded into the system and all the functionality was tested on it to ensure that the system was able to correctly handle the file without crashing the system or corrupting the file. The file tested was a 5GB file which was as large a file as could be tested due to the system running on a Virtual Machine with only 12GB space remaining prior to the dummy file being generated.

9 Results

9.1 Unit Testing

A total of 49 tests were run and all of the tests passed. These tests encompassed each of the methods involved with user management, project management, email sending and cookie handling. This shows that the individual methods of the program run correctly and produce the correct output for every kind of manipulation or handling of the digital library. Included in figure 8 is the command line output demonstrating the passing of all of the tests.

```
richy@richy-virtual-machine:~/Documents/Honours-Project-Archive/public_html/cgi-bin$ python3 -m unittest discover test -b
set path to: /home/richy/Documents/Honours-Project-Archive/public_html/cgi-bin
set path to: /home/richy/Documents/Honours-Project-Archive/public_html/cgi-bin
set path to: /home/richy/Documents/Honours-Project-Archive/public_html/cgi-bin
set path to: /home/richy/Documents/Honours-Project-Archive/public_html/cgi-bin
.................................................
----------------------------------------------------------------------
Ran 49 tests in 11.784s

OK
```

Figure 8: Unit Testing Output

9.2 Happy Path Testing

Test	Result
Uploading a file to the project	PASS
Deleting a file from the project	PASS
Unzipping a file to be added to the project	PASS
Uploading metadata to the project	PASS
Uploading a thumbnail to the project	PASS
Submitting a project to be moderated	PASS
Creating a new user	PASS
Deleting a user	PASS
Creating a project manually	PASS
Creating a project from a CSV	PASS
Approving a project during moderation	PASS
Denying a project during moderation	PASS

All the above tests demonstrate the ability of the system to correctly handle the different functionality expected of the system given valid data and inputs. The above tests were performed with standard file sizes, including an archive taken from the current honours archive as a reasonable choice for testing a submission.

9.3 Monkey Testing

Through this testing, it was discovered that the Cookie Manger sometimes failed to fetch the correct student number. This was due

to an error in the User Manager class that was promptly fixed and then the error no longer persisted.

Some other errors were discovered when empty fields were entered in forms which necessitated adding extra checks to handle missing inputs.

There were also errors when a student tried to rename the root directory of their project, when a student tried to view metadata when there was no metadata already uploaded and also when special characters were used to create a student's student number.

Each of these errors required fixing and when tested again with the same inputs the correct outputs were observed.

9.4 Usability Testing

The total score for each question was calculated based of the sum of each participant's score for a given question. This was then divided by the number of participants to get the average for each question. A graph showing the average score for each question is shown in Figure 9.

Questions one and two were relating to the User actions such as uploading a project and questions three and four relate to design of the User actions subsystem. Questions five to eight relate to administrator actions and questions 9 and ten relate to the design of the administrator subsystem. A list of the transformed questions can be found in Appendix 3.

From the graph in Figure 9, it can be seen that nearly all of the responses averages for the questions were above 4, with only two being below 4. This tells us that the users, on average, did not have a poor experience utilizing the software as all of the results have been transformed so as to relate a higher score to a more positive experience and a lower score to a more negative experience. The average score for all questions was also 4.3 which shows us that on average users had a positive experience with the system.

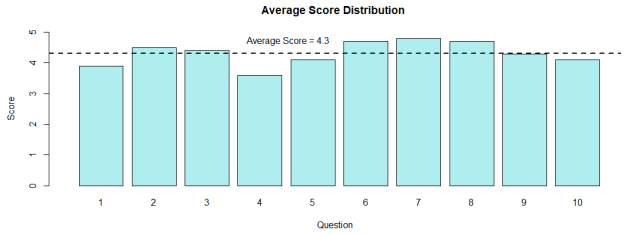


Figure 9: Average Score Distribution

Along with the average scores, the standard deviation for each question was also calculated. Standard deviation shows how much variability there is in each response, where a low standard deviation indicates that most of the results were very close together, and a large standard deviation indicates that the results were very spread out and different from the mean.

The standard deviations for the usability testing are shown in Figure 10. From this graph it can be seen that most of the results had a standard deviation around 1, with only 3 questions having a standard deviation greater than 1 but still being less than 1.5. This means that the results were often very close together and generally not being further than 1 score point away from the average. Some of the results were very close together, such as in questions seven and eight, which means that most users scored very similarly.

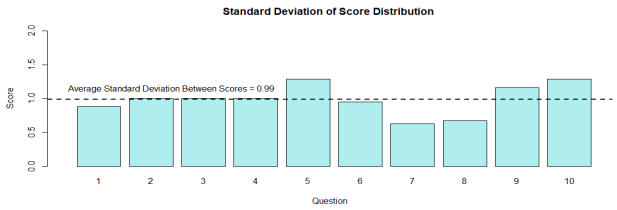


Figure 10: Standard Deviation of Score Distributions

Besides the average scores, the individual distributions for each question were generated and these can be found in Appendix 4. However, the distribution for question one has been included in the report to how such a distribution would look. This can be seen in Figure 11.

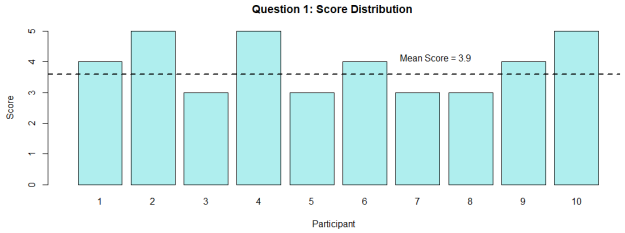


Figure 11: Question 1 Score Distribution

As the average scores for question one and two were around 4 and the standard deviation was less than one, it can be concluded that users found the user project submission process to be easy and efficient. Users also gave an average score, for the visual appeal of the user subsystem, of 4.4 coupled with a standard deviation of 1, it shows that most of the users agreed with the design being pleasing. With users also commenting that they found the design to be "visually pleasing" and that they liked the use of color. For question 4 about how much assistance users needed, an average score of 3.6 was calculated. This is the lowest average score out of all of the questions, and this is because of a single score of 1, indicating that participant 6 required significant assistance. Besides this user, all other users gave a score of 3 or higher with the mode score being a 4. Showing that no other users required significant help to understand this subsystem and the individual result may be an outlier. However, in light of the user needing help to understand the system a help button was included on the user upload page and clearer instructions were given. The process also was changed to automate a small step in the process, reducing the amount of clicks required to submit by 1.

For question five regarding how easy it was to create all of the projects for a given year, an average score of 4.1 was calculated, however, 60% of the users scored it a 5 stating that it was very easy to create all of the projects for a given year. As for questions six, seven and eight the average scores were 4.7, 4.8 and 4.7 respectively, with at least 80% of users scoring it a 5 on how easy and efficient they found these functionalities. As these scores are

all very high and standard deviations being low (all below 1.3), it can be concluded that users found all of the administrative actions such as project creation and moderation to be very efficient and simple to perform.

As for the visual appeal and ease of learning to use the system, average scores of 4.1 and 4.3 were calculated respectively. The standard deviation for both questions again was lower than 1.3 showing the majority of the results were close together and the modal response for both questions nine and ten were 5.

From the quantitative analysis performed, it can be concluded that most of the users found the new Honours archive to be simple to learn and use, while also being efficient and visually pleasing.

Some user comments from the free response questions included multiple users apreicating the color choice for the webpage, they liked the style and design approach of the banner and buttons. Only a single user said they didn't like the buttons available on the navigation bar as they did not show what they were looking for, however, no other user stated that they disliked anything in particular about the webpage. Many users also requested more clear naming for the nav bar links as well as help buttons on the user upload page as well as on the home page. Some of the names have been rechosen to improve clarity as well as a help button on the upload page. One user also mentioned that they would like the inclusion of a close button on the smaller popup webpages when the action is completed so as to clearly know they should close it which has been added.

A final comment mentioned by two users was that they did not like the homepage for the website, however, this was not included in the scope of this project and rather in the scope of Simangaliso who, at the time of the usability testing, had not completed and submitted his section to the shared repository. As such, the users were informed of the situation and these comments can be excluded.

9.5 Volume Testing

During the volume testing the system was able to handle all of the expected tasks, that being uploading the file, zipping the final archive including the file and moving the file. While the correct output was achieved, there were substantial delays while uploading and zipping the file, which led to the webpage becoming unresponsive. This is due to the tasks being handled not being asynchronous, which was a design decision so as to inform the user of the outcome of the actions as they are done. While it is possible to potentially improve the website by including processing messages using technologies such as AJAX, this is beyond the scope of this project and can be looked into as a future improvement.

10 Conclusions

After reviewing the results, it is apparent that the software managed to meet all of the functional requirements. The software was able to handle the student's upload, it was able to create new users and projects and the moderation was able to reject or incorporate projects into the archive. Additionally, the software also met the non-functional requirements of being reliable, easy to use and providing an aesthetically pleasing user interface. The website demonstrated its achievement of having an aesthetically pleasing user interface as the mean scores for the visual appeal of the user interface were above 4. Similarly, users also rated the ease of use to be around or above 4, and with only a single mean score of 3.6 for requiring little to no assistance navigating the website which is still within the agreement range of the Likert scale. The non-functional requirement of being reliable was also demonstrated through the successful handling of a large file.

Not only were the requirements met but also the aims of the development project. The software was able to encompass all of the functionality of the old honours archive by being able to upload a project, metadata for the project and allow a view of the project before the submission. It was able to include new features such as user and project management, facilitated creating a shared workspace for multiple students, allow editing of the project before submission and allow an administrator to approve of any projects being added to the archive. The software also includes new technologies such as utilizing the SimpleDL toolkit as well as styling frameworks such as Bootstrap. Users' needs that were met that the prior archive did not meet were providing a more pleasing user interface and user experience. Some users commented that they felt this solution was an improvement over its predecessor thus meeting the aim of providing a better user interface.

In conclusion, this software managed to meet all its aims and fulfil all of the development requirements set out for it and could potentially be used as a starting point for a new Computer Science Honours Archive for the University of Cape Town.

11 Future Work

A potential future development could include utilizing AJAX calls and making the webpage show a progress status during long periods of processing. Another potential development would be allowing a drag and drop feature for file uploads. There is also potential to focus on improving the security of the website by including HTTPS and modifying the verification method of cookies as this was not able to be done by the Python backend due to the reliance on a specific random number generator specific to Perl.

There is also potential benefit to conducting a user study to evaluate what names would provide the best clarity and improve the ease of use for users. As well as providing a video guide on how to upload a project for users that are very lost on how to use the website.

REFERENCES

[2] Blandford, A. 2004. *Usability of Digital Libraries (Editorial)*. International Journal on Digital Libraries.

[11] Chapter 6 - Self-Reported Metrics In Interactive Technologies Measuring the User Experience: 2013. *https://doi.org/10.1016/B978-0-12-415781-1.00006-6*. Accessed: 2022- 09- 10.

[9] Feature Driven Development (FDD) and Agile Modeling: 2022. *http://agilemodeling.com/essays/fdd.htm*. Accessed: 2022- 09- 09.

[1] Honours Project Archive: 2003. *https://projects.cs.uct.ac.za/honsproj/2003/*. Accessed: 2022- 09- 08.

[10] Likert Scale Response Options: 2006. *https://mwcc.edu/wp-content/uploads/2020/09/Likert-Scale-Response-Options_MWCC.pdf*. Accessed: 2022- 09- 09.

[8] Manifesto for Agile Software Development: 2022. *https://agilemanifesto.org/*. Accessed: 2022- 09- 09.

[3] Norman,, D. 2005. *Human-Centered Design Considered Harmful*. Interactions.

[7] Personas: 2022. *https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/personas*. Accessed: 2022- 09- 08.

[5] Suleman, H. and Phiri, L. 2012. *In Search of Simplicity: Redesigning the Digital Bleek and Lloyd*. DESIDOC Journal of Library & Information Technology.

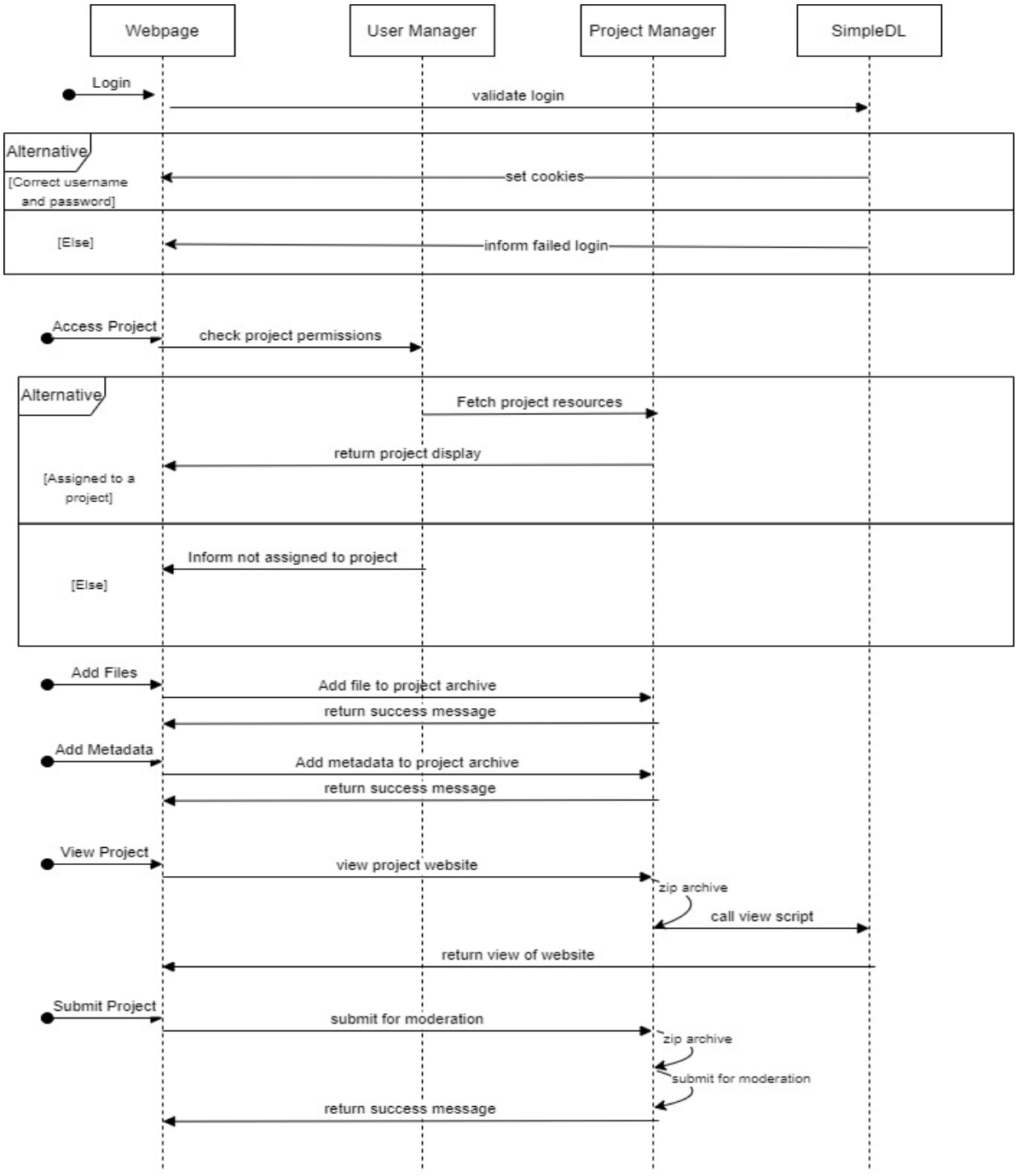
[6] Suleman, H. 2007. *An African Perspective on Digital Preservation*. Multimedia Information Extraction and Digital Heritage Preservation.

[4] Suleman, H. 2021. *Simple DL: A Toolkit to Create Simple Digital Libraries*. Towards Open and Trustworthy Digital Societies.

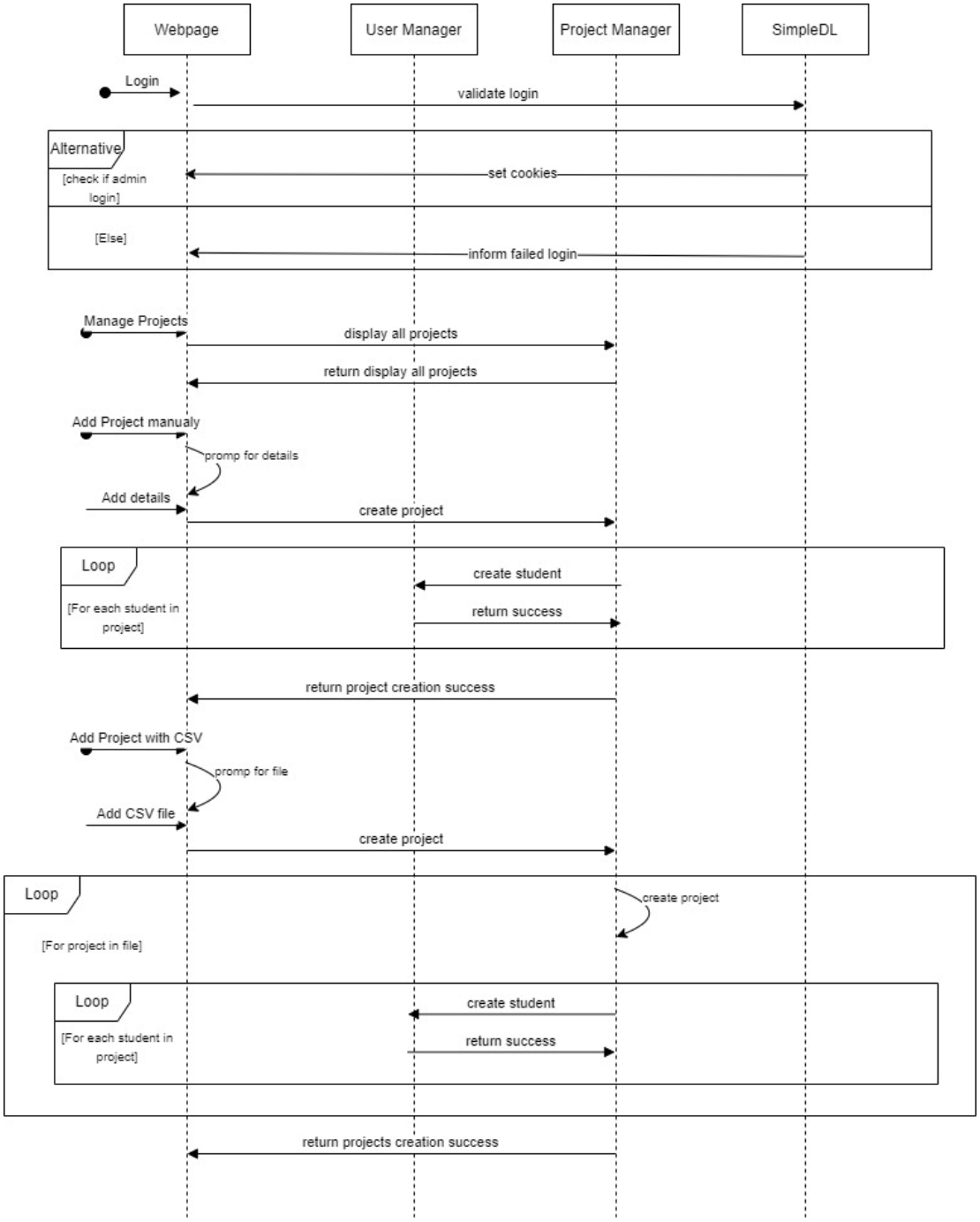
APPENDICIES

Appendix 1 – Sequence Diagrams

Appendix 1.1 Student View

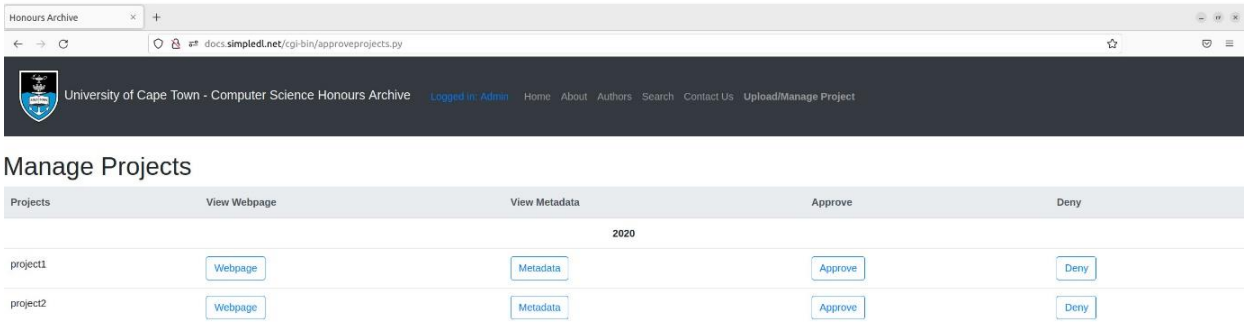


Appendix 1.2 Admin Project Creation

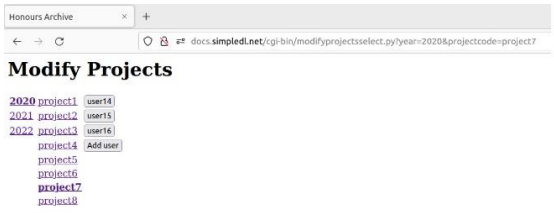


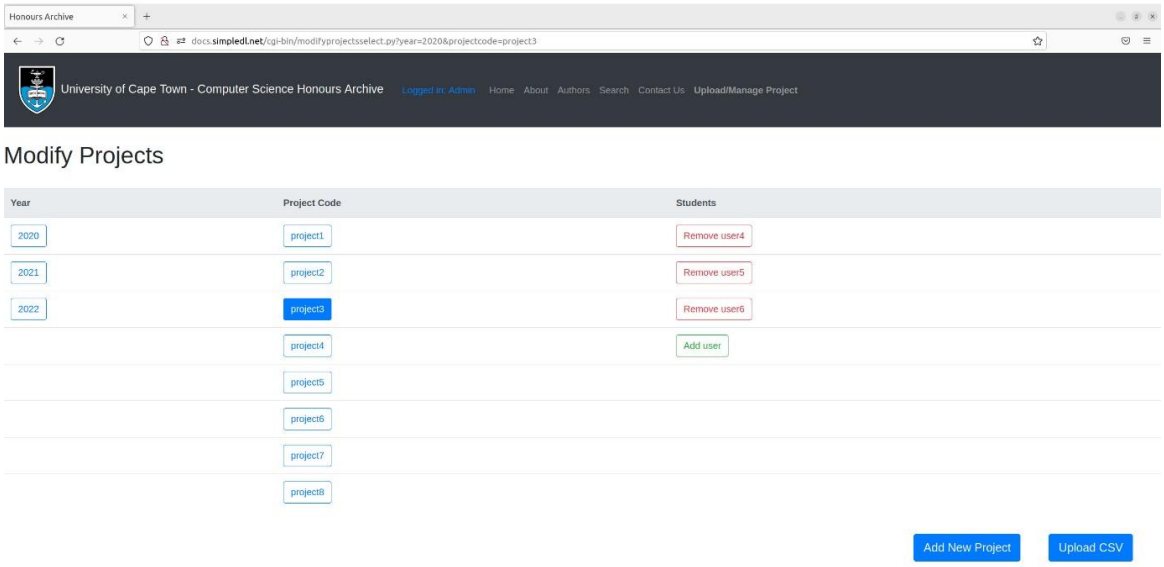
Appendix 2 – Styling Changes

Appendix 2.1 Approve Project Page

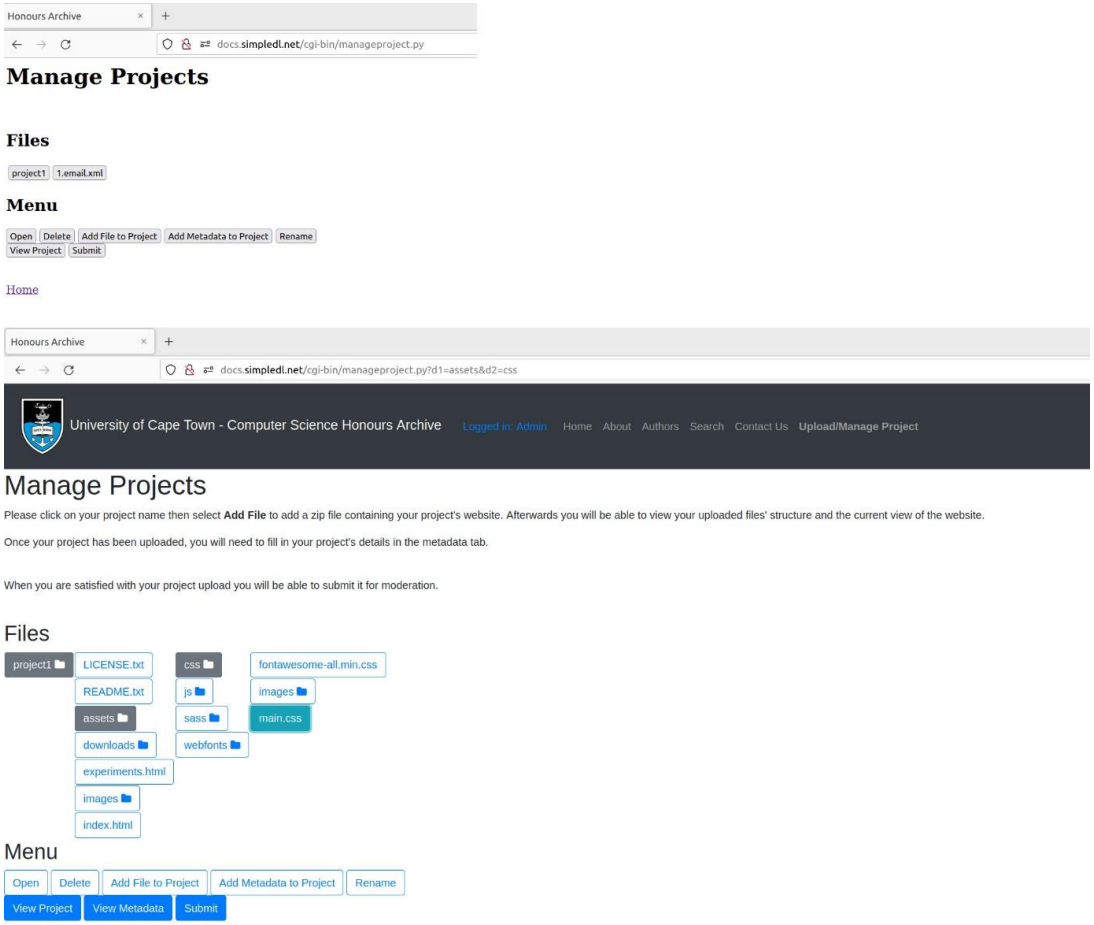


Appendix 2.2 Create Project Page





Appendix 2.3 Manage Project Page



Appendix 3 – Transformed Question List

User Actions

1 I experienced little to no difficulty uploading my project and metadata.
2 I felt that the process was efficient for uploading my project.
3 I found the design of the website to be visually pleasing.
4 There were very few points where I required assistance to understand how to use the system.

Administrator Actions:

Creating Projects:

5 I found it easy to create all the projects for the year.
6 I found the process to be efficient.

Approving Projects:

7 I found it easy to approve or deny projects that the student submitted.
8 I found the process to be efficient.

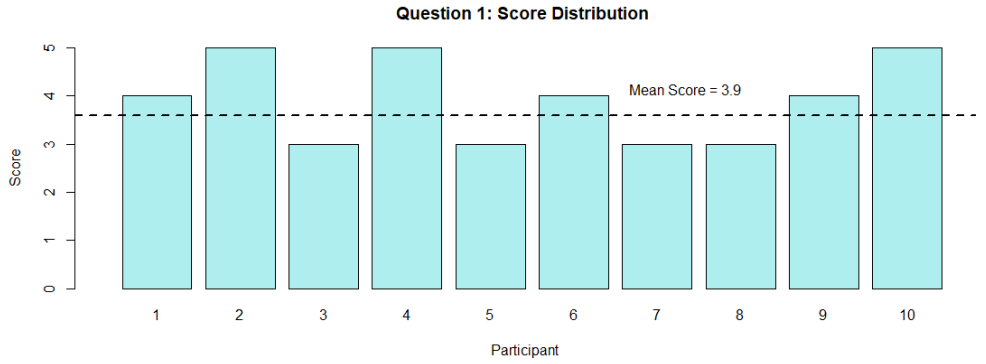
General Administration:

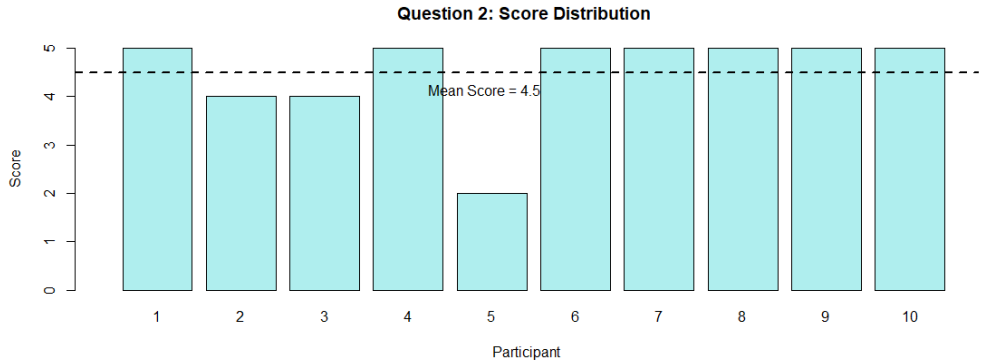
9 I feel that the system is easy to learn how to use.
10 I found the design of the website to be visually pleasing.

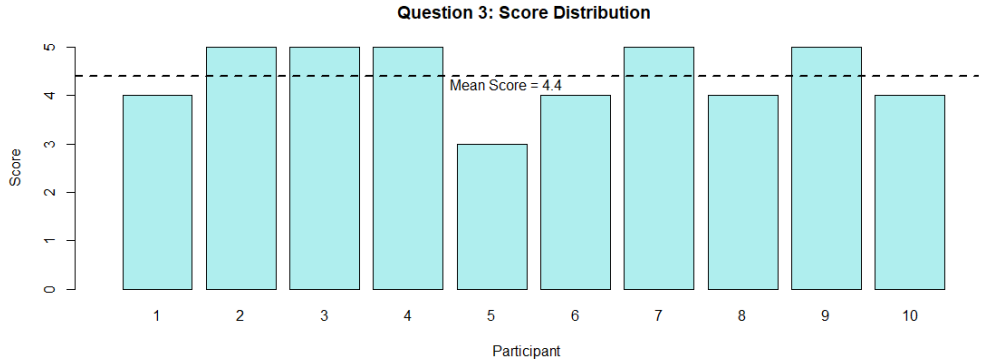
General:

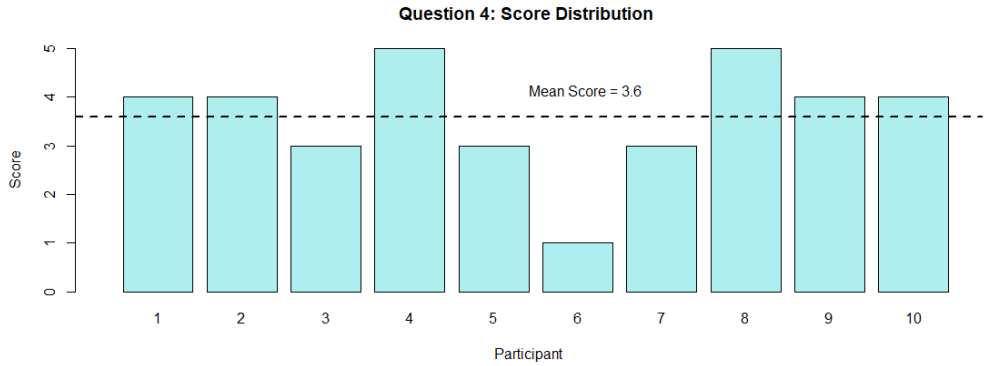
11 Was there anything you like to be improved on the website?
12 Were there any new features you would like to have been implemented?
13 Was there anything you particularly liked about the website?
14 Was there anything you particularly disliked about the website?

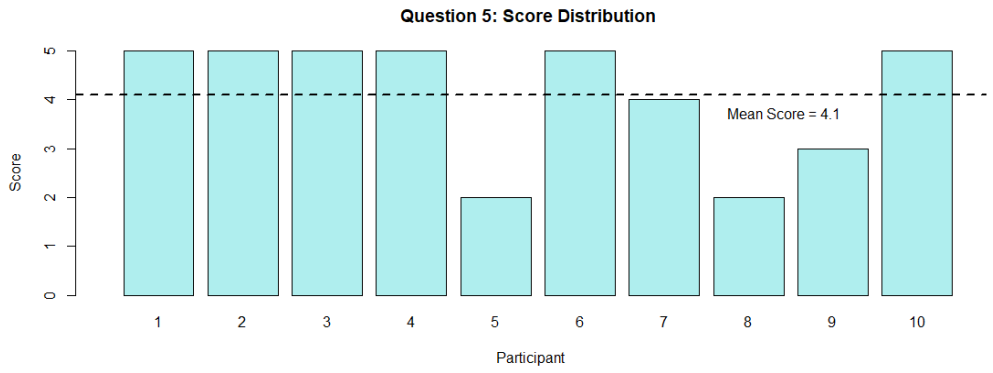
Appendix 4 – Usability testing graphs

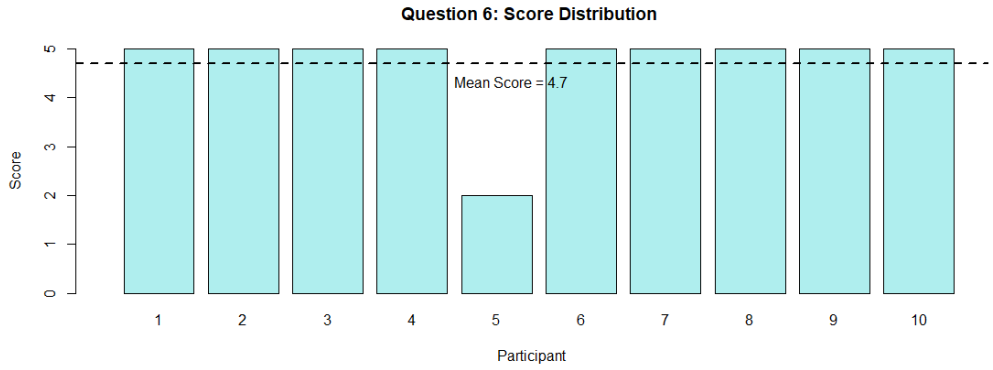


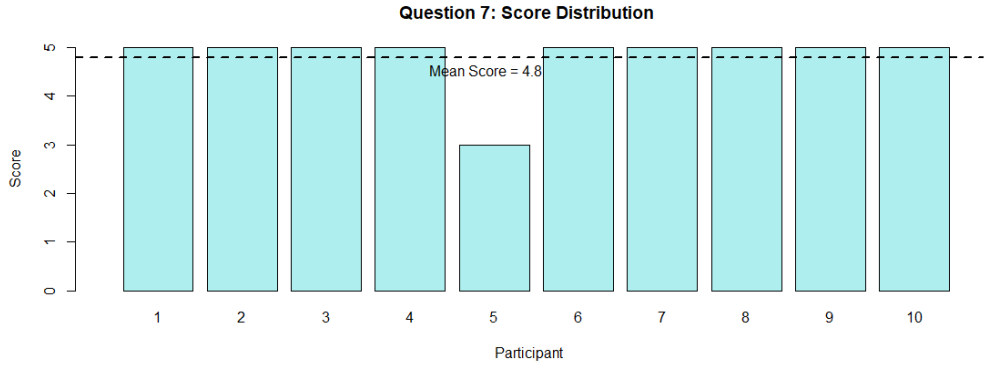


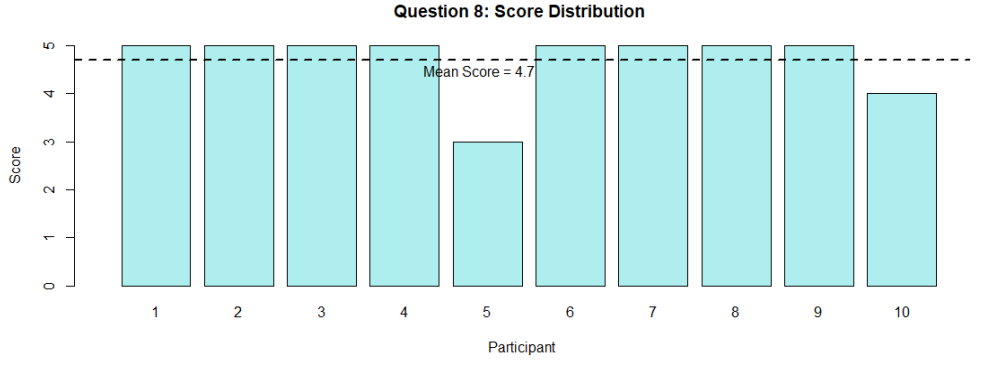


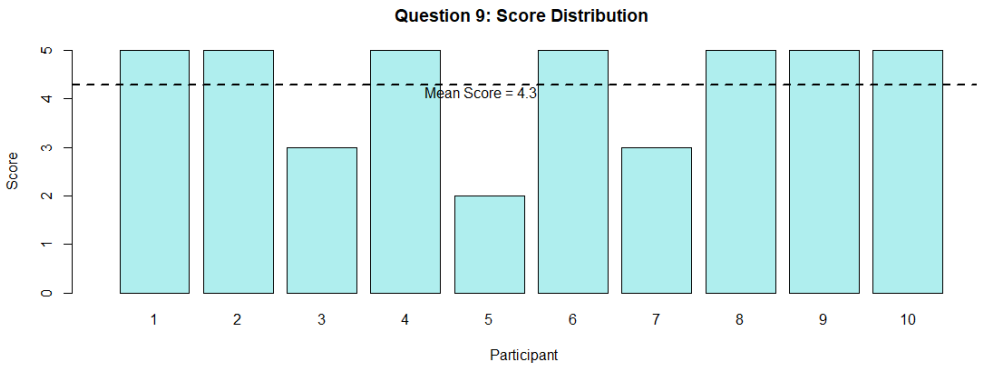


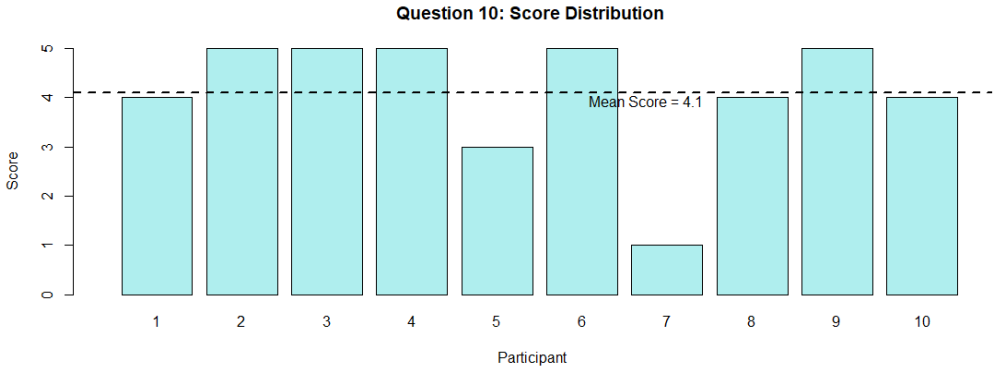

Question 5: Score Distribution
Mean Score = 4.1
Score
Participant
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10


Question 6: Score Distribution
Mean Score = 4.7
Score
Participant
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10


Question 7: Score Distribution
Mean Score = 4.8
Score
Participant
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10


Question 8: Score Distribution
Mean Score = 4.7
Score
Participant
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10


Question 9: Score Distribution
Mean Score = 4.3
Score
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10
Participant


Question 10: Score Distribution
Mean Score = 4.1
Score
0
1
2
3
4
5
1
2
3
4
5
6
7
8
9
10
Participant