Contribution Report

TAWE-Lib

Group 26:

Rhys Jacka (955475), Kevin Pan (969449),

Artiom Serstobitov (962863), Eduardo Camacho (966945),

Suman Gurung (965150), Dale Butt (957450).

Table Of Contents

Rhys Jacka's Contribution

In the implementation stage of the project I was assigned with overseeing the submission of A2. This includes checking up on the team to ensure the TAWE-Lib was complete to the standards of the specification; as well as the quality of the video and what is covered. Suman and I worked closely together to discuss what software we would use create the video as well as sharing ideas of how we could showcase TAWE-Lib in a professional manner but also a simple way for the audience to understand. We concluded we'd share this task and record sections, of which then I would edit the video ready for submission.

Additionally, I did also support the implementation of the users and librarians setting their own profile picture. I was supporting this through creating the class Drawing as well as writing its Javadoc. This class allowed the user or librarian to create their own profile picture if they didn't like the 6 images already available to them, which I also drew out to fit the style of the TAWE-Lib user interface.
Personally, I felt this was a class I really wanted to focus upon, as it was good opportunity for me to improve my programming skills while learning JavaFX. I was also meant to create the class Avatar. Avatar was where the user and librarians' picture would be stored as well as 6 other images they could choose from. However, during development of this class I realised it would be easier to save the images as files in a local repository as well as including the choice of 6 other images in the Profile Picture User Interface.
Additionally, I did also help find errors and bugs of the project to ensure it was up to submission standards.

Kevin Pan's Contribution

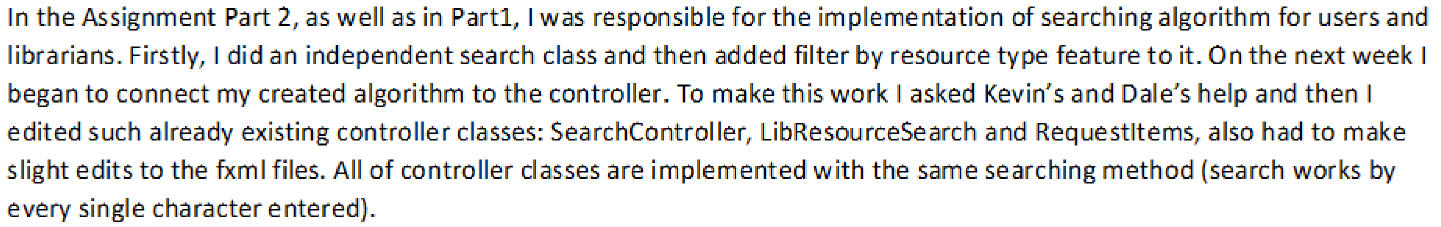
In the Design phase of the module, I help designed resources class with all its subclasses, searching and drawing pictures for the avatars. I did not continue with these classes during the implementation phase, instead I was creating athe user interface. I designed the graphical user interface and created many fxml files that display the scenes and all controller classes for handling data outputs and user inputs.

I have made the following scenes and respective controllers: AddFunds, Book, ChangePicture, ChangeResourcePic, CheckItemHistory, CollectItems, DVD, PayFines, GiveItems, HomePage, IssueDesk, Laptop, LibBook, LibDVD, LibHome, LibIssueDesk, LibLaptop, LibProfile, LibResourceSearch, LibUserSearch, Loan, LogIn, LookLib, LookUser, Profile, ReceiveFines, ReceiveReturns, Request, RequestItems, ReturnItems and Search.

I helped Eduardo to link his scenes with mine and updated some of his scenes to be more refined such as payFines and CreateResource. I have also assisted with other group members to help link their classes to work with the controller classes. During the linking with front end code with back end code, some were very difficult to link or even didn't link up completely.

I have also help written JavaDoc for all of the controller classes.

Artiom Serstobitov's Contribution



In the Assignment Part 2, as well as in Part1, I was responsible for the implementation of searching algorithm for users and librarians. Firstly, I did an independent search class and then added filter by resource type feature to it. On the next week I began to connect my created algorithm to the controller. To make this work I asked Kevin's and Dale's help and then I edited such already existing controller classes: SearchController, LibResourceSearch and RequestItems, also had to make slight edits to the fxml files. All of controller classes are implemented with the same searching method (search works by every single character entered).

Eduardo Camacho's Contribution

As the secretary of the group I took the responsibility of making all of the minutes and contribution reports and submitting all of them to blackboard.

I worked with Kevin to make the user interface so that the user has an interface to use the program and be able to use the code trough the user-friendly interface. I made half of the FXMLs for the UI and made some edits to Kevin's so that it had what was necessary on the page and got rid of any of the extra parts that were not necessary. Made controllers for create user and create resource as these were classes added on later I added controllers to them so that it can link to the rest of the Interface. I made adjustments to the design of the interface so that everything was aligned and looked consistent between the pages that the user can go through.

Suman Gurung’s Contribution

For my A2 implementation I have decided to attempt the following classes; Resource, DVD, Book, Resource, Copy and Laptop. I followed the UML diagram for those classes and thus implemented attributes, methods and constructors of those classes. The feedback from the A1 assignment suggested that the Resource hierarchy would not be completely correct. Therefore I took a different approach to the hierarchy for the classes. Also I included variables that would communicate with other classes. Dale have then made some modification to those classes later on by adding function for generating ‘copyID’ as well as setting up lists for communicating with other classes. As everyone else got their classes to be working as intended, I began recording a video that showed the features of the program and how to use them. After recording my video I then sent it to Rhys who then edited the video I recorded.

Dale Butt’s Contribution

In A2, Dale Launched a gitlabs project for the group to collaborate on & let everyone code at their own pace without delegating classes: he solely decided to do this based on past experience being aware of the different skill set between each member has.

Dale created the initial code for all the classes he designed in A1, this being:

- User
- Controller
- Loanrequest
- Librarian
-Library
- OnLoan

However, as he created the standalone classes and the controller hierarchy, he soon realised that the library would need more than what was documented. In the first attempt, the Library class contained preloading effects which created resources and users/librarian based on locally saved text files (CSV’s).

After Suman and Rhys completed their classes (Resource hierarchy & Fine) he realised how differed they would have to look so he collaborated with both members & changed the code to suit Library. After all classes were created, Dale with help from Artiom, went over all the classes and improved the overall style and logic of the code itself.

In the end, Dale thought it would be best to have data saving in a separate file because we could have some methods which would have better functionality, so a datastore class was made. This contained the methods: getASync, setASync, updateASync, appendASync with various private methods to support the main ones.

Dale also combines the front end to the back end of the project with help from Kevin. He decided to do this because he understood and knew the Library best.