UNIVERSITY OF TECHNOLOGY SYDNEY

ASSIGNMENT COVERSHEET

UTS: ENGINEERING & INFORMATION TECHNOLOGY

SUBJECT NUMBER & NAME	NAME OF STUDENT(s) (PRINT CLEARLY)	STUDENT ID(s)
41093 Software Engineering Studio 1A	*Mills, Chantel* *Przyrembel, Mattias* *Karim, Mohammad* *Jain, Pulkit* *Khalidi, Umair* *Ma, Yize*	13543638 13892409 13140429 13653934 14326336 13149594

STUDENT EMAIL	STUDENT CONTACT NUMBER

NAME OF TUTOR	TUTORIAL GROUP	DUE DATE
Tejbir Chopra	4	5/11/2021

ASSESSMENT ITEM NUMBER & TITLE

Assessment 4: Technical Support Documentation

☐ I confirm that I have read, understood and followed the guidelines for assignment submission and presentation on page 2 of this cover sheet.
☐ I confirm that I have read, understood and followed the advice in the Subject Outline about assessment requirements.
☐ I understand that if this assignment is submitted after the due date it may incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.

Declaration of originality: The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s) and has not been previously submitted for assessment. I understand that, should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with University policy and rules. In the statement below, I have indicated the extent to which I have collaborated with others, whom I have named.

Statement of collaboration:

Signature of student(s) ______MP_____CM_____UK______PJ_______YM_________MK_______ Date ________5/11/2021________

✂- -

ASSIGNMENT RECEIPT

To be completed by the student if a receipt is required

SUBJECT NUMBER & NAME	NAME OF TUTOR

SIGNATURE OF TUTOR	RECEIVED DATE

STYLE GUIDE for ASSIGNMENT SUBMISSION

Before submitting an assignment, you should refer to the policies and guidelines set out in the following:

- FEIT Student Guide
- UTS Library - referencing
- HELPS - English and academic literacy support
- UTS GSU - coursework assessment policy and procedures

Unless your Subject Coordinator has indicated otherwise in the Subject Outline, you must follow the instructions below for submission of assignments in the Faculty of Engineering and Information Technology.

Writing style

It is usually best to write your initial draft in the default settings of your software without formatting. Use the following guides in your writing.

Purpose and audience: use the correct genre and language style expected for the particular task.

Language: use 'plain English' for all technical writing. More information about this language style can be found at www.plainenglish.co.uk/free-guides.html.

Use spelling and grammar software tools to check your writing. Edit your document.

Standards: always use:

- Australian spelling standards (Macquarie Dictionary)
- SI (International System of Units) units of measurement
- ISO (International Organisation for Standardisation) for writing dates and times for international documents. For example **yyyy-mm-dd** or **hh-mm-ss**. However, for most applications it is more helpful to present the date in full as **26 August 2016**.

Graphics and tables should:

- be numbered
- have an appropriate heading and/or caption
- be fully labelled
- be correctly referenced.

Presentation

Unless otherwise instructed, all assignment submissions should be **word processed** using spell-check and grammar-check software. Work should be well **edited** before submission. Use the following default settings:

Page setup: set margins at no less than 20mm all around.

Paper: print on A4 bond, double-spaced and preferably double-sided, left justified.

Font: use the software default style to provide consistency. The recommended style includes:

- 10-12 pt font
- consistent formatting with a limited number of fonts
- lines no more than 60 characters (use wider margins or columns if you need to make lines shorter)

Header should include:

- your name and student number
- the title of the paper or task.

Footer should include the page number and current date.

Cover sheet and statement of originality: all work submitted for assessment must be the original work of the student(s) submitting the work. A standard faculty cover sheet (see over) must be attached to the front of the submission. Any collaboration between the submitting student and others must be declared on the cover sheet.

Referencing

All sources of information used in the preparation of your submission must be acknowledged using the Harvard system of referencing. This includes all print, video, electronic sources.

Phrases, sentences or paragraphs taken verbatim from a source must be in quotation marks and the source(s) cited using both **in-text** referencing and a **reference list**.

Plagiarism is the failure to acknowledge sources of information. You should be fully aware of the meaning of plagiarism and its consequences both to your marks, position at the university and criminal liability. The plagiarism in your assignment submissions can be assessed both in hard copy and in soft copy through software such as Turnitin.

The UTS Library and UTS HELPS (web links above) provide extensive information for students on referencing correctly to support you in avoiding plagiarism.

TECHNICAL SUPPORT DOCUMENTATION

TABLE OF CONTENTS

INTRODUCTION

This report will outline key details for the eLibrary Management System (herein ELMS) project and the development of the ELMS application. It will also provide recommendations for the future of the project as it is handed over to the new team. The release of this document coincides with a demonstration video and the first release version of the application, both of which can be found on GitHub. Links have been included throughout this document where relevant; although, they have also been included in their respective sections in the appendix.

Use Cases

Narratives

The following narratives outline the minimum functionality of the ELMS application. All the following narratives have been fully implemented at the time of release.

Student (S)

Use Case ID Number	**UCS1.1**
Name of Use Case Narrative	Login to Account
User Story/Stories	As a student, I want to be able to create, login to and view my account, so that I can keep track of my library record and book rentals As a staff member, I want to be able to create and login to my account, so that I can keep track of my library record and book rentals As an admin, I want to have access to an administrative account that has administrative privileges, so that I can maintain the library system and update the catalogue
Goal	Allow the user to login (or register so that they can login) to their account
Pre-conditions	None
Post-conditions	Users can now access the functions of the ELMS as they are logged into their account
Actor(s)	User (Any Permission Level)
Trigger	User has opened the ELMS application
Task and Response to actor	The user will load up the eLibrary Management System application on their device. The system will display the prompt to the user to input the necessary details to access the system (university email ids and password). After the login is successful, the system will navigate the user to the rest of the application, where they can use the functionalities, they require (for example, searching for a book, viewing their user profile, or adding a book into the database).
Primary Flow	1. The user opens the ELMS application either on mobile or on the web 2. The user is prompted to input their account details into the system's login fields 3. If login is successful, the user now enters the system with their corresponding levels of access (for example, administration staff have access to the administration and management of the system and database) a. If unsuccessful due to a student or staff account not previously registered with the system, please see Alternate Flow 1
Exceptions	**Exception 1 (steps 2 to 3):** If the user does not currently have an account registered, the login attempt will fail as it is not recognised

Alternate Flow 1	Staff or student is required to register before they can log in
Trigger	Staff or student attempts to login but does not have a previously registered account
Steps	1. Step 1 to 2 is the same as the primary flow

	2. As the user attempts to log in, they won't be granted access as no account has been registered 3. The user will navigate to a sign-up form where they will be required to input any necessary details 4. Re-joins with step 2 of the primary flow
Post-conditions	The user gains the ability to log into the system and gains access to the system's functionality

Use Case ID Number	**UCS1.2**
Name of Use Case Narrative	View Account
User Story/Stories	As a student, I want to be able to create, login to and view my account, so that I can keep track of my library record and book rentals As a staff member, I want to be able to create and login to my account, so that I can keep track of my library record and book rentals As a student, I want to be able to see all books that have been assigned to me by my lecturers, so that I can complete any necessary reading before class
Goal	Allow the user to view their account and details regarding borrowings
Pre-conditions	User is logged in
Post-conditions	Can view details regarding book rentals, library records, and outstanding fines
Actor(s)	User (Any Permission Level)
Trigger	User has accessed the "View Account" window
Task and Response to actor	If the user is required to check what books they have borrowed, book return dates, or any outstanding fines, they can find this information from their user profile. Once their user profile is open, the library management system loads in the required information to be displayed. This information entails currently borrowed books and their corresponding return date, previously borrowed books, fines given (if any), and see if a book has an ability for renewal.
Primary Flow	1. The user will log into their account to access the library management system 2. The user will navigate to the tab where they can access their library profile 3. Once in their profile, the user can view different aspects: a. Currently borrowed book titles, with details regarding their return date, if renewals are allowed, and potential fines if the book is not returned by the return date b. Any current outstanding fines incurred from not returning a book on time c. Previously borrowed books, including borrowed and returned dates for the most recent copy, and the number of times the user has borrowed the same book d. Books that have been prescribed by a student's teacher, with the teacher's name also visible so that the student can determine which class it is for

Includes/Inherits/ Extends	**Extends:** View Prescriptions (UCS4)

Use Case ID Number	**UCS2**
Name of Use Case Narrative	Search Library
User Story/Stories	As a student, I want to be able to search through the catalogue and refine my search with multiple parameters, so that I can find the books I am looking for more easily
Goal	Allow a user to search the library database to find the desired book
Pre-conditions	Login to Library Management System
Post-conditions	Book is found in the search and then can be borrowed or previewed
Actor(s)	User (Any Permission Level)
Trigger	User clicks the "View eLibrary" button to navigate to the search window
Task and Response to actor	Users will log into the system using their account details and use the available search function to look for the book they require. Users can also use parameters to search for the book they require more easily. The system will take the information the user input and return books that match their search criteria. If the user finds the books they need, they will have the ability to preview and/or borrow that specific book (if copies are available).
Primary Flow	1. The user will navigate to the designated area to search for a book they require 2. Users can create a more advanced search by adding parameters in the search criteria (For example, category, name, and author) 3. The list of books found by the system will display, and the user locates the one they require 4. Once the book is found, the user can borrow the book. More details regarding borrowing can be found in UCS3.
Exceptions	**Exception 1 (step 3 to 4):** No applicable books match the search criteria **Exception 2 (step 4):** There are no currently available copies of the searched book

Use Case ID Number	**UCS3**
Name of Use Case Narrative	Borrowing of Books
User Story/Stories	As a student, I want to be able to borrow digital and reserve physical copies of books, so that I can spend more time reading and less time travelling to the library
Goal	To successfully borrow a book from the library's database
Pre-conditions	User will need to be in the search window to access the borrow function
Post-conditions	Borrowed book successfully added to the user's rented books list
Actor(s)	User (Any Permission Level)
Trigger	User selects a book in the search window
Primary Flow	1. User accesses the search window and looks for the book they require. Please see UCS2 for more information regarding searching through the system.

	2. Once the user finds the book they wish to borrow and have selected the book, the button to borrow will become enabled. 3. Once the user clicks the "Borrow" button, the selected book will be added to their profile under rented books and a success message will display 4. The stock for the borrowed book will decrease an update accordingly in the search
Exceptions	**Exception 1 (Step 3 to Step 4):** Borrow fails due to the selected book having no stock available

Use Case ID Number	**UCS4**
Name of Use Case Narrative	View Prescriptions
User Story/Stories	As a student, I want to be able to see all books that have been assigned to me by my lecturers, so that I can complete any necessary reading before class
Goal	The student should be able to view the books prescribed
Pre-conditions	The student has an ELMS account and can log in
Post-conditions	The student can view the prescribed books
Actor(s)	Student
Trigger	The student opens the "View Account" menu
Task and Response to actor	The list of prescribed books
Primary Flow	1. The student selects 'View Account' on the main menu 2. The student selects 'Prescribed Books' in the list of toggles on the left of the view 3. The student can view their prescribed books
Includes/Inherits/ Extends	**Extends:** View Account (UCS1.2)

Use Case ID Number	**UCS5**
Name of Use Case Narrative	Renewal of Books
User Story/Stories	As a student, I want to be able to renew a borrowed book, so that I do not incur a fine when passing the return date.
Goal	To renew a book
Pre-conditions	Actor has logged into the ELMS and had access to their account profile
Post-conditions	Selected book's return date is update
Actor(s)	User (Any Permission Level)
Trigger	User selects a book in their profile under the "rented books" tab
Task and Response to actor	Once user has a book selected in their rented books list, the option to renew the book will become enabled. Once the renew button is clicked, if the book isn't fined, the due date successfully updates to a new date two weeks from the current return date.

Primary Flow	1. In the account profile, navigates to the "rented books" button to display all the books they have currently borrowed. Please see UCS3 for more information regarding the borrowing of books. 2. User can select one of the books they have rented and proceed to renew the book. 3. When the renew book button is clicked, the due date for the selected book is changed to a date two weeks following the original return date. 4. The user can repeat steps 2 to 3 as required for the number of books they wish to renew
Exceptions	**Exception (Step 2 to Step 3):** The selected book is overdue and a fine is pending payment; the book cannot be renewed
Includes/Inherits/ Extends	**Extends:** Borrow Book (UCS3)

Alternate Flow 1	Book currently selected is fined
Trigger	The book is attempted to be renewed, however it's currently fined
Steps	1. Follows steps 1-2 of the primary flow 2. As the user attempts to renew the selected book, an error message will appear, alerting the user that the book is currently fined 3. User will navigate to the "fines" tab and resolve the outstanding fines before attempting another renewal 4. Book is renewed automatically upon payment of the fine
Post-conditions	Book successfully renewed after fined status is removed

Use Case ID Number	**UCS6**
Name of Use Case Narrative	Interface Response Time
User Story/Stories	As a student, I want my searches and book previews to load within three seconds, so that I can organise my borrowing quickly and continue with my classwork
Goal	The search and book previews load within 3 seconds.
Pre-conditions	None
Post-conditions	None
Actor(s)	User (Any Permission Level)
Trigger	The user interacts with the application
Primary Flow	1. The user logs in to the system 2. The system should react to interaction, whether that be searches or button clicks, within three seconds 3. All elements and data should be fully loaded within the time period
Includes/Inherits/ Extends	**Extends:** All functionality

Staff (T)

Use Case ID Number	**UCT1**
Name of Use Case Narrative	Staff Account
User Story/Stories	As a staff member, I want to be able to create and log into my account, so that I can keep track of my library record and book rentals
Goal	Track of library record and book rentals
Pre-conditions	Staff has an ELMS account and can view books on the system
Post-conditions	Staff can keep track of the library record and book rentals
Actor(s)	Staff
Trigger	None
Primary Flow	Follows flow of UCS1

Use Case ID Number	**UCT2**
Name of Use Case Narrative	Staff Borrow and Browse Books
User Story/Stories	As a staff member, I want my account to have all the privileges and access options as a student account, so that I can also browse and borrow books
Goal	Staff being able to browse and borrow books
Pre-conditions	Staff has an ELMS account
Post-conditions	Staff can browse and borrow books
Actor(s)	Staff
Task and Response to actor	System shows the menu after the staff has logged into the ELMS
Primary Flow	1. Automated by system – no steps

Use Case ID Number	**UCT3**
Name of Use Case Narrative	Request Book
User Story/Stories	As a staff member, I want to be able to place a request for a new book to be added to the library system, so that my students have access to the textbooks I want to teach with
Goal	Let Staff account make a request for a new book
Pre-conditions	Actor has logged into the ELMS as a staff
Post-conditions	Add book request been sent to Admin
Actor(s)	Staff
Trigger	User clicks on the “Staff Options” button
Task and Response to actor	Staff member completes the request book form, and after submission of the form, a message will display stating “The book has been requested. You can now close this window.”. The requested book is then successfully sent to an administrator for approval.
Primary Flow	1. Open “Staff Option” button led to a new window 2. By click “request a book” button a pop-up window will be appear 3. Enter the information of the book user want to request, and click “request book”

	4. The user can repeat step 2-3 to request more than one book

Use Case ID Number	**UCT4**
Name of Use Case Narrative	Prescribing Books to Students
User Story/Stories	As a staff member, I want to be able to prescribe books to my students, so that they can have access to required readings and be informed for my classes
Goal	Staff being able to prescribe books to students
Pre-conditions	Staff has an ELMS account
Post-conditions	Staff can prescribe books to students
Actor(s)	Staff, Student
Task and Response to actor	User logs into the Library and logs in as staff in the menu option. The staff then choose the book they will be use in class to prescribing.
Primary Flow	1. Staff member logs into ELMS 2. Staff member navigates to library menu 3. Staff member selects the book they would like to prescribe 4. Staff member clicks the 'prescribe' button 5. Staff member selects the student they would like to prescribe the book to a. They can also refine the table of students by using the search text field and entering either a name or ID 6. The staff member clicks 'prescribe'

Admin (A)

Use Case ID Number	**UCA1.1**
Name of Use Case Narrative	Admin Account
User Story/Stories	As an admin, I want to have access to an administrative account that has administrative privileges, so that I can maintain the library system and update the catalogue
Goal	Admin to have all admin privileges so the library system can be maintained
Pre-conditions	Admin account exists in the database
Post-conditions	None
Actor(s)	Administrator
Trigger	None
Primary Flow	Flow follows UCT1. Admin should be able to view all menus due to their elevated permission level

Use Case ID Number	**UCA1.2**
Name of Use Case Narrative	Adding a new book in the database

User Story/Stories	As an admin, I want to have access to an administrative account that has administrative privileges, so that I can maintain the library system and update the catalogue
Goal	A new book is added to the library
Pre-conditions	Admin has logged into the ELMS
Post-conditions	A new book object is added to the relevant lists and database
Actor(s)	Admin
Trigger	User clicks on the "Add a Book" button in the admin menu
Primary Flow	1. In Administrative option show a "Add a new book" button to open Add book window 2. User can enter the book details and click "Add book" to add the book in library database 3. After the addition of the book is successful, click "close" to close the window 4. User can repeat step 1-3 to add multiple books
Exceptions	**Exception (Step 2 to Step 3):** If the book is already in the library database, it will show an error message "A book has already been added to the library with this ID"

Use Case ID Number	**UCA1.3**
Name of Use Case Narrative	Modifying an existing book
User Story/Stories	As an admin, I want to have access to an administrative account that has administrative privileges, so that I can maintain the library system and update the catalogue
Goal	Modify or remove a book from the library database
Pre-conditions	Admin has logged into the ELMS
Post-conditions	Book object is updated or deleted
Actor(s)	Admin
Trigger	User clicks on the "Modify an Existing Book" button in the admin menu
Task and Response to actor	The administrator can view all the books available in the library database. Once a book is selected, the user has two options for modifications: either to remove the book currently selected, or to modify the details of the book selected. If the user chooses to modify the details of the book, a form to update the book will appear containing all the current details in contains.
Primary Flow	1. User opens the 'Modify an Existing Book' menu, displaying a list of all books available in the database 2. Once the user finds the book they wish to modify and selects it, the two options to modify the book will appear—either to remove the book, or modify the books current details 3. If the user wishes to remove the book, the "Remove Book" button is clicked, and it's removed from the entire database. 4. If the user wishes to update the details of a book, the "Modify Book" button is pressed displaying a form pre-filled with the books data

	5. The user can alter the details of the selected book and press "update book" once done.
Exceptions	**Exception (Step 4 to Step 5):** If the user cancels the form with the updates they've made, the changes will be lost.

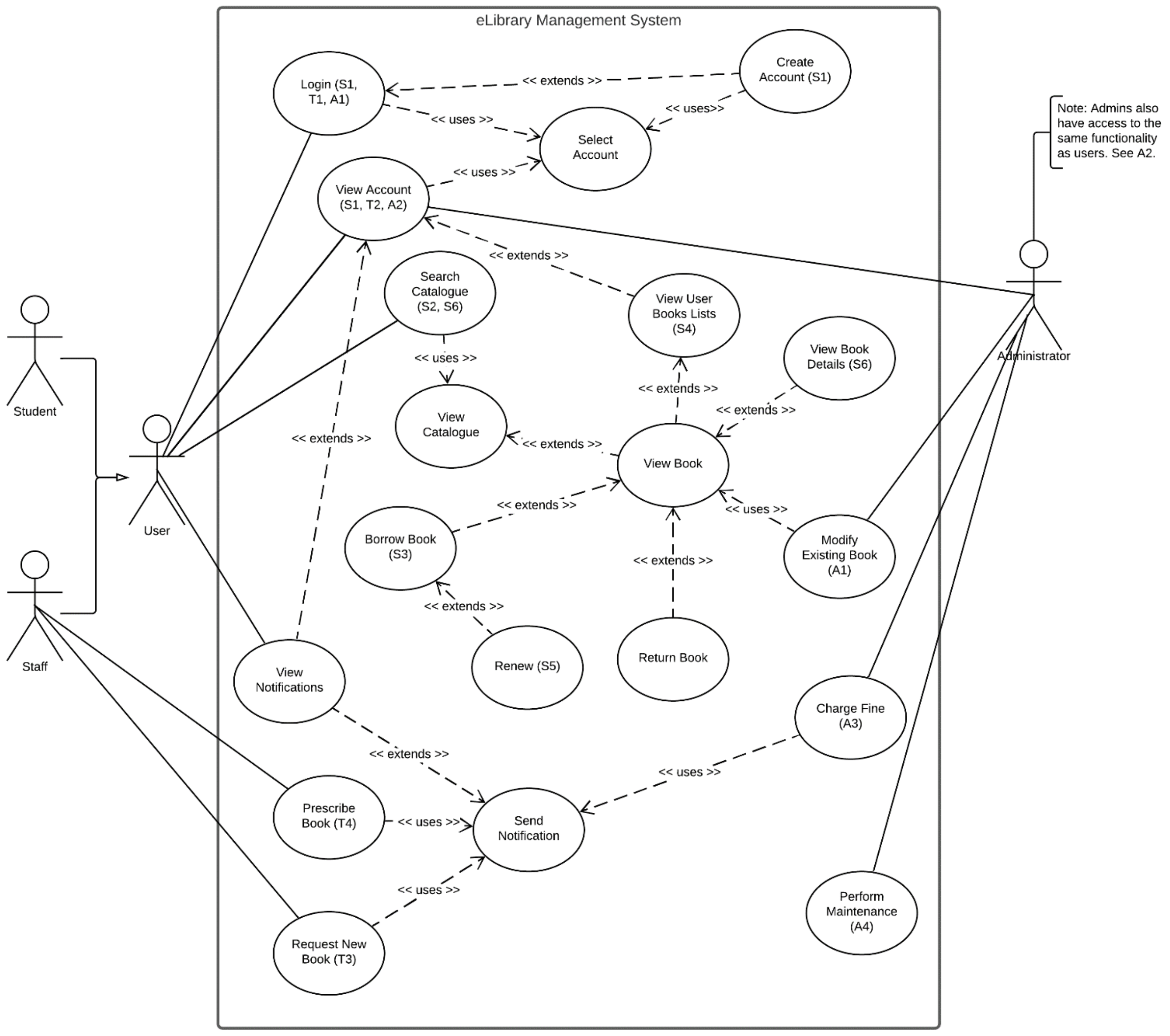
Use Case ID Number	**UCA2**
Name of Use Case Narrative	Admin Account Privileges
User Story/Stories	As an admin, I want my account to have all the privileges and access options as a staff account, so that I can test and ensure the system is operating normally
Goal	Admin account should have access to all admin functionality and all functionality of other user permission levels
Pre-conditions	Admin account exists
Post-conditions	None
Actor(s)	Admin
Trigger	None
Primary Flow	1. Automated by system – no steps

Use Case ID Number	**UCA3**
Name of Use Case Narrative	Fine calculation
User Story/Stories	As an admin, I want fines to be automatically calculated when a book passes its return date, so that a notification can be sent automatically to the offender and so that return dates and enforced diligently
Goal	Automatically calculate the fine for later return
Pre-conditions	User has borrowed a book and hold for more than 2 weeks without renew it
Post-conditions	Users need to pay fine before they can return the book
Actor(s)	Student, Staff
Trigger	Currently rented book has passed its return by date
Task and Response to actor	Need to be pay the fine of later return
Primary Flow	1. User has a book been borrow for more than 2 weeks without renew 2. In the view account window, the user will not be able to return the book, it will display an error message 3. The "fine" button now is able for user to click and refresh the window to show the book's fine detail 4. User can use "Pay fine" button to pay the fine 5. When fine is paid, the book will be removed from the fined books table 6. The book is automatically renewed so that the user can elect to return the book or continue renting it
Includes/Inherits/ Extends	**Extends:** Borrow books, Return books

Use Case ID Number	**UCA4**
Name of Use Case Narrative	Maintenance Timing
User Story/Stories	As an admin, I want a maintenance period from 02:00 till 06:00 on Sundays, so that I can ensure any necessary maintenance can be carried out at a time when the users will be aware and notified in advance
Goal	Users are aware and notified of maintenance timing and locked out of the application
Pre-conditions	Admin is able to log in
Post-conditions	Admin can begin maintenance knowing that no users can log in
Actor(s)	Primary: Admin Secondary: Staff and Students
Trigger	Admin toggling on maintenance
Primary Flow	1. Admin logs into the ELMS 2. Admin goes to administrative menu 3. Admin clicks on the enable maintenance button 4. The login view displays that the maintenance period is in effect and disables the login and register buttons 5. Admin begins maintenance 6. Admin begins at applicable step and follows the same process to disable the maintenance period

Diagram

Like the narratives, this use case diagram outlines the minimum functionality of the application. It also includes requested features that are expected to be implemented by the take-over team (e.g., notification handling) but were outside our scope (outlined in the initial documentation).



User Stories and Acceptance Tests

The following section outlines the user stories established during sprint 0 and their respective acceptance tests. They have been separated according to the user in question.

Student (S)

1. As a student, I want to be able to create, login to and view my account, so that I can keep track of my library record and book rentals
    a. The application should be accessible by the user (student)
    b. The application should have an interface
    c. The interface should allow user input
    d. The inputs should be in a form that can be passed securely to be checked against accounts in the database
    e. The system provides an error message if registration is attempted more than one for the same id
    f. Once logged in, the user should be able to view a list of their books and library records
    g. The interface should have a button at the bottom to exit the application
2. As a student, I want to be able to search through the catalogue and refine my search with multiple parameters, so that I can find the books I am looking for more easily
    a. The application should have an interface
    b. The interface should have a button that allows the user to navigate to the search page
    c. Once on the search page, the user should be able to enter keywords into a field or select from a list of categories
    d. After searching or selecting a category, the list of available books should populate with relevant results
3. As a student, I want to be able to borrow digital and reserve physical copies of books, so that I can spend more time reading and less time travelling to the library
    a. The application should satisfy all the criteria of S2
    b. Individual entries in the list should be selectable
    c. When a student selects a book from the list, two buttons should become active which allow the user to preview or borrow the book
    d. Clicking the preview button should open a new window which will show a preview of the book
    e. Clicking the borrow button should prompt the user to choose a download location, if it is a digital book, or notify the user that their book will be ready to be picked up within a given timeframe if it is a physical copy
    f. Clicking the borrow button should decrement the stock of an available book
4. As a student, I want to be able to see all books that have been assigned to me by my lecturers, so that I can complete any necessary reading before class
    a. The application should satisfy all the criteria of S1
    b. When viewing their account, users should be able to click a button to display their prescribed texts
    c. The application should update the book list to show only prescribed books
5. As a student, I want to be able to renew a borrowed book, so that I do not incur a fine when passing the return date
    a. The application should satisfy all the criteria of S1
    b. When viewing the list of rented books, the user should be able to select individual entries
    c. When an entry is selected, the user should be able to click the renew button, permitting that they have renewals available
    d. The application should update the renewal date following the books rental period

6. As a student, I want my searches and book previews to load within three seconds, so that I can organise my borrowing quickly and continue with my classwork
    a. The application should satisfy all the criteria of S2
    b. The application should display all previews and update/populate lists in under 3 seconds
7. As a student, I want to be able to access the library system at any time of the day, on any day of the week, so that I can borrow and return books when I find it most convenient
    a. The application should be always accessible

Staff (T)

1. As a staff member, I want to be able to create and log into my account, so that I can keep track of my library record and book rentals
    a. The application should satisfy all the criteria of S1
    b. The application should recognise a difference in account permission level
2. As a staff member, I want my account to have all the privileges and access options as a student account, so that I can also browse and borrow books
    a. The application should satisfy all the criteria of S1-S7 while being applicable for a staff account
3. As a staff member, I want to be able to place a request for a new book to be added to the library system, so that my students have access to the textbooks I want to teach with
    a. The application should satisfy all the criteria of S3
    b. The application should have a third button that is visible only by staff
    c. The application should, when this button has been clicked, display an interface with labelled text fields
    d. The user should be able to enter the applicable information for the requested text into the fields
    e. The user should be able to click a button the submit the request when they have finished entering details
    f. The application should store the details of the request
4. As a staff member, I want to be able to prescribe books to my students, so that they can have access to required readings and be informed for my classes
    a. The application should satisfy all the criteria of S3
    b. The application should have a fourth button that is visible only by staff
    c. The application should, when this button has been clicked, allow the staff member to input a student ID to prescribe the book to
    d. The application should satisfy all the criteria of S4

Admin (A)

1. As an admin, I want to have access to an administrative account that has administrative privileges, so that I can maintain the library system and update the catalogue
    a. The application should recognise a difference in account permission level
    b. The application should have an interface for administrative controls
    c. This interface should have buttons to control the catalogue
    d. Clicking each button should create a window that allows the administrator to control the catalogue
        i) 'Add' should allow the administrator to enter the details of a new book
        ii) 'Remove' should allow the administrator to remove a book from the list of all books – this should not be available for books currently being rented
        iii) 'Modify' should allow the administrator to select a book from the list of all books. This will then present a window identical to add; however, the fields will be pre-populated with the data of the selected book

   e. When the user is finished with the interface, there should be a button at the bottom the save the changes and return to the previous window
2. As an admin, I want my account to have all the privileges and access options as a staff account, so that I can test and ensure the system is operating normally
   a. The application should satisfy all the criteria of T1 while being applicable for an admin account
3. As an admin, I want fines to be automatically calculated when a book passes its return date, so that a notification can be sent automatically to the offender and so that return dates and enforced diligently
   a. At the beginning of each day, the application should check the list of borrowed books for all users
   b. The application should be able to view the return date for a book
   c. The application should be able to recognise if a fine has already been incurred for the given book
   d. The application should be able to view the fine value for the book
   e. The application should be able to update the user's account who is currently borrowing the book to reflect the new fine if a book they are renting has exceeded the return date
4. As an admin, I want a maintenance period from 02:00 till 06:00 on Sundays, so that I can ensure any necessary maintenance can be carried out at a time when the users will be aware and notified in advance
   a. The application should satisfy all the criteria of A1
   b. The application should have a toggle to enable and disable the maintenance
   c. The application should be able to disable the login interface during maintenance
   d. The application should be able to display a message on the login interface to notify the other users of the maintenance period

Non-Passing Results

As demonstrated in the demonstration video, all the acceptance tests have been passed as of the time of release. It could be argued that S7 did not pass by criteria 'a' due to the conflict between it and A4; however, we believe that S7 must be taken in a 'within reason' context and, as it is not unreasonable for the administrators to request a maintenance period, S7 has been deemed to have passed. It should also be noted that S3 could be interpreted as not passing due to criterion 'e'; currently, clicking the borrow button notifies the user that the book has been borrowed as the ELMS currently assumes it only contains digital books – it does not provide details for physical books, nor any pickup instructions – however, as we do not have access to the library database, we cannot provide downloads for copies of the texts. It is presumed this functionality will be implemented when the database is connected by the take-over team.

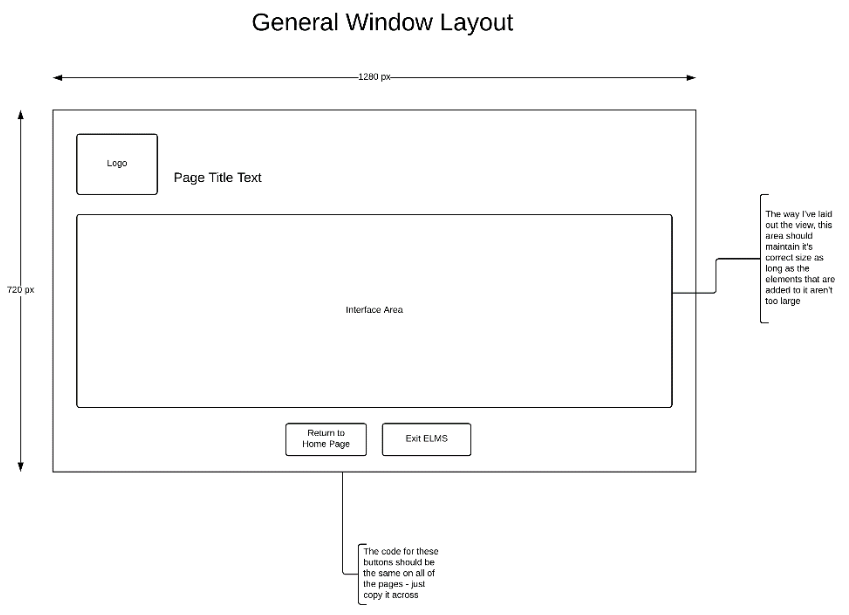
Design Planning

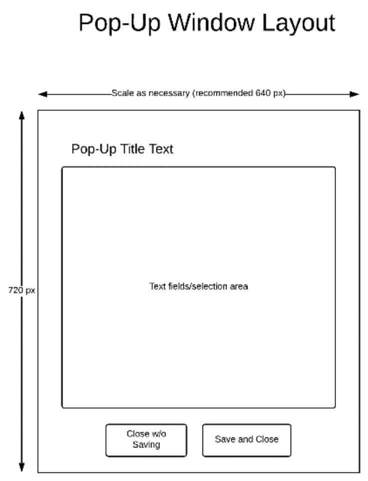
Design Principles

At the outset of developing the ELMS application, design principles and wireframes were created to ensure each UI view was consistent across the application. Each window of the UI followed the general layout developed early in the project timeline. A style sheet was also implemented to create a consistent colour scheme and font styles throughout each of the sections of the library system. In turn, keeping the general layout of the interface makes the application a lot more user friendly and optimises the user's work efficiency whilst using all the functionalities of the application. This also minimises the time it would take to learn all the functionality available through the UI. The general layout and key views of the ELMS can be seen in the following wireframes.

Wireframes

General Layout

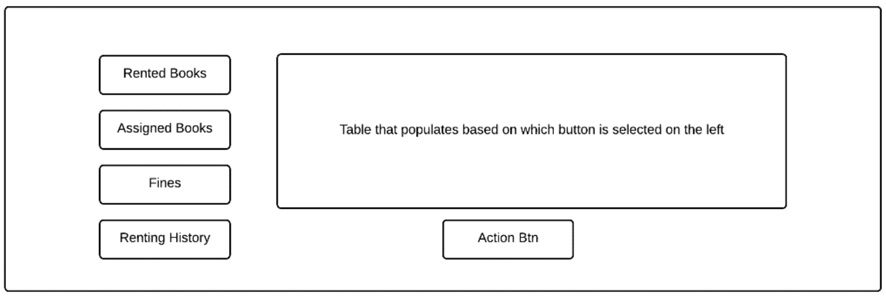

General Window Layout
1280 px
720 px
Logo
Page Title Text
Interface Area
Return to Home Page
Exit ELMS
The way I've laid out the view, this area should maintain it's correct size as long as the elements that are added to it aren't too large
The code for these buttons should be the same on all of the pages - just copy it across



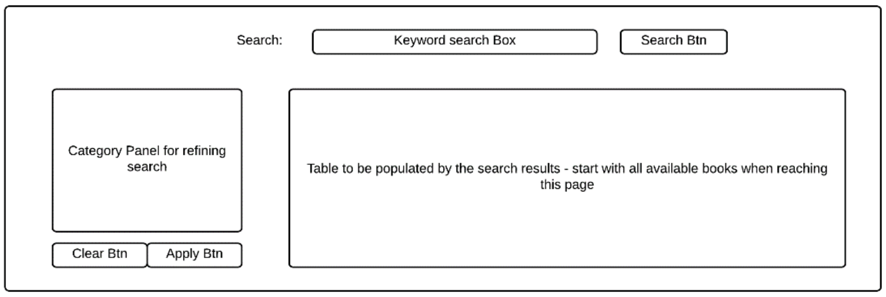
Key Views

The key views represent the 'interface area' in the 'General Window Layout'. In the current build, most of the views match the preliminary layout designs. The main exception is the library search, as some buttons have been removed because they were unnecessary, and radio buttons have been added for simplicity for the user in browsing the available options or sub-sections within a given category. The other views were left at the discretion of the team member/s assigned to complete it, with the design language to be inferred from the other views – though, it was largely dictated by the stylesheet.

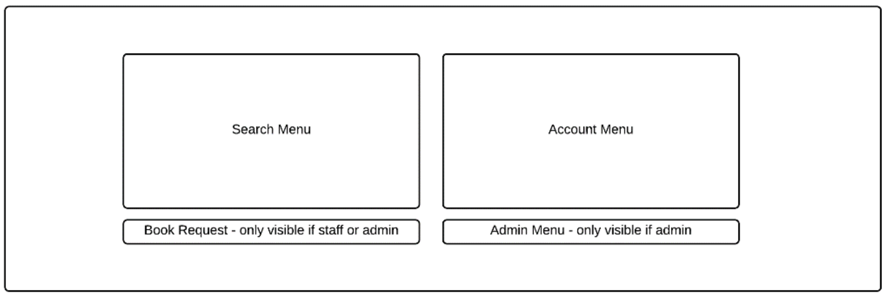
Account View Layout


Rented Books
Assigned Books
Fines
Renting History
Table that populates based on which button is selected on the left
Action Btn

Library Search Layout


Search:
Keyword search Box
Search Btn
Category Panel for refining search
Clear Btn
Apply Btn
Table to be populated by the search results - start with all available books when reaching this page

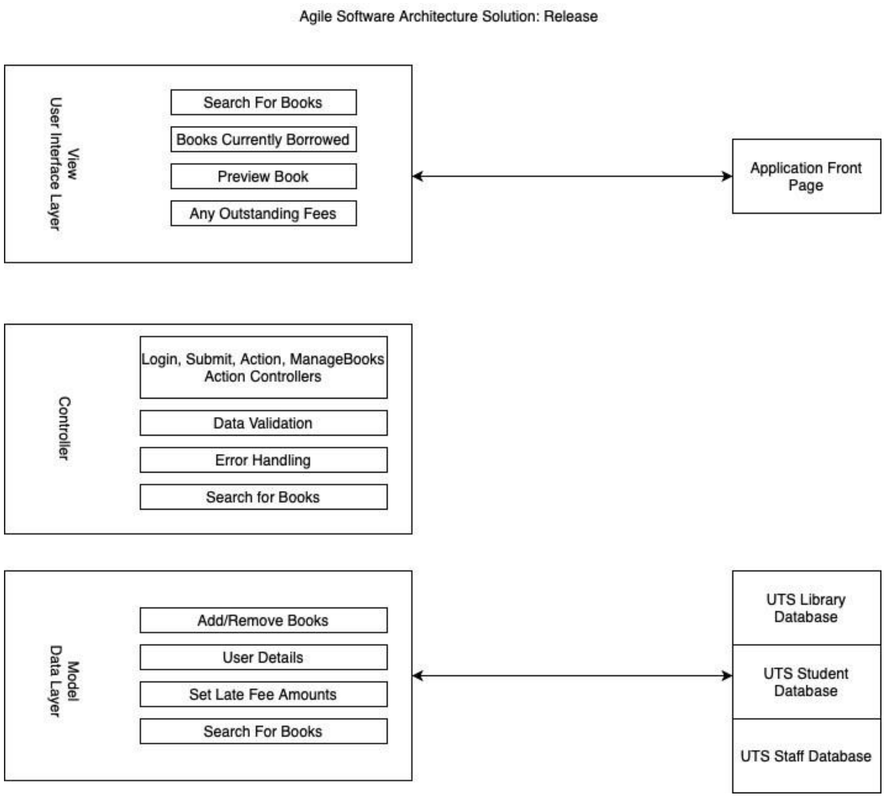
Home Page


Search Menu
Account Menu
Book Request - only visible if staff or admin
Admin Menu - only visible if admin

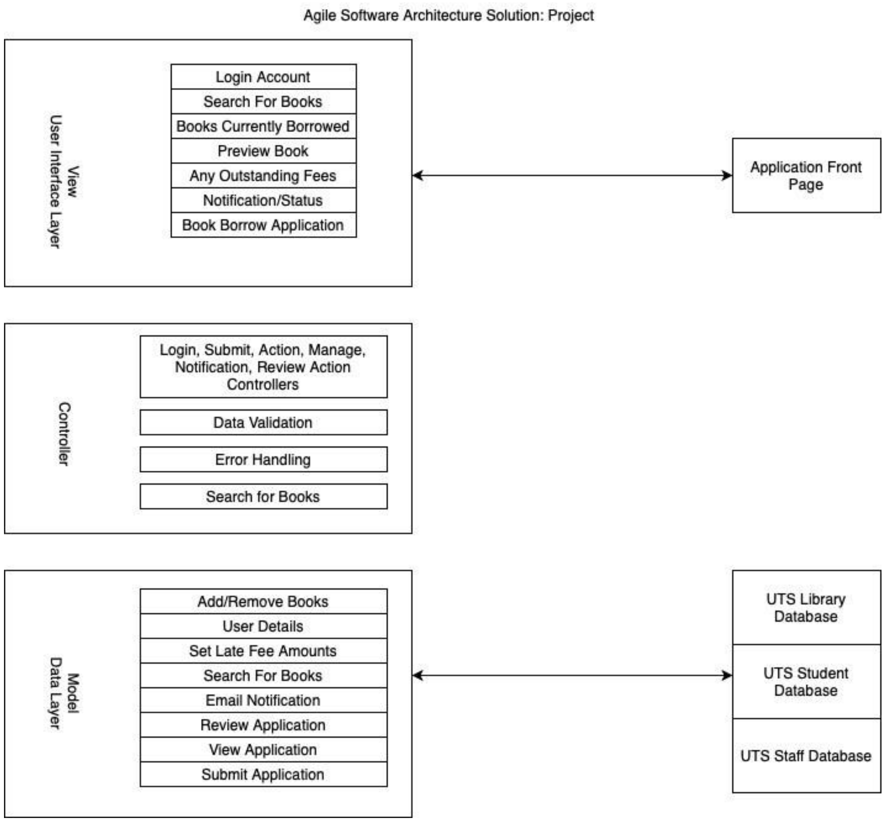
STRUCTURE

ARCHITECTURE DIAGRAMS

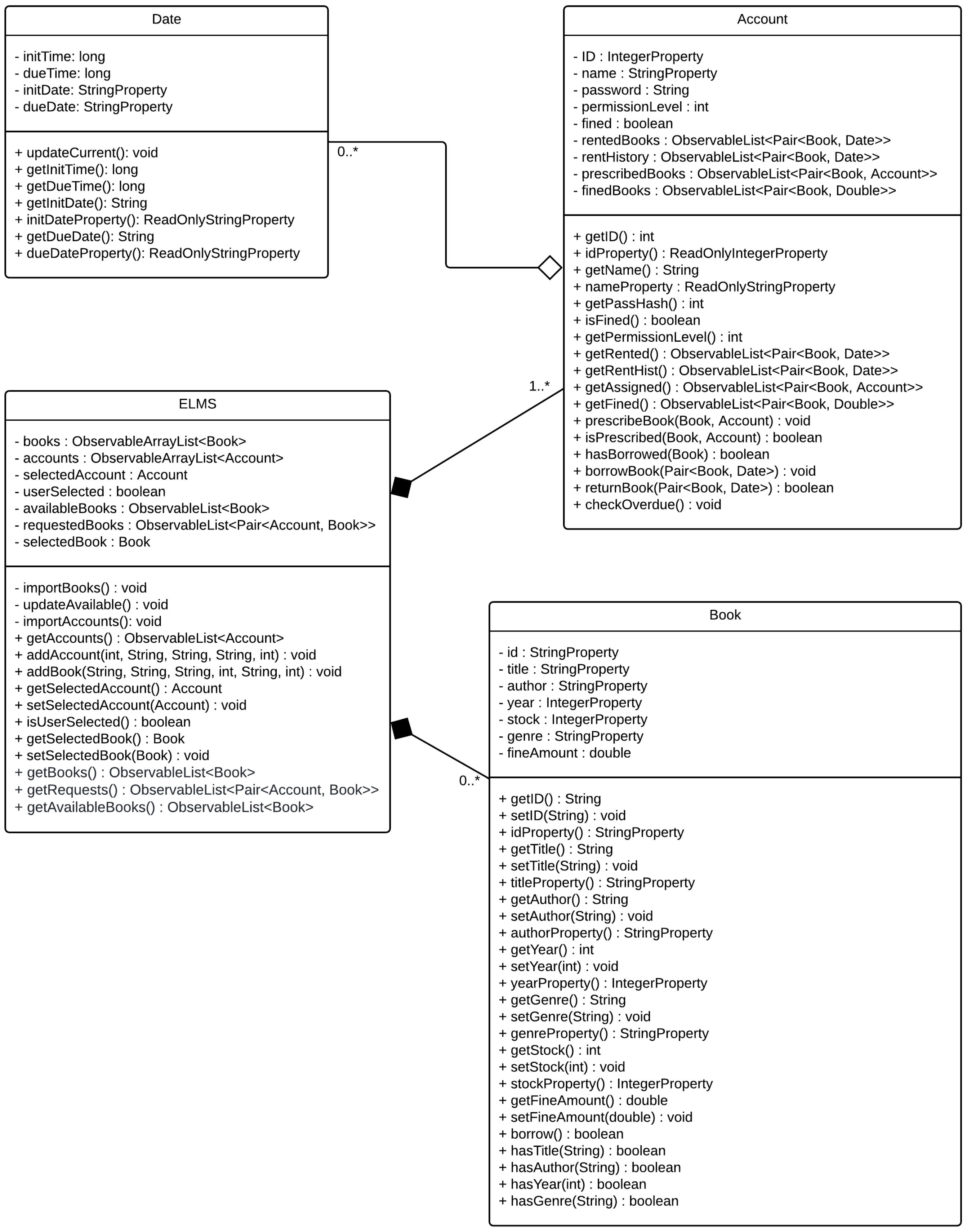
As all the project's initially outlined goals have been completed, the following architecture diagrams that have been included are unchanged from the preparation report.



The following diagram outlines the proposed architecture after the completion of the project by the team taking over this project. The most notable addition is the inclusion of a notification system.



Class Diagram


Date
- initTime: long
- dueTime: long
- initDate: StringProperty
- dueDate: StringProperty
+ updateCurrent(): void
+ getInitTime(): long
+ getDueTime(): long
+ getInitDate(): String
+ initDateProperty(): ReadOnlyStringProperty
+ getDueDate(): String
+ dueDateProperty(): ReadOnlyStringProperty
0..*
Account
- ID : IntegerProperty
- name : StringProperty
- password : String
- permissionLevel : int
- fined : boolean
- rentedBooks : ObservableList<Pair<Book, Date>>
- rentHistory : ObservableList<Pair<Book, Date>>
- prescribedBooks : ObservableList<Pair<Book, Account>>
- finedBooks : ObservableList<Pair<Book, Double>>
+ getID() : int
+ idProperty() : ReadOnlyIntegerProperty
+ getName() : String
+ nameProperty : ReadOnlyStringProperty
+ getPassHash() : int
+ isFined() : boolean
+ getPermissionLevel() : int
+ getRented() : ObservableList<Pair<Book, Date>>
+ getRentHist() : ObservableList<Pair<Book, Date>>
+ getAssigned() : ObservableList<Pair<Book, Account>>
+ getFined() : ObservableList<Pair<Book, Double>>
+ prescribeBook(Book, Account) : void
+ isPrescribed(Book, Account) : boolean
+ hasBorrowed(Book) : boolean
+ borrowBook(Pair<Book, Date>) : void
+ returnBook(Pair<Book, Date>) : boolean
+ checkOverdue() : void
1..*
ELMS
- books : ObservableArrayList<Book>
- accounts : ObservableArrayList<Account>
- selectedAccount : Account
- userSelected : boolean
- availableBooks : ObservableList<Book>
- requestedBooks : ObservableList<Pair<Account, Book>>
- selectedBook : Book
- importBooks() : void
- updateAvailable() : void
- importAccounts(): void
+ getAccounts() : ObservableList<Account>
+ addAccount(int, String, String, String, int) : void
+ addBook(String, String, String, int, String, int) : void
+ getSelectedAccount() : Account
+ setSelectedAccount(Account) : void
+ isUserSelected() : boolean
+ getSelectedBook() : Book
+ setSelectedBook(Book) : void
+ getBooks() : ObservableList<Book>
+ getRequests() : ObservableList<Pair<Account, Book>>
+ getAvailableBooks() : ObservableList<Book>
0..*
Book
- id : StringProperty
- title : StringProperty
- author : StringProperty
- year : IntegerProperty
- stock : IntegerProperty
- genre : StringProperty
- fineAmount : double
+ getID() : String
+ setID(String) : void
+ idProperty() : StringProperty
+ getTitle() : String
+ setTitle(String) : void
+ titleProperty() : StringProperty
+ getAuthor() : String
+ setAuthor(String) : void
+ authorProperty() : StringProperty
+ getYear() : int
+ setYear(int) : void
+ yearProperty() : IntegerProperty
+ getGenre() : String
+ setGenre(String) : void
+ genreProperty() : StringProperty
+ getStock() : int
+ setStock(int) : void
+ stockProperty() : IntegerProperty
+ getFineAmount() : double
+ setFineAmount(double) : void
+ borrow() : boolean
+ hasTitle(String) : boolean
+ hasAuthor(String) : boolean
+ hasYear(int) : boolean
+ hasGenre(String) : boolean

Defects and Traceability

Defect Management

Defect tracking for this project was conducted using the Trello board; any defects found were marked with the 'Bug' label. Any defects that were found and handled immediately during the sprints were not logged and, consequently, there are very few bug cards on the Trello. There were three defects that were logged; however, only one related specifically to single use case. This is an outlier as, as alluded to, the defects discovered that related to a single use case were typically resolved immediately. Hence there are more general defects than those relating to a specific use case.

Defect Summary and Evaluation

Of the two non-specific defects that were logged during the sprints, one was functional and the other aesthetic. The former, being a functional defect, was prioritised more highly than the aesthetic defect and, consequently, has been resolved. At the time of release, the aesthetic defect has yet to be resolved. The case-specific defect, being functional and therefore given high priority, has also been resolved. The team believes that, while we did not discover many defects, we have handled those that have been found effectively, ensuring they were tracked in Trello with their key details.

TRACEABILITY

Given the brevity of the previous section, this section will also be. All the user stories have been completed, with all requested functionality that was within the scope fully implemented into the ELMS application. As can be seen in the video, the sections that have been completed follow and abide by all the respective acceptance tests. As all the testing was conducted manually, we do not have a concrete list that reflects all the testing that has been conducted. Nonetheless, a table has been included below, which outlines some of the known defects discovered during development. Again, many of these were resolved immediately and were, therefore, not logged in Trello.

USER STORY	ACCEPTANCE CRITERIA	DEFECTS
S3	Criterion A. The application should satisfy all the criteria of S2 Criterion B. Individual entries in the list should be selectable Criterion C. When a student selects a book from the list, two buttons should become active which allow the user to preview or borrow the book Criterion D. Clicking the preview button should open a new window which will show a preview of the book Criterion E. Clicking the borrow button should prompt the user to choose a download location, if it is a digital book, or notify the user that their book will be ready to be picked up within a given timeframe if it is a physical copy Criterion F. Clicking the borrow button should decrement the stock of an available book	Criterion A. Criteria A: No defects; All criteria are satisfied Criterion B. Criteria B: No defects Criterion C. Criteria C: No defects; When a book is selected from the table, the disabled prescribe and borrow buttons become enabled, which the user can select to perform further actions Criterion D. Criteria D: No defects Criterion E. Criteria E: Possible defect present; Outlined in detail in the 'Non-Passing Results' subsection Criterion F. Criteria F: No defects
S4	Criterion A. The application should satisfy all the criteria of S1 Criterion B. When viewing their account, users should be able to click a button to display their prescribed texts Criterion C. The application should update the book list to show only prescribed books	Criterion A. No defects Criterion B. One defect found during unit testing. Button presses would lock after action and not allow for change in table 'view'. Resolved by fixing order of actions in method. Criterion C. Affected by previous defect; otherwise, no defects
S1/A1	N.B. As this defect affected multiple user stories, no specific criterion is listed	One defect found during unit testing. Text field binding was not functioning as intended if the field was cleared after having details entered. Added checks during button press to resolve.

S5	Criterion A. The application should satisfy all the criteria of S1 Criterion B. When viewing the list of rented books, the user should be able to select individual entries Criterion C. When an entry is selected, the user should be able to click the renew button, permitting that they have renewals available Criterion D. The application should update the renewal date following the books rental period	Criterion A. No defects; Satisfies all criteria of S1 Criterion B. No defects Criterion C. No defects Criterion D. One defect found during unit testing. The date of the book selected wouldn't change when the renew button was pressed. Fixed immediately by modifying the date class slightly to cater for the changing of date. Date now changes to the correct date and satisfies this acceptance criteria.
A1	Criterion A. The application should recognise a difference in account permission level Criterion B. The application should have an interface for administrative controls Criterion C. This interface should have buttons to control the catalogue Criterion D. Clicking each button should create a window that allows the administrator to control the catalogue i. 'Add' should allow the administrator to enter the details of a new book ii. 'Remove' should allow the administrator to remove a book from the list of all books – this should not be available for books currently being rented iii. 'Modify' should allow the administrator to select a book from the list of all books. This will then present a window identical to add; however, the fields will be pre-populated with the data of the selected book Criterion E. When the user is finished with the interface, there should be a button at the bottom the save the changes and return to the previous window	Criterion A. No defects Criterion B. No defects Criterion C. No defects Criterion D. No direct defects; See following: i. One defect found during demonstration practise. Fine details were being passed as an integer (when they are a double) causing errors. Methods have been updated to resolve defect. ii. No defects iii. No defects Criterion E. No defects

MOVING FORWARD

RUNNING AND DEPLOYMENT INSTRUCTIONS

In its current form, the ELMS has been compiled as a Java application. Consequently, it is currently a self-contained application that can be shared and will function; however, this would include the current 'database', which is insufficient for proper use. Presuming the code is updated to include a network-accessible database (and likewise for the payment system for fines), the application could be made available for staff and students to download to their device, giving them access to the full functionality of the ELMS. Alternatively, the application could be hosted online – using one of the tools designed to convert a JavaFX application into a web-accessible application (all of which, to our knowledge, require payment and, as per the specification, have consequently not been employed) – so that the students and staff do not need to download it locally; however, it would likely be more efficient to build a web application that is specifically intended for this purpose if this is the desired use case.

Upon being launched, the application can be interacted with using the mouse, keyboard, or both. The user is presented with the login view upon launch, and after successfully logging in, are presented with the main menu from which they can access all the functionality available to their given permission level (student, staff or administrator) through the sub-menus. A video explaining the full functionality is available on the GitHub page, which outlines the possible interactions.

With its current design, there is some maintenance and checking required by the library or administrative staff: they must log in to check any book requests that have been placed by the staff; there is no ability to pay fines, as aforementioned, so they would need to be involved so that fines can be paid in-person at the library (this would also involve modifying the current code – most notably, disabling the demo 'pay fine' button); they would be required to log in to enable/disable the maintenance period; and, they would be required to log in to make any changes to the database.

IMPLEMENTATION EVALUATION AND RECOMMENDATIONS

Overall, given the available knowledge and resources, the team believes we have met the brief thanks, in part, due to our effective implementation of the tools. Using git/GitHub to manage our version management and collaboration with the code allowed us effectively manage issues and track progress throughout the sprints. Similarly, Trello was also very helpful in these regards.

Given the breadth of the task, our design was largely affected by the team's lack of experience. This is most clearly evidenced in that our application lacks a proper database. With a more experienced team, this project could have been completed to a much higher standard, with more features and options for the client – the latter largely due to more being able to have been completed within the given timeframe. For example, we could have developed a web application, which would allow easier integration to a remote database and greater accessibility for the end-users.

For the aforementioned reasons, it is evident that, while it is functional and meets the majority of the requested functionality/criteria, there is a great deal of improvement that could be made to the application. Given the number of tasks that are being passed on to the next team and the inefficiencies currently present in our application, it would almost be more efficient, both in the sense of time and costs, for the application to be redeveloped by a more experienced team. It is for these reasons that I, as the project leader, recommend that this application not progress in development beyond this stage. That is to say, I believe this application is a 'no go'.

APPENDIX

LINKS

GitHub

Link: https://github.com/IIIMattiasIII/SES1A-Spr21-T3G4/

Trello

Link: https://trello.com/b/o49eao1g/ses1a-g4-trello

Invite link for non-members (though, this should not be needed): https://trello.com/invite/b/o49eao1g/0d6faf7dba16434a7ad342d30e652f54/ses1a-g4-trello

Teams

SES 1A Tut 3 Group 4

CONTRIBUTION TABLE

For contributions to this document, please see the following table. For contributions to the application itself, please see the contribution details included at the end of the presentation slides.

	Chantel	**Mattias**	**Mohammad**	**Pulkit**	**Umair**	**Yize**
Use Cases	Completed	Contributed	Contributed	Contributed	Contributed	Contributed
User Stories and Acceptance Tests		Completed				
Design Planning	Contributed	Contributed				
Structure	Contributed	Contributed		Contributed		
Defects		Completed				
Traceability		Contributed	Contributed			
Going Forward		Completed				
Appendix		Completed				

MEETINGS

Meetings for this project were conducted in the subject-allocated class, Friday 5pm, on Zoom and the additional meetings were conducted on Teams at 3:30pm on Monday and, where necessary, Thursday. For this reason, a log of all meeting attendance can be found in Teams – each meeting was titled with the date and start time for convenience in tracking attendance. Additionally, I, the project lead, sent out a weekly 'newsletter' – if you will – that summarised the discussions/minutes from the weeks' meetings and provided an outline of the tasks that were to be completed in the coming week – these can be distinguished in Teams by their 'TBD' title banner.