California University of Pennsylvania

CSC 490: Senior Project

CalU Book Exchange

Design Document

Nick Cavalancia

Joe Rimsky

Ryan Dean

INSTRUCTOR COMMENTS/EVALUATION PAGE

Table of Contents

Table of Figures

Abstract

The CalU Book Exchange is an online resource that will allow students to buy and post books at cheaper rates than the traditional methods. Its purpose is to provide students with a stress-free method of acquiring their textbooks by keeping costs low and transaction between other students in the same situation. Users will be able to access The CalU Book Exchange through a website URL. The website will have both a desktop and mobile version. In order to access the website, users will need a valid internet connection along with a web browser installed on their device. The design details contained within this document are an expansion of the details discussed in both the requirements document and the specifications document. The main purpose of this design document is to provide additional details to the design team to implement the project completely and correctly. This includes input and output arguments, what the database will consist of, what modules are called, and how the framework of the website will look. Designers will use this document to prepare for implementation by reviewing all software components and ensuring they are working with each other properly. During implementation, this document will be used to complete each module and accurately check their inputs/outputs.

Description of this Document

Purpose and Use

This document is meant to provide a more in-depth description of the CalU Book Exchange. Each component of The CalU Book Exchange will be described fully and in-depth such that both client and developer will have a clear understanding of the individual mechanisms that make up the whole. This document is meant for those developers to look back upon for clear direction during the implementation phase. In essence this document serves as a descriptive blueprint for the software that is to be created.

Ties to the Specification Document

The specifications document overviewed the processes of the system and was the contract between the client and the developers. The design document is meant to describe those processes in greater detail. The individual functions will be broken down to their base level and described in detail. This will allow the developers to have a clearer picture of each module before and during creation.

Intended Audience

The main audience for this document will be the developers that are responsible for implementing the project through writing the code, and the secondary audience is the client. The development team will use this document as a clear guide as to how each module of the software should work. The client will use this document to gain a better understanding as to how this software works and to ensure that the project will operate in the intended way. This document may contain things unclear to the client and that is why we list the client as a secondary audience.

Project Block Diagram

The CalU Book Exchange application will be stored on a webserver. After authenticating, the user will be given permissions based on their access level, either Student or Administrator. The application will send data and receive data through a HTTP connection. The data coming from the user will consist of buttons and links on the web page that are clicked, text and selections that are filled in through forms, and images that will be uploaded to the server. That data will be validated and then stored in the system's database. Data coming from the web server to the user will consist of generated web pages that have static layouts but have dynamically generated results based on user input. Students and Administrators will be differentiated at authentication and will have permissions based on the type of user.

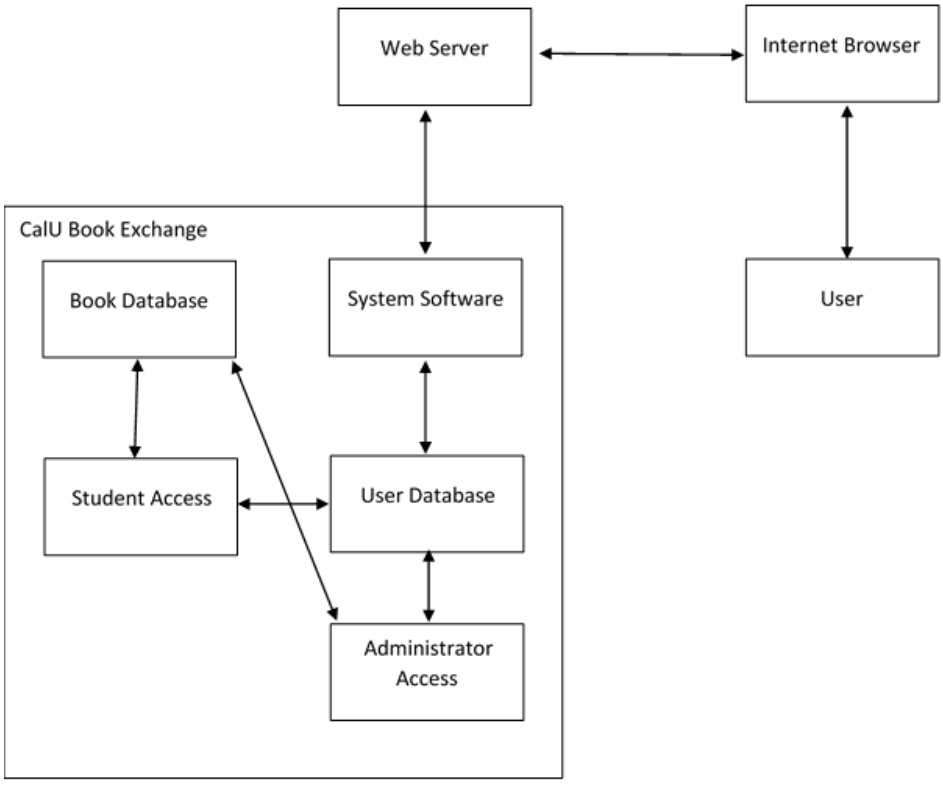

Figure 1- CalU Book Exchange Block Diagram

Design Details

System Modules and Responsibilities

The following elements of this section will go over the framework of integral system components and their functionality. Following is the architectural diagram of the project, a description of module cohesion, a description of module coupling, and a detailed tabular description of each module. Also, a dataflow diagram is included.

Architecture Diagram

Below is the architecture diagram which details the possible functionality based on the User type. With many branching paths, it is difficult to describe every possible path that a User can take in the scope of the project. Instead this diagram details user functionality based on User type and the path that they could take.

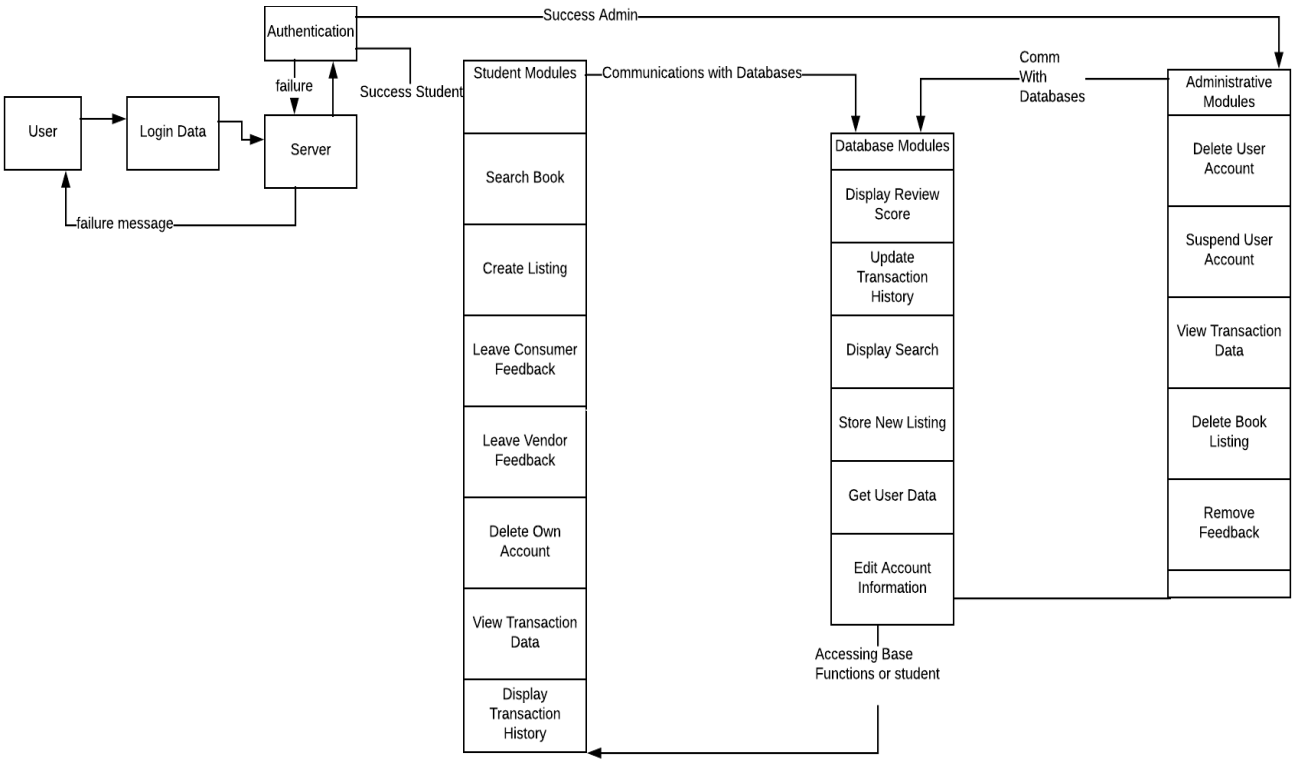

Figure 2 - Architecture Diagram

Student User

The Student User's modules are defined as what modules the student has access to directly. Access to the Student modules will be granted through authentication through the web server. An Administrator User will also have access to the modules under Student User Access.

Search Books Module

Module Name	Search Books
Module Type	Function
Return Type	Book Listing
Input Arguments	Book Name or ISBN, Price Filter
Output Arguments	None
Error Messages	"Invalid ISBN"
Files Accessed	Book Listing Database
File Changed	None
Modules Called	None
Narrative	The Search Books Module has the purpose of querying the database of book listings that match user created criteria that has been inputted through a form. The module will return all the matches that exist in the database and will generate a web page with the results. If no matches are found, it will display a message. If an invalid ISBN is entered, it will display an error message prompting to change the ISBN.

Module Cohesion

The Search Books Module has functional cohesion. It is purposed for a single goal and that is to retrieve a listing of books that have been requested by the user, but it uses data specific to this application only.

Create Listing Module

Module Name	Create Listing
Module Type	Function
Return Type	Generated Listing on Success
Input Arguments	Book Name, ISBN, Price, Username, Image
Output Arguments	None
Error Messages	"Required Information Missing”
Files Accessed	Book Listing Database
Files Changed	Book Listing Database
Modules Called	None
Narrative	The Create Listing Module has the purpose of inserting a new book listing into the database based on the user’s input. If part of the listing is left out, the user will be presented with an error message and told to modify their listing. If the listing is successfully created, the user’s web browser will load the page of the new listing for the user to view.

Module Cohesion

The Create Listing Module has functional cohesion. It is purposed for a single goal and that is to create a new listing to the database files for a user.

Leave Consumer Feedback Module

Module Name	Leave Consumer Feedback
Module Type	Function
Return Type	Consumer Feedback
Input Arguments	Feedback Score
Output Arguments	None
Error Messages	"Invalid Score”
Files Accessed	Student Data
File Changed	Student Data
Modules Called	None
Narrative	The Leave Consumer Feedback Module has the purpose of allowing the vendor to leave feedback for a consumer account.

Module Cohesion

The Leave Consumer Feedback Module has functional cohesion. Its purpose is to allow a vendor to leave a consumer account feedback.

Leave Vendor Feedback Module

Module Name	Leave Vendor Feedback
Module Type	Function
Return Type	Vendor Feedback
Input Arguments	Feedback Score
Output Arguments	None
Error Messages	"Invalid Score"
Files Accessed	Student Data
File Changed	Student Data
Modules Called	None
Narrative	The Leave Vendor Feedback Module has the purpose of allowing the consumer to leave feedback for a vendor account.

Module Cohesion

The Leave Vendor Feedback Module has functional cohesion. It is purposed with allowing the consumer to leave a vendor feedback for their account information.

Delete Own Account Module

Module Name	Delete Own Account
Module Type	Function
Return Type	None
Input Arguments	None
Output Arguments	None
Error Messages	None
Files Accessed	Student Data
File Changed	Student Data
Modules Called	None
Narrative	The Delete Own Account Module has the purpose of allowing the user to delete their own account from the database.

Module Cohesion

The Delete Own Account Module has functional cohesion. Its only purpose is to allow the user to delete their own account.

View Transaction Data Module

Module Name	View Transaction Data
Module Type	Function
Return Type	None
Input Arguments	Transaction Number
Output Arguments	None
Error Messages	"Invalid Transaction Number”
Files Accessed	Transaction Data
File Changed	Transaction Data
Modules Called	None
Narrative	The View Transaction Data Module has the purpose of allowing the user to view data from a specific transaction.

Module Cohesion

The View Transaction Data Module has functional cohesion. Its purpose is to all the user to view data from a specific transaction.

Display Transaction History Module

Module Name	Display Transaction History
Module Type	Function
Return Type	None
Input Arguments	None
Output Arguments	List of Transactions
Error Messages	None
Files Accessed	Student Data
File Changed	Student Data
Modules Called	None
Narrative	The Display Transaction History Module has the purpose of allowing a user to view their full transaction history.

Module Cohesion

The Display Transaction History Module has functional cohesion. It is purposed with the task of displaying a user's transaction history from their information on the database file.

Module Coupling

The modules in the Student section do not interact with each other for coupling to exist.

Administrator Modules

Delete User Account Module

Module Name	Delete User Account
Module Type	Function
Return Type	Boolean
Input Arguments	Username
Output Arguments	None
Error Messages	"User Does Not Exist"
Files Accessed	User Database
Files Changed	User Database
Modules Called	User Data
Narrative	The Delete User Account Module is used to remove a user from the service. It will check the name given with the usernames present in the system. Then it will display a confirmation message displaying the username entered. After confirming, this module will return true if it was deleted and false if it fails

Module Cohesion

The Delete User Account Module has functional cohesion. It is purposed with the task of allowing an admin account to remove a specific user's account from the database.

Suspend User Account Module

Module Name	Suspend User Account
Module Type	Function
Return Type	Boolean
Input Arguments	Username
Output Arguments	None
Error Messages	“User Does Not Exist”
Files Accessed	User Database
Files Changed	User Database
Modules Called	User Data
Narrative	The Suspend User Account Module is used to suspend a user from the service. It will check the name given with the usernames present in the system. Then it will display a confirmation message displaying the username entered. After confirming, this module will return true if it was deleted and false if it fails.

Module Cohesion

The Suspend User Account Module has functional cohesion. It is purposed with the task of allowing the admin account to suspend a specific user’s account for a certain amount of time.

View Transaction Data Module

Module Name	View Transaction Data
Module Type	Function
Return Type	None
Input Arguments	Transaction Identification (Book name, ISBN, ect)
Output Arguments	Transaction Data
Error Messages	“No Transaction Found”
Files Accessed	Transaction Database
Files Changed	None
Modules Called	None
Narrative	The View Transaction Data Module is meant to view a transaction's information such as the sold price, who the transaction was between and all relevant data concerning the transaction.

Module Cohesion

The View Transaction Data Module exhibits functional cohesion. Its only purpose is to allow the admin to view the transaction data from a specific transaction.

Delete Book Listing Module

Module Name	Delete Book Listing
Module Type	Function
Return Type	Boolean
Input Arguments	Book Identification (Book name, ISBN, ect)
Output Arguments	None
Error Messages	“Listing Does Not Exist”
Files Accessed	Book Database
Files Changed	Book Database
Modules Called	None
Narrative	The Delete Book Listing Module deletes a book from the listings by accessing the storage in which it is located and removing it. This is done when book listings violate standards.

Module Cohesion

The Delete Book Listing Module exhibits functional cohesion. It has a clear purpose in that it removes a book listing from the service.

Remove Feedback Module

Module Name	Remove Feedback
Module Type	Function
Return Type	Boolean
Input Arguments	None
Output Arguments	None
Error Messages	“Feedback Does Not Exist”
Files Accessed	Feedback Database
Files Changed	Feedback Database
Modules Called	None
Narrative	The Remove Feedback Module is meant to remove feedback that does not fall in line with community standards. This module sends a confirmation message to the user prior to deletion. After confirming, the feedback is removed.

Module Cohesion

The Remove Feedback Module exhibits functional cohesion. It has a clear purpose in that it removes the feedback from the service.

Module Coupling

The modules do not exhibit coupling inside the admin section. All the modules exhibit data coupling to the modules in the database class through data coupling.

Database Modules

Display Review Score Module

Module Name	Display Review Score
Module Type	Function
Return Type	None
Input Arguments	Username
Output Arguments	None
Error Messages	None
Files Accessed	Student Data
Files Changed	None
Modules Called	None
Narrative	The Display Review Score Module displays the average review score given to a user by other users.

Module Cohesion

The Display Review Score Module has functional cohesion. It is purposed for a single goal and that is to retrieve the average review score of a specific user and display it to the appropriate field on the website.

Update Transaction History Module

Module Name	Update Transaction History
Module Type	Function
Return Type	None
Input Arguments	List
Output Arguments	None
Error Messages	None
Files Accessed	Student Data
Files Changed	Student Data
Modules Called	None
Narrative	The Update Transaction History Module updates a user's transaction history.

Module Cohesion

The Update Transaction History Module has functional cohesion. Its purpose is to update the transaction history of a specific user and save it to the database files.

Display Search Module

Module Name	Display Search
Module Type	Function
Return Type	None
Input Arguments	List of Strings
Output Arguments	None
Error Messages	None
Files Accessed	Listing Data
Files Changed	None
Modules Called	None
Narrative	The Display Search Module displays the list of relevant posts that match the user's search criteria.

Module Cohesion

The Display Search Module has functional cohesion. It is purposed for a single goal and that is to display the appropriate search to the user after they input the requirements for a search (ISBN, textbook name, author, etc.).

Store New Listing Module

Module Name	Store New Listing
Module Type	Function
Return Type	None
Input Arguments	List
Output Arguments	None
Error Messages	None
Files Accessed	Listing Data
Files Changed	Listing Data
Modules Called	None
Narrative	The Store New Listing Module stores a new listing to the listings file on the database.

Module Cohesion

The Store New Listing Module has functional cohesion. It is purposed for a single goal and that is to store a new listing to the database after a user makes a post.

Get User Data Module

Module Name	Get User Data
Module Type	Function
Return Type	List
Input Arguments	Username String
Output Arguments	None
Error Messages	None
Files Accessed	Student Data
Files Changed	None
Modules Called	None
Narrative	The Get User Data Module retrieves the user data of a specific user from the Student data files.

Module Cohesion

The Get User Data Module has functional cohesion. It is purposed for a single goal and that is to retrieve a specific user's information for use by other functions (Displaying user information, processing user data, etc.).

Edit Account Information Module

Module Name	Edit Account Information
Module Type	Function
Return Type	None
Input Arguments	Username and any desired changes
Output Arguments	Student Data
Error Messages	None
Files Accessed	Student Data
Files Changed	Student Data
Modules Called	None
Narrative	The Edit Account Information Module will edit/update any information that the user would like to change.

Module Cohesion

The Edit Account Information Module has functional cohesion. It is purposed for a single goal and that is to edit a specific user's account information based off of the input that the user enters. It will then be updated on the database.

Module Coupling

The modules in the Database section do not interact with each other for coupling to exist.

Design Analysis

The following diagram, Figure 3, depicts the dataflow of The CalU Book Exchange. Data is sent through the website and stored into a database after the user edits anything that they have access to. After this, the data is updated for use across the entire website.

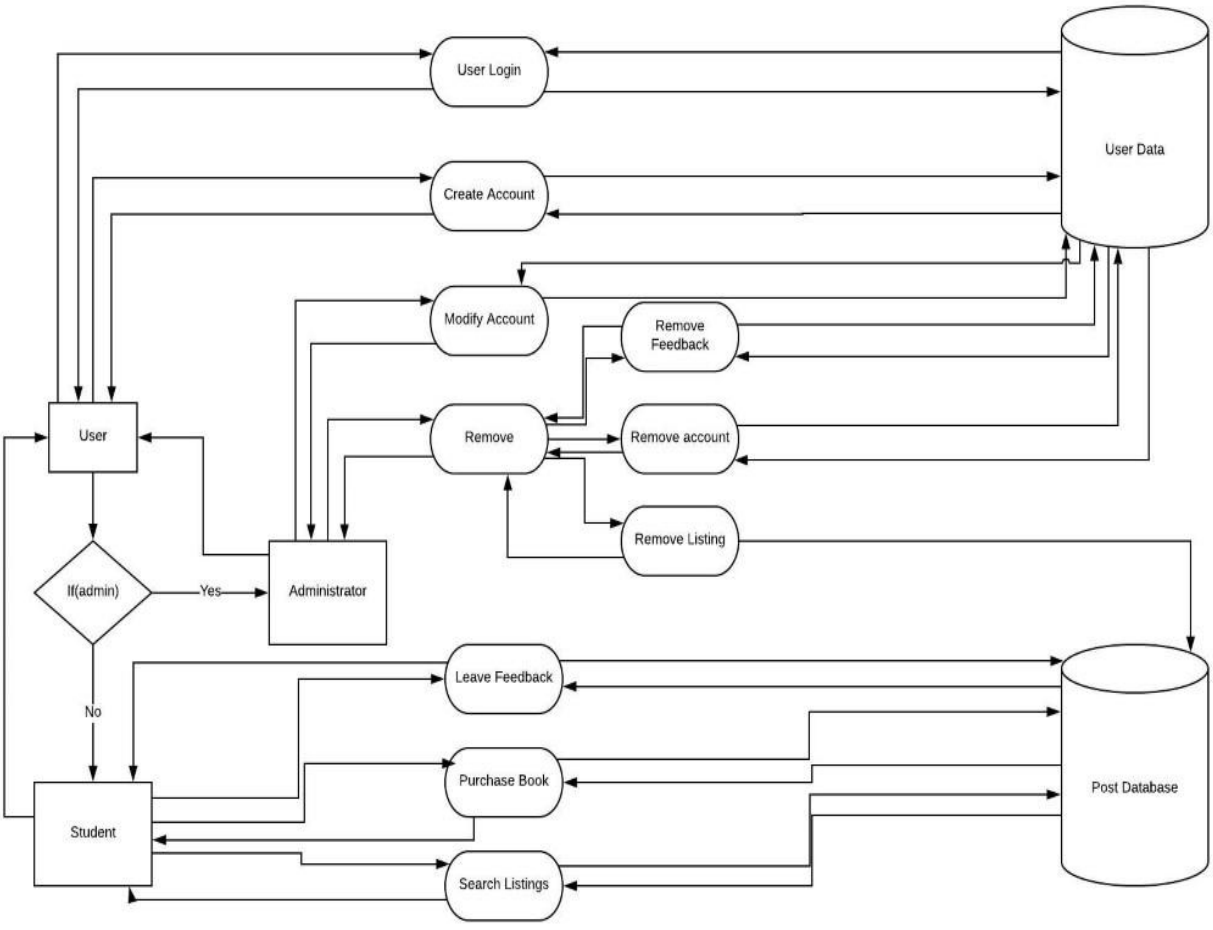

Figure 3- CalU Book Exchange Dataflow Diagram

State Chart Diagram for CalU Book Exchange

The CalU Book Exchange application will require a user, whether it be Admin or regular User, to login, manage their account, and work with the website by posting and searching.

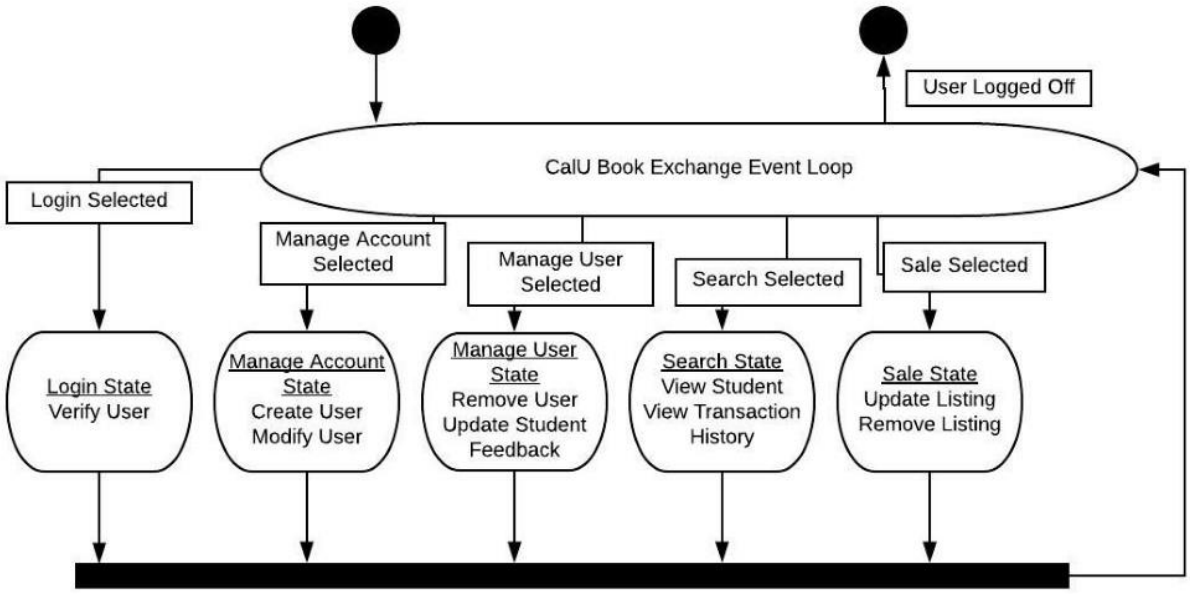

Figure 4- CalU Book Exchange State Diagram

Design Organization (Object oriented design)

Detailed Tabular Description of Classes/Objects Description

User Class	Student Class	Admin Class	Database Class
Username : String Password : String Category : Integer Permissions : List	Name : String Location : String Feedback : integer	Name: String Location: String Feedback: integer	
VerifyUser() AddUser() ModifyUser() ModifyPost() GetUserName() GetUserCategory() GetFeedback()	SearchBook() CreateListing() LeaveConsumerFB() LeaveVendorFB() DeleteOwnAccount() ViewTransactionHi() DisplayTransaction()	RemoveFeedback() DeleteUserAccount() ViewTransactionData()SuspendUserAccount()DeleteBookListing()	GetPost() AddStudent() DisplaySearch() DisplayReview() GetUserData() EditAccInfo() StoreNListing()

Figure 5- CalU Book Exchange Class Diagram

User Class

The User Class is the parent class of the Student and Admin classes. It will contain functions available to both classes.

- Username, data type: string – this data will hold the name of the account
- Password, data type: string – this data will hold the secure copy of the password to compare to the typed password on login
- Category, data type: integer – this data will hold the value that will determine whether the user is a student or administrator
- Permissions, data type: list – this will hold the permissions given to each user
- VerifyUser() - this will allow the system to verify if the username and password that the user entered are correct
- AddUser() - this will allow the system to add a user to the database
- ModifyUser() - this will be used to allow both students and admins to modify their account
- ModifyPost() - this will be used to allow both students and admins to modify a post
- GetUserName() - this will be used to retrieve the username of a user
- GetUserCategory() - this will be used to retrieve the category of a user (Student or Admin)
- GetFeedback() - this will be used to retrieve a user's feedback score (1-10)

Admin Class

The admin class is used to store admin and student information in the server. Admins will have access to adding and deleting users/postings.

- Name, data type: string - this data holds the name of the user for contact information
- Location, data type: string – this data holds the location of the user for posting purposes
- Feedback, data type: integer – this data will hold the feedback rating of the user (1-10)
- DeleteUserAccount() - allows the admin to delete a specific user's account
- SuspendUserAccount() - allows the admin to suspend a specific user's account
- View TransactionData() - allows the admin to view any transaction data on the database
- DeleteBookListing() - allows the admin to delete a book listing
- RemoveFeedback() - allows the admin to remove any feedback given to a consumer/vendor

Student Class

The student class is used to store student information in the database that users can view and change with the given functions.

- Name, data type: string - this data holds the name of the user for contact information
- Location, data type: string – this data holds the location of the user for posting purposes
- Feedback, data type: integer – this data will hold the feedback rating of the user (1-10)
- SearchBook() - searches through the database for a specific book title
- CreateListing() - allows a user to create a listing
- LeaveConsumerFeedback() - allows the vendor to leave consumer feedback
- LeaveVendorFeedback() - allows the consumer to leave fender feedback
- DeleteOwnAccount() - allows the user to delete their own account
- ViewTransactionData() - allows a user to view their own transaction data
- DisplayTransactionHistory() - displays a user's transaction history

Database Class

The database class is used to store all data made by users in the database.

- DisplayReviewScore() - displays the average review score given to a user by other users
- UpdateTransactionHistory() - updates a user's transaction history
- DisplaySearch() - displays the list of relevant posts that match a given search criteria
- StoreNewListing() - stores a new listing onto the database
- GetUserData() - retrieves a user's data from the database
- EditAccountInformation() - edits a user's account information

Module Cohesion

User Class

The class "User Class" will be used as a parent class to the Admin and Student classes. It will consist of functions and data that both classes will have access to.

Admin Class

The class "Admin Class" will be used to store and process Admin information on the server. This class will be the backbone of all Admins, giving them the ability to access certain modules.

Student Class

The class "Student Class" will be used to store and process Student information in the database that users can view and change the modules that they are given.

Database Class

The class "Database Class" will be used to store and process all information on the database.

Modules

The following modules will operate with functional cohesion. Each function listed will consist of its own unique code.

User Class: The following operations can be used within this module: VerifyUser(), AddUser(), ModifyUser(), ModifyPost(), GetUserName(), GetUserCategory(), GetFeedback().

Admin Class: The following operations can be used within this module: DeleteUserAccount(), SuspendUserAccount(), ViewTransactionData(), DeleteBookListing(), RemoveFeedback().

Student Class: The following operations can be used within this module: SearchBook(), CreateListing(), LeaveConsumerFeedback(), LeaveVendorFeedback(), DeleteOwnAccount(), ViewTransactionData(), DisplayTransactionHistory().

Database Class: The following operations can be used within this module: DisplayReviewScore(), UpdateTransactionHistory(), DisplaySearch(), StoreNewListing(), GetUserData(), EditAccountInformation().

Module Coupling

UserClass (*Class*)

The UserClass class functions under data coupling by how it interacts with the user and database. The module is used to process user information before it is sent to the database for storage. All functions in this class have a unique input and output.

AdminClass (*Class)*

The AdminClass class functions under data coupled by how it interacts between the Student, Admin, and Database. The module is used to process any system maintenance, whether it be deleting spam accounts or illegal posts. All functions in this class have a unique input and output.

StudentClass (*Class*)

The StudentClass class functions under data coupled by how it interacts between Student and Database. The module is used to process user information before it is sent to the

database for storage. All functions in this class have a unique input and output. This will give the students a simplistic service to use.

DatabaseClass (*Class*)

The DatabaseClass is the backbone of data. It is data coupled to the other classes as all classes access the data from the DatabaseClass. The Database class is used to update the database in which the relevant data for our project is stored.

Real-time Requirements

The system will have near real time requirements. That means the system will give itself an allotted specified amount of time before failure. This is to ensure that enough time has been given for the processes to complete and to account for any delays.

Messages

Message Code	Message Description
0000	User Account Approved
0001	User Login Approved
0010	Nothing for Search
0011	Search Results returned
0100	User Data Modified
0101	User Suspended
0110	User Removed
0111	User Feedback Removed
1000	User Listing Removed
1001	Display User Transactions
1010	Display User Reviews
1011	Display User Information
1100	Updated Transaction History
1101	Updated User Reviews
1110	Updated Book Listing
1111	Own account Deletion Success

Error Code	Error Description
00A	Invalid Registration Data
00B	Invalid Login Data
00C	User Not Found
00D	Transaction Not Found
00E	Invalid User Update
00F	Listing Could Not Be Created
00G	Listing Not Found
00H	Feedback Not Found
00I	Transaction History Could Not Be Found

Narrative/PDL

1. Login Menu:
    a. Register account
        Successful?
        i. No? → Error message "Invalid information, please check that all required fields are filled out correctly"
        ii. Yes? → Create Account → login database update
    a. Login account
    Successful?
        i. No? → Error Message "Invalid Login Credentials"
        ii. Yes? → Login to Account → Main Menu
2. Main Page:
    a. Main Menu {
    b. Search
    Matches Found?
        i. No? → Error Message "Seems like nothing is here"
        ii. Yes? → Display matches in a list
            1. Fields
            2. Seller
            3. ISBN
            4. Title
    c. User Account Data
    d. View Transaction History
    e. Current Transactions
    f. Cart
    g. Create listing
    h. Administrator's options
    Administrator?
        i. No? → Do not display
        ii. Yes? → Display }
3. Search Page:
    a. Main Menu {
    b. Search

Matches Found?
i. No? → Error Message “Seems like nothing is here”
ii. Yes? → Display matches in a list
1. Fields
2. Seller
3. ISBN
4. Title

c. User Account Data
d. View Transaction History
e. Current Transactions
f. Cart
g. Create Listing
h. Administrator’s options
Administrator?
i. No? → Do not display
ii. Yes? → Display }
i. Select listing
i. Display listing
1. Title
2. ISBN
3. Seller
4. Edition
5. Date Published
6. Seller rating
7. Book condition
8. Asking price
ii. Add to Cart

4. User Account Data
a. Main Menu {
b. Search
Matches Found?
i. No? → Error Message “Seems like nothing is here”
ii. Yes? → Display matches in a list
1. Fields
2. Seller
3. ISBN
4. Title
c. User Account Data
d. View Transaction History
e. Current Transactions
f. Cart
g. Create listing
h. Administrator’s options
Administrator?
i. No? → Do not display
ii. Yes? → Display }

i. Display User account Data
    i. First Name
    ii. Last Name
    iii. Phone Number * optional
    iv. Email Address
    v.Customer Rating
    vi. Vendor Rating
    vii. Username
j. Change Password
    Password?
    i. Current =! Current → Error "Current Password was incorrect
    ii. New?
        1. No? → Error "New cannot be Current"
        2. Yes? → Password changed → update login

5. View Transaction History
    a. Main Menu {
    b. Search
        Matches Found?
        i. No? → Error Message "Seems like nothing is here"
        ii. Yes? → Display matches in a list
            1. Fields
            2. Seller
            3. ISBN
            4. Title
    c. User Account Data
    d. View Transaction History
    e. Current Transactions
    f. Cart
    g. Create listing
    h. Administrator's options
        Administrator?
        i. No? → Do not display
        ii. Yes? → Display }
    i. Display Transaction History
        i. Asking Price
        ii. Title
        iii. ISBN
        iv. Date Finalized
    j. Search Transaction history
        Transaction found?
        i. No? Error "Transaction seems to be missing"
        ii. Yes? Display Transaction details
            1. Asking Price
            2. Title
            3. ISBN
            4. Date Finalized

6. Current Transactions
    a. Main Menu {
    b. Search
        Matches Found?
        i. No? → Error Message “Seems like nothing is here”
        ii. Yes? → Display matches in a list
            1. Fields
            2. Seller
            3. ISBN
            4. Title
    c. User Account Data
    d. View Transaction History
    e. Current Transactions
    f. Cart
    g. Create listing
    h. Administrator’s options
        Administrator?
        i. No? → Do not display
        ii. Yes? → Display }
    i. Display current transactions list
        i. Asking Price
        ii. Title
        iii. ISBN
        iv. Time since listing
    j. Search Current Transactions
    Transaction found?
        i. No? Error “Transaction seems to be missing”
        ii. Yes? Display Transaction details
            1. Asking Price
            2. Title
            3. ISBN
            4. Time since listing
7. Cart
    a. Main Menu {
    b. Search
    Matches Found?
        i. No? → Error Message “Seems like nothing is here”
        ii. Yes? → Display matches in a list
            1. Fields
            2. Seller
            3. ISBN
            4. Title
    c. User Account Data
    d. View Transaction History
    e. Current Transactions
    f. Cart

g. Create listing
h. Administrator's options
Administrator?
i. No? → Do not display
ii. Yes? → Display }
i. Display Cart items

8. Create Listing
a. Successful?
i. No? → Error "Error, Check that all required fields are completed properly"
ii. Yes? → Create Listing

Decision: Programming Languages/Reuse/Portability

The reason for selecting the programming languages we did was because of their effectiveness in the domain of which we are working. JavaScript is good for creating web-based services. HTML is effective at creating and managing web page designs. The flexibility of JavaScript allows for the development team to create an interactive front end, while also easily managing back end code through additional JavaScript based languages.

Implementation

Implementation of the "CalU Book Exchange" will take place after development in the Spring 2019 semester. The design team consists of three California University of Pennsylvania students pursuing their bachelor's degree in Computer Science. Nick Cavalancia and Joe Rimsky possess a strong knowledge base of JavaScript and similar languages. Ryan Dean is our lead database developer, as he has the most experience with SQL and other database management languages. Our project is going to require an extensive amount of maintenance if this system were to continue to be used for many years after our graduation. To ensure that the system will be easy to understand and maintain, we will be paying close attention to all aspects of the software through the development stage. Having a simple and structured software will be necessary to keep user experience as smooth as possible.

Design Testing

Rigorous testing will be employed in order to deliver the best product that the development team can. That testing will be in the form of individual testing, group testing, and outside testing. Individual testing will be the responsibility of each group member. Each group member is encouraged both to put forth ideas and address bugs. Group testing allows for the development team to all hear each other's fresh ideas as they come. Outside testing is to get even more input into what can be improved. This will be done by people outside the development team. This constant testing will improve the quality of the final product.

Appendix A: Team Details

The leader for the design document was Ryan Dean. He was chosen because he is aware of the structure of the program the best and has a clear idea of how each module will work. The design details are a major part of the CalU Book Exchange to ensure that each section of the program is performing is required function. This section required the most time and thought to properly list and describe every module and its function. The core details of this were completed by all three members, as well as the informational portions. Each member was required to review the document and understand what each section details. Doing so helped ensure that there were no errors and the overall content was thoughtful and correct.

Appendix B: Writing Center Report

I, the undersigned, certify that this document has been reviewed by a member of the staff at the California University of Pennsylvania Writing Center.

Signature:_____ Date: _____

Appendix C: Workflow Authentication

I, Nick Cavalancia, certify that I have performed the actions specified in this document.

Signature:_____ Date: _____

I, Ryan Dean, certify that I have performed the actions specified in this document.

Signature:_____ Date: _____

I, Joe Rimsky, certify that I have performed the actions specified in this document.

Signature:_____ Date: _____