NExTbooks

A Group Project For CMPT370 T1 2011

Final Milestone

Submitted By
Group04: *"The League of Extraordinary Programmers"*
Jason Knight *jwk881*
Colin Larson *cml220*
Jiayuan Hu *jih748*
Luke Robinson *ljr458*

Submitted To
Dr. Chanchal Roy
November 24, 2011

Abstract: Our group project was to conceive, design and implement a software system. of our choosing. We chose "Nextbooks", a Java based GUI for searching and renting textbooks. We developed the back end of the system with mySQL. We choose to focus on consistency and simplicity as our core values for this project. Though we have been instructed in all we needed to know we were still starting out in a somewhat unfamiliar landscape. We had limited experience with having the kind of freedom in direction that we had with this project. Our goals were simple: learn how to work in a group and develop a clean and usable system. Overall I feel we were successful in this. Our software design isn't breathtaking and probably isn't very portable, but the overall result is solid.

-The League

Table of Contents

1. Introduction

1.1 System Description

Our system allows users to create an account, search and browse through a catalogue of textbooks to rent, pay for a rental, and read their rented books all from within the same service. The system uses a mySQL database to store information about users and textbooks and a Java front end. The books will be downloaded and read from an online host. The system should be easy to use from both a user and administrator's perspective.

1.2 Business Case

Digital stores have potential to be very lucrative. With no waste and less staff, a digital bookstore is way more efficient than a street-based store. Textbook rentals could be an untapped goldmine among university students. The service could be sold to Universities. The opportunity for profit is undeniable.

1.3 User Goals

- Clean, uniform interface
- Fast and easy to find the book you want
- Front end for administrator duties

**Note: An important goal that we cannot implement for obvious reasons is having the Payment System reliable and secure

1.4 User Scenarios

Reading

A college student desires to read through his or her purchased textbooks from anywhere. Our System will allow the user to login from any location with an internet connection and have remote access to all of their purchased books.

Searching

A student needs to find all the required textbooks for his or her upcoming courses. Our system will support searching by Title, ISBN and Author.

Renting

A student is having trouble affording the costs of buying textbooks year after year to only use them for 4 months each. Our system will allow users to rent the textbooks only for desired term.

Cart Usage

To improve the user experience, we will model modern Web 2.0 sale systems' idea of a cart. A cart is a digital equivalent of a physical shopping cart. It will allow users to temporarily store their textbook selections to make all the purchases in one lump sum. Obvious requirements are user's being able to add to, subtract from and view the cart.

1.5 Scope Document

The first version of our software will simply be a catalogue with 3 books in it which are capable of being viewed by the user, purchased and finally viewed in our .PDF reader [1].

1.6 Original Project Plan (Circa September 25th, 2011)

Goal	Date	Primary Developer	Status	Hours
Logo Design	Sept. 26th	Luke Robinson	Completed	1
Initial Database Design	Sept. 29th	Colin Larson	In Progress	1
Raw Layout Design	Sept. 29th	Jiayuan Hu	In Progress	2
PDF Reading Prototype	Sept. 29th	Colin Larson	In Progress	2
Catalogue Prototype	Oct. 3rd	Jason Knight	Yet To Start	~2
Profile Prototype	Oct. 10th	Luke Robinson	Yet To Start	~3
Book Info Page	Oct. 10th	Colin Larson	Yet To Start	~1
Page Notes	Oct. 24th	Jason Knight	Yet To Start	~5
Login System	TBA	Jiayuan Hu	Yet To Start	~5
Review System	TBA	TBA	Yet To Start	~8
Iterative Development of all aspects until completion				

1.7 User Involvement Plan

We will involve users in two ways: evaluation and modification. During development we will use other students to act as users of our system. We shall ask them to evaluate the different features of our system. Specific questions are “evaluate the ease of process of finding a book”, “using the cart” or “did you enjoy the aesthetics of the program” etc. We will consider changes proposed by the testers.

2. Requirements and Early Design

2.1 Summary Use Cases:

Level: Summary
Actors: Users
Goal: We want a user to be able to find textbooks for their classes
Activities: From the Catalogue users will be able to select the find books by searching for the course number associated with the textbook.

Level: Summary
Actors: Users
Goal: We want users to be able to easily find what books they have.
Activities: From the users profile page, they should be able to easily see what books they have currently rented.

Level: Summary
Actors: Administration Staff
Goal: System Admins should be able to read all a users information (except password).
Activities: An Administrator should be able to search for a user by name or Email and view their rental history.

Level: Summary
Actors: Administration Staff
Goal: Errors should be properly handled
Activities: Any errors due to network problems or other issues should be cleanly logged in a file for the Administrators to view.

Level: Summary
Actors: Users
Goal: Users should be able to take notes specific for each page.
Activities: A user should be able to write notes on each page of a textbook wherever they want these notes should be specific to the user.

Level: Summary
Actors: Administrator Staff
Goal: A Administrator should be able to add, remove and modify books.
Activities: When logged in as an administrator, new functionality to add remove or modify books from the database should be enabled.

Level: Summary
Actors: Users
Goal: Each book should contain all relevant information
Activities: Users should be able to see all information about a book from the book page. Information such as number of rentals, price, summary, author, title, ISBN, version, publisher, publish date etc.

Level: Summary

Actors: Users
Goal: Users should be able to modify their profile
Activities: A user should be able to change their password, email & favourite books all from the users profile page. Users should be able to see this information about other users.

Level: Summary
Actors: Users
Goal: Textbooks should show user reviews
Activities: Users should be able to see reviews of a textbook from other users on the book page. Users should also be able to easily add a new review with a 5 star rating system.

Level: Summary
Actors: Administration Staff
Goal: Admins should be able to remove offensive reviews
Activities: An administrator should be able to remove any offensive reviews from the book page.

2.2 Fully Dressed Use Cases:

1. Use Case Name (Primary Use Case)
Simple Book Rental
Scope
NextBooks Textbook Rental Service
Level
User Goal
Primary Actor
User
StakeHolders & Interests
User: Wants renting to be easy, fast and simple
Wants proof of purchase
Administrator: Wants count of purchases for each book
Wants record of purchases and payment
Pre-Conditions
User has a validated & unique account
User has internet connection
Success Guarantee
Rental is saved. Cost is properly calculated
User account is updated
Main Success Scenario
1. User logs in to NextBooks
2. User opens the Catalogue window
3. User finds and selects the book
4. User proceeds to checkout
5. User confirms their selection
6. User proceeds to payment.
7. User enters payment information and sale is completed
8. User is presented with a Receipt
9. Users account is updated with the purchased book

Extensions

If payment system fails, system must inform user and offer
an alternative payment method if available.
If a user selects a book they already own, they will be notified
and not charged for these books.

Special Requirements

Multiple books may be selected for purchase at a time.

Technology

Mouse used to select items, keyboard used to enter search terms.
Books identified by Name, Author, ISBN

Frequency

High use during start of semesters where every user rents textbooks.
Low use throughout duration of semesters.

2. Use Case Name

Book Reading

Scope

NextBooks Textbook Rental Service

Level

User Goal

Primary Actor

User

StakeHolders & Interests

User: Wants to easily read purchased books
User wants no lag when reading

Pre-Conditions

User has a valid account and has at least one book rented
User has internet connection

Success Guarantee

User is able to open their rented book.

Main Success Scenario

1. User logs in to NextBooks
2. User opens his/her profile
3. User selects a book from the Current Rentals list
4. Book is opened in Reader

Extensions

None

Special Requirements

Custom notes implemented if budget allows
Technology
.PDF Reader [1] to open PDF files
Server to store PDF files on
Database to store relevant book data
Database to store user login information

Frequency

Extremely frequent

3. Use Case Name

Searching

Scope

NextBooks Textbook Rental Service

Level

User Goal

Primary Actor

User

StakeHolders & Interests

User: Wants to easily search through the catalogue
User wants to search by Title, ISBN, Author
User wants searching to be fast

Pre-Conditions

User has a valid account
User has an internet connection

Success Guarantee

User is able to search by ISBN, Author and Book Name

Main Success Scenario

1. User logs in
2. User opens the catalogue window
3. User enters a name of a desired textbook
4. User is given a list of items whose title contains search term

Extensions

If no item is found, a message should be displayed with that information
User can also choose to search by ISBN or Author

Special Requirements

None

Technology

Database to store book information
Database to store user information
Database search techniques for searching

Frequency

Extremely frequent

4. Use Case Name

Shopping Cart Usage

Scope

NextBooks Textbook Rental Service

Level

User Goal

Primary Actor

User

StakeHolders & Interests

User: Wants to be able to pay for multiple rentals at a time

Pre-Conditions

User has a valid account
User has internet connection

Success Guarantee

User is able to search by ISBN, Author and Book Name

Main Success Scenario

1. User logs in
2. User opens the Catalogue window
3. User selects a desired book and adds it to Cart
4. Repeat step 3 until all desired books are in Cart
5. User continues to checkout
6. User verifies shopping Cart contents
7. If desired, user can remove items from cart
8. User enters payment information (Credit Card)
9. User receives receipt and Cart contents added to profile page

Extensions

User can remove items from cart at anytime from Cart page

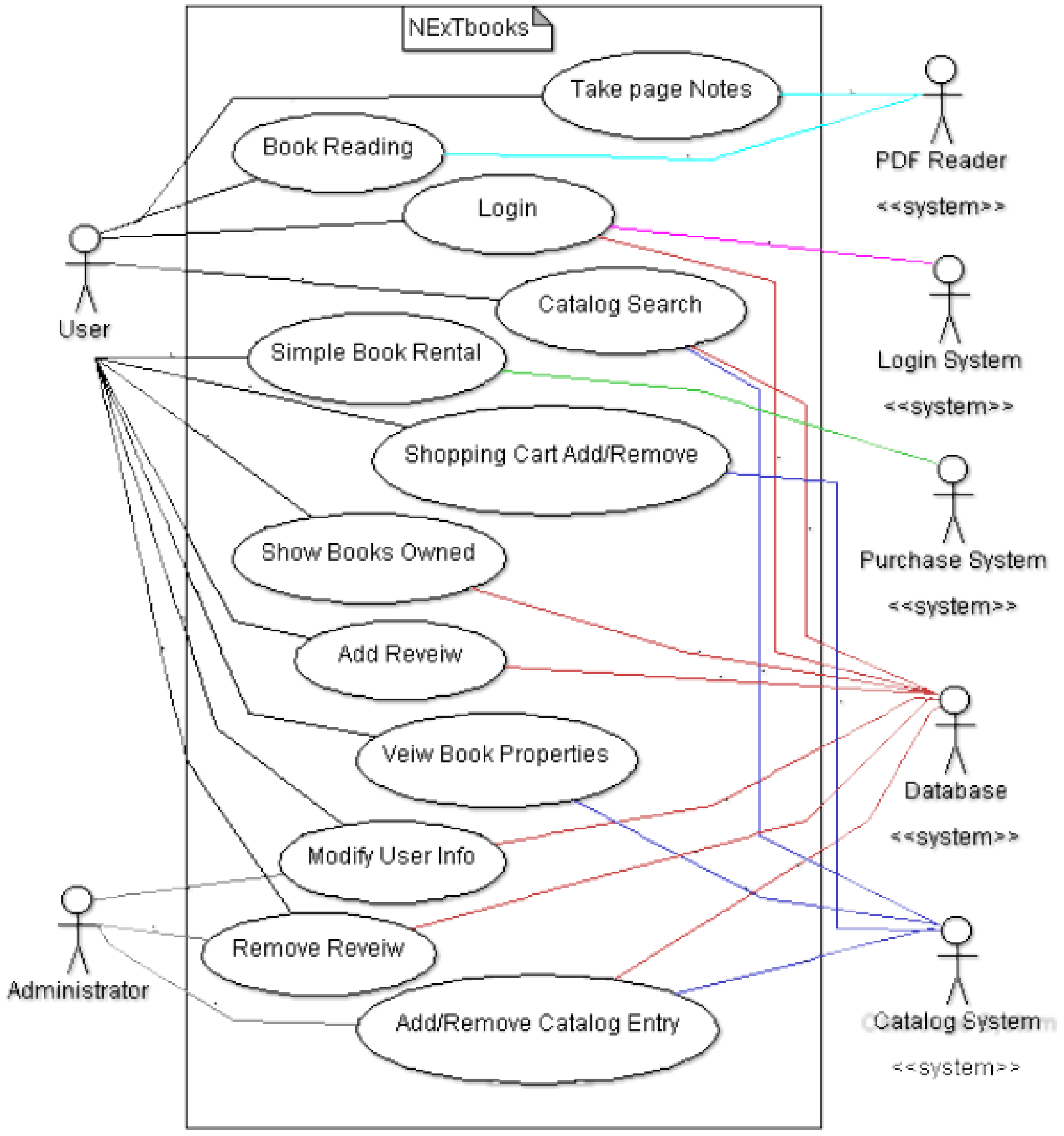
Technology

Database for user information and book information
Cart will be stored locally, not on a database
Credit Card Payment System

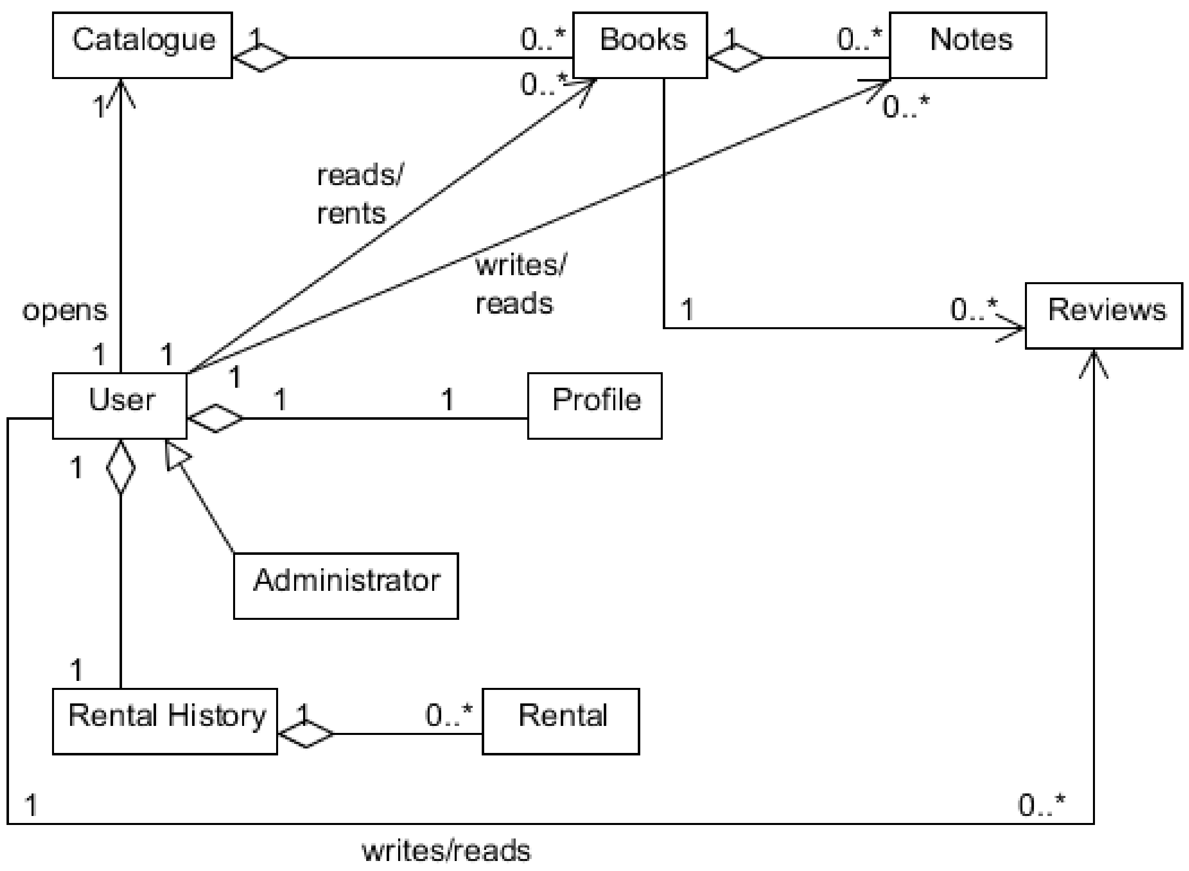
Special Requirements

If payment fails user must be notified and given a reason
Frequency
Extremely frequent

2.3 Use Case Diagram



2.4 Domain Model



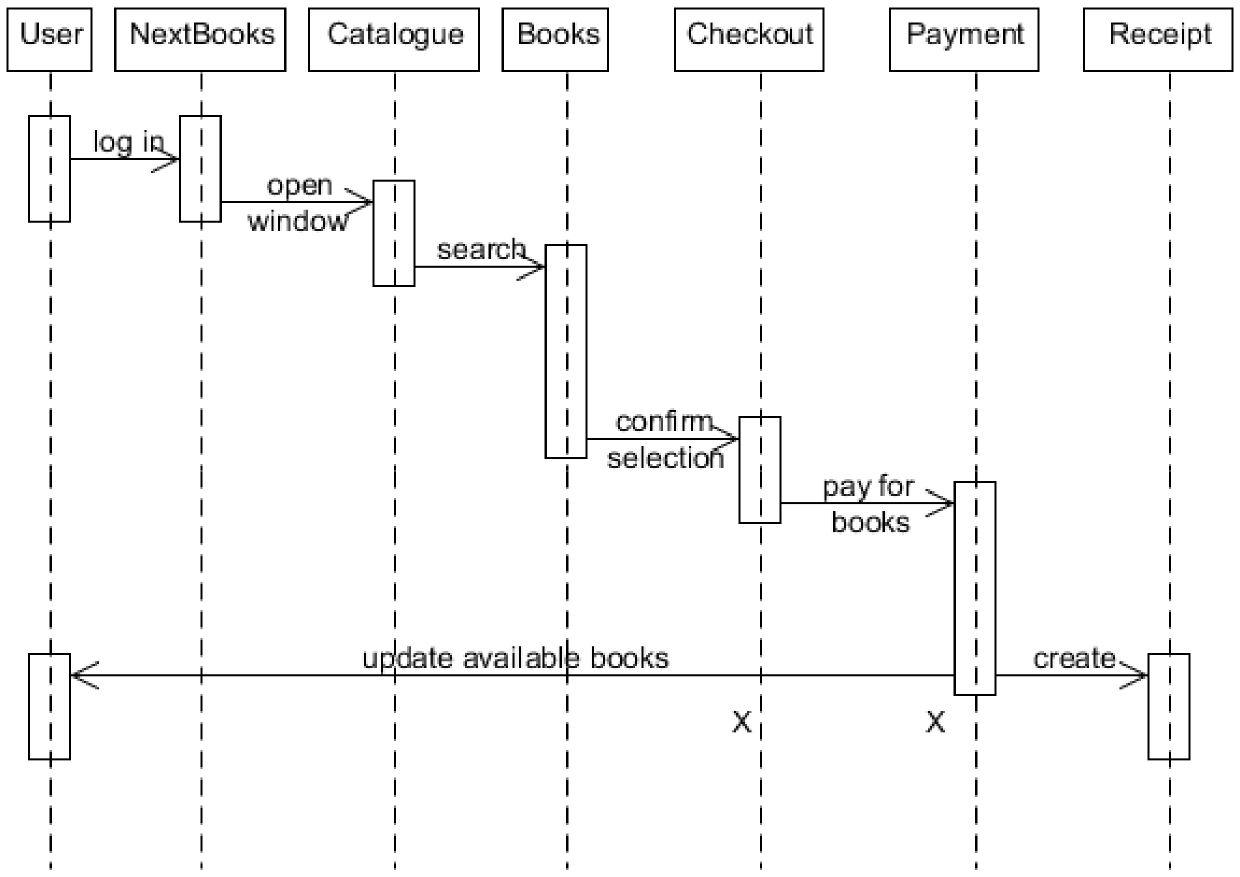
2.5 Glossary

Account: Represents all of a unique users information (username, password, ID, rentals etc).
Author: The author of a book.
Catalogue: The system for browsing textbooks and adding textbooks to the Shopping Cart.
ISBN: International Standard Book Number, a unique ID for each book.
Profile: Users personal page that tells them about their current Rentals and Rental History.
Reader: The system for reading your downloaded .PDFs. Referenced at [1].
Receipt: A proof of purchase for a rental.
Rental: The actual action of selecting a book that you will own for the duration.
Rental History: An informative collection of past rentals.
Shopping Cart: A temporary storage system for textbooks a user is considering.
Textbooks: The .PDF files that users will be renting and reading.
Title: The title of a book.

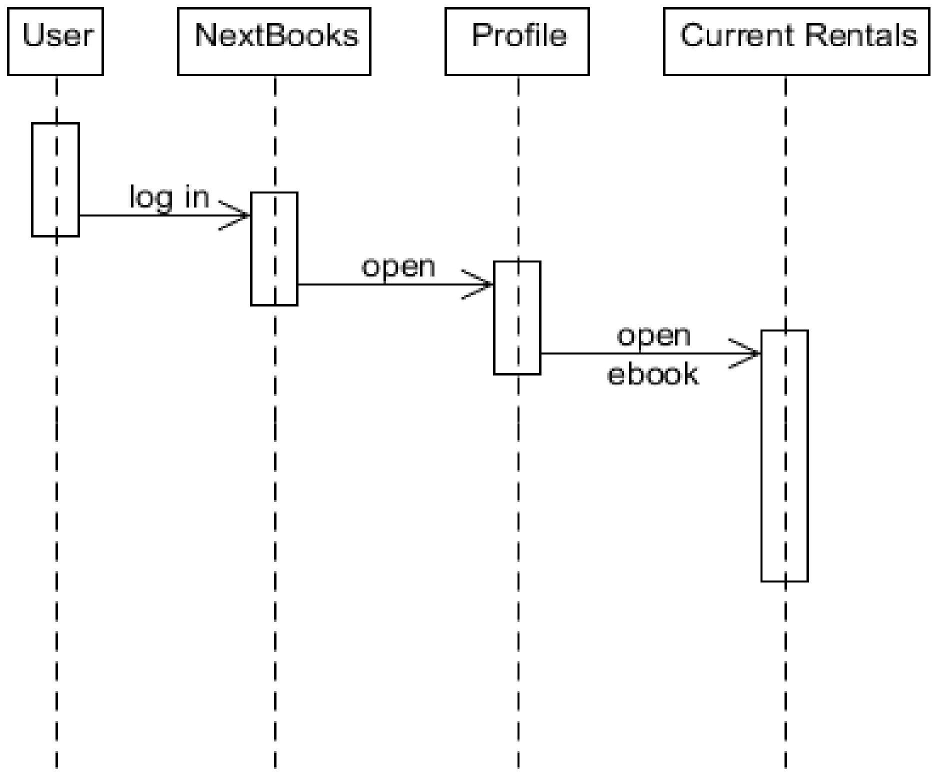
2.6 Supplementary Requirements
Performance: We want our system to be quick. It uses internet resources so this will be a real requirement for us.
Usability: If our system is functional but not usable, no students will use it.
Reliability: We need fail safes if the internet, or our other external systems fail.
Supportability: We need to keep the code clean and well designed so that we can interface with many systems in the future.

2.7 System Sequence Diagrams

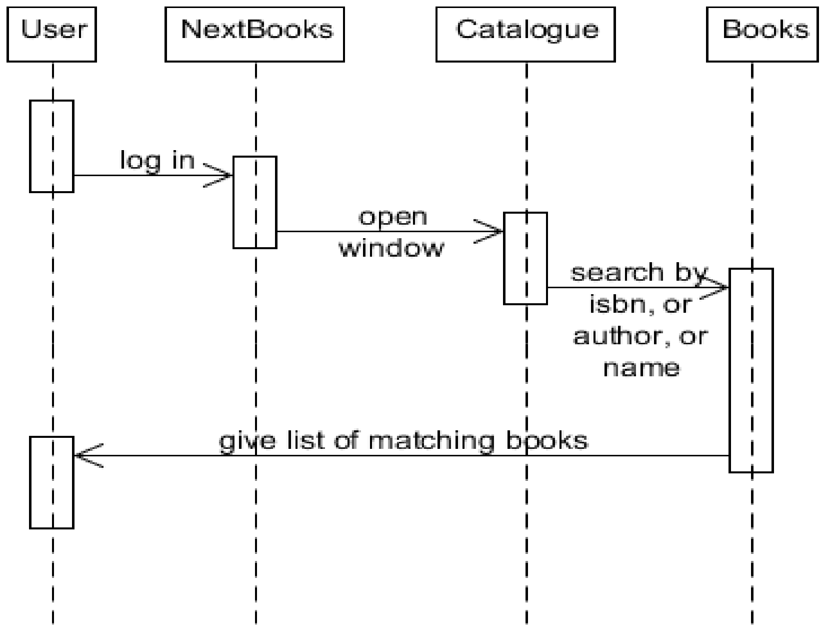
Rental:



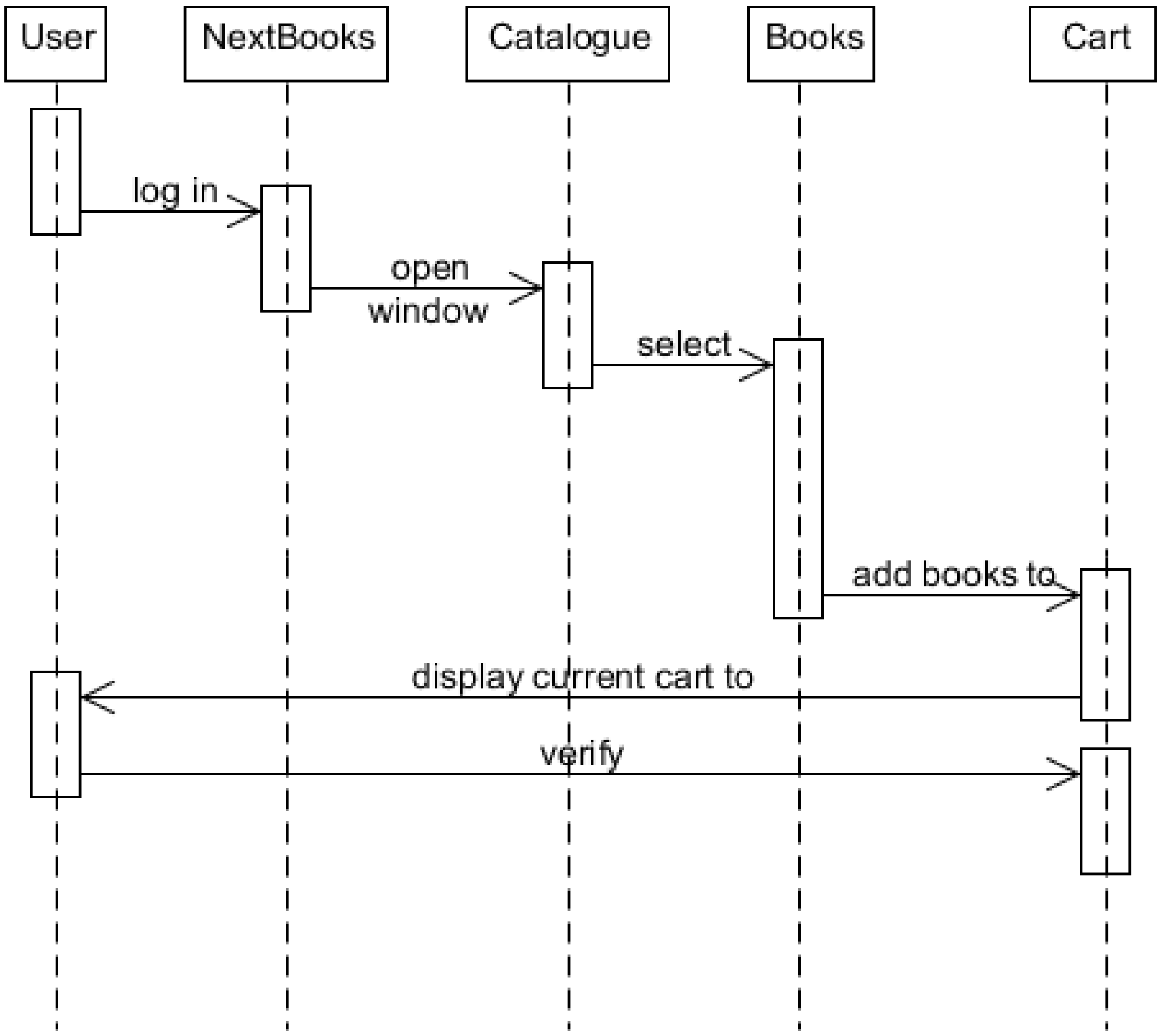
Reading:



Searching


User
NextBooks
Catalogue
Books
log in
open window
search by isbn, or author, or name
give list of matching books

Cart Use:



2.8 Operation Contracts

1. *Operation:* (isbn: integer, author: string, name: string)
Cross References: use cases: Searching
Preconditions:
-At least 1 of isbn, author, name is passed a non-null value
-Search instance exists
Postconditions:
-A List instance "results" is created
-results associated with current Search
-Database is queried for items matching 1 or more of {isbn, author, name}
-Items returned from database query are placed in results

2. ***Operation:*** *showResults(results: List<eBooks>)*
Cross References: use cases: Catalog Searching
Preconditions:
-List<eBooks> instance results exists
-Search Instance exists
Postconditions:
-A Window instance ResultsView is created
-ResultsView associated with current Search
-results is iterated through and each item is placed in ResultsView
-ResultsView is made the active Window

3. ***Operation:*** **openEbook(book: eBook, user: User)**
Cross References: use cases: Book Reading
Preconditions:
-eBook instance book exists
-eBook instance book is "owned" by current User
-User instance user exists
Postconditions:
-An eBookReader instance "reader" was created
-reader associated with user
-book is loaded into reader

4. ***Operation:*** **addBookstoCart(books: List<eBooks>)**
Cross References: use cases: Cart Usage
Preconditions:
-Cart instance c exists
-List<eBooks> instance books exists
Postconditions:
-books is iterated through and each item is placed in current Cart

5. ***Operation:*** **userLogin(username: string, password: string)**
Cross References: use cases: Login
Preconditions:
-Internet Connection available
Postconditions:
-Database is queried for User entries matching both {username, password}
-Current User set to User result from database query
-If query finds no matches Current User is set to null

6. ***Operation:*** **payForBooks(cart: Cart, user: User)**
Cross References: use cases: Rental
Preconditions:
-Cart instance cart exists
-User instance user exists
Postconditions:
-A Sale instance s was created
-Sale was associated with User

-sale_total in s was initialized
-cart is iterated through and sale_total in s is incremented by the cost
field of each book in cart
-cart is iterated through and each book is added to (List)owned_books in user
-sale_total is charged to creditCardNumber in user
-cart.paid set to true

7. ***Operation:*** **createReceipt(cart: Cart)**
Cross References: use cases: Rental
Preconditions:
-Cart instance cart exists
-Cart.paid is true
Postconditions:
-A Window instance ReceiptVeiw is created
-ReceiptVeiw ascociated with cart
-ReceiptView contains information that was in cart
-ReceiptVeiw is made the active Window

8. ***Operation:*** **showUserRentals(user: User)**
Cross References: use cases: Rental, Profile View
Preconditions:
-User instance user exists
Postconditions:
-A Window instance myBooks is created
-myBooks associated with user
-user.owned_books is iterated through and each book is added to myBooks
-myBooks is made the active Window

2.9 Obtaining User Feedback

We simulated user feedback by asking other members of the class to observe and utilize our system. We gathered their feedback by observing their use of the system and by asking for them for their thoughts on usability afterwards. Since our functionality was still being implemented, we focused on usability and interface design elements in obtaining our user feedback.

3 Updated Design And Testing

3.1 System Operations

Boolean checkLogin(username,password)
Returns true if the user exists and the passwords match, false otherwise.

LinkedList<Book> search{ISBN, Author, Name}(term)
Returns a LinkedList of Books matching the search criteria based on the field.

LinkedList<Book> getCart()
Returns a LinkedList of Books that were in the users cart.

addBookToCart(cart, book)
Adds an item to the cart.

openCart(cart)
Displays the cart.

removeFromCart(cart, book)
Removes a book from the cart.

checkout(cart)
Takes user to a checkout page, with total price and items in cart.

runReader(url)
Opens the PDF reader window and reads the pdf at the given URL.

addBookToCatalogue(book)
Adds a book to the catalogue.

Book getBookByName(name)
Returns a book matching the searched name.

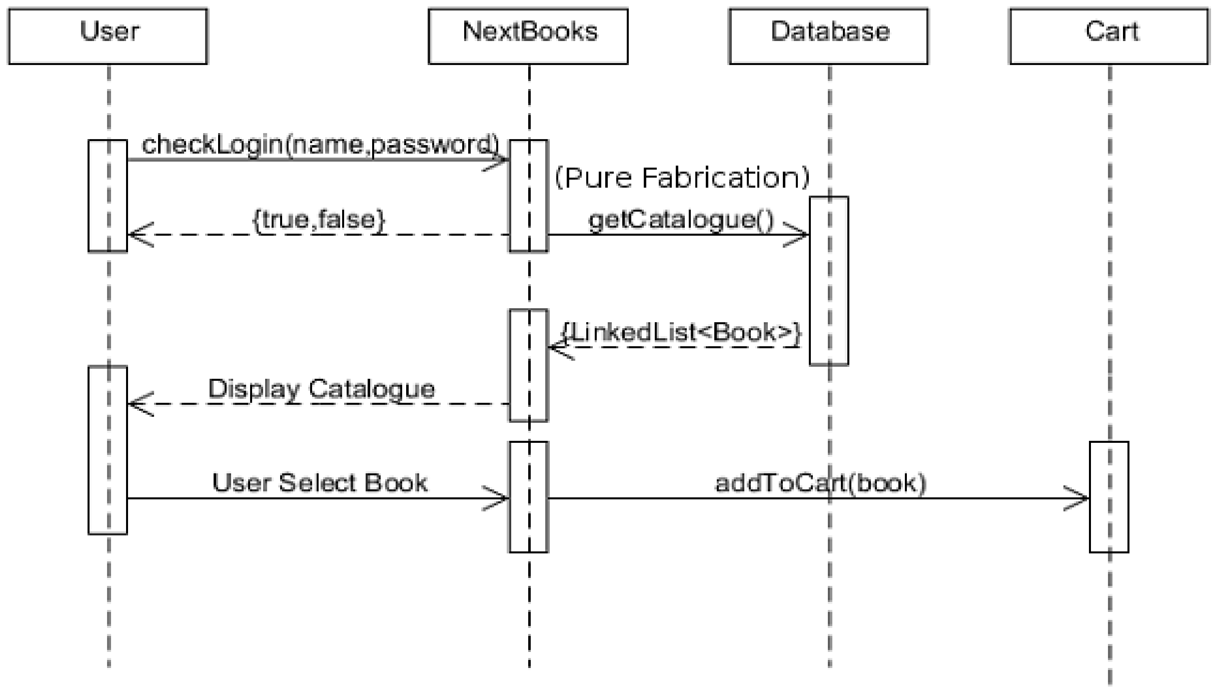
Book getBookByAuthor(author)
Returns a book matching the searched author.

Book getBookByTitle(title)
Returns a book matching the searched title.

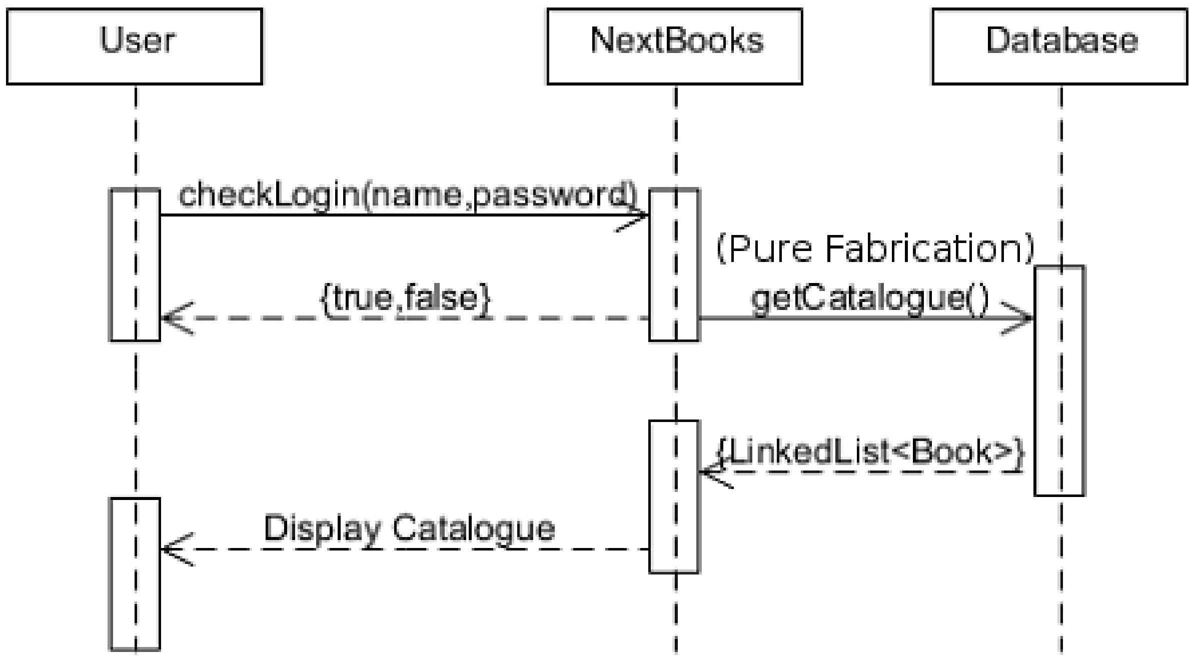
LinkedList<Book> getCatalogue()
Returns all the books in the catalogue.

3.2 GRASP Sequence Diagrams

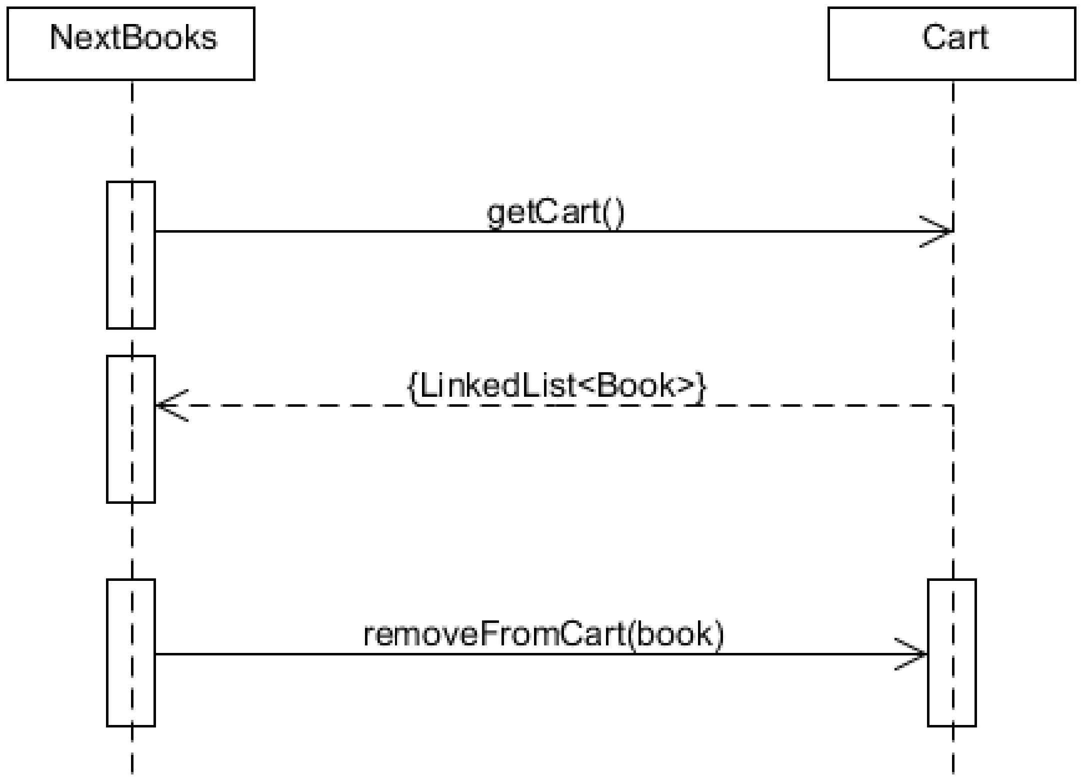
3.3 Viewing Catalogue



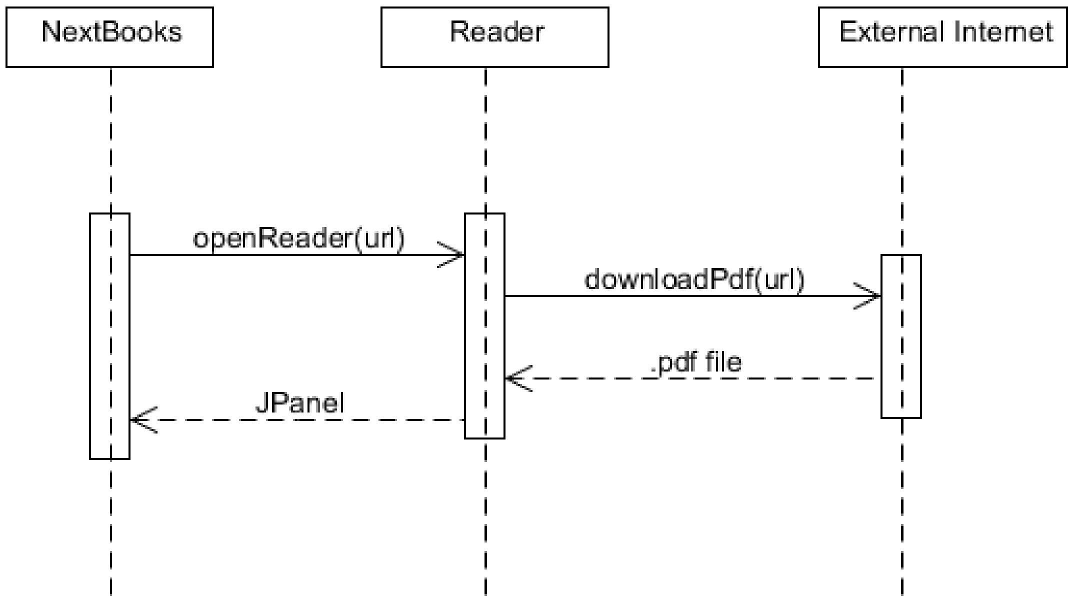
Adding To Cart:



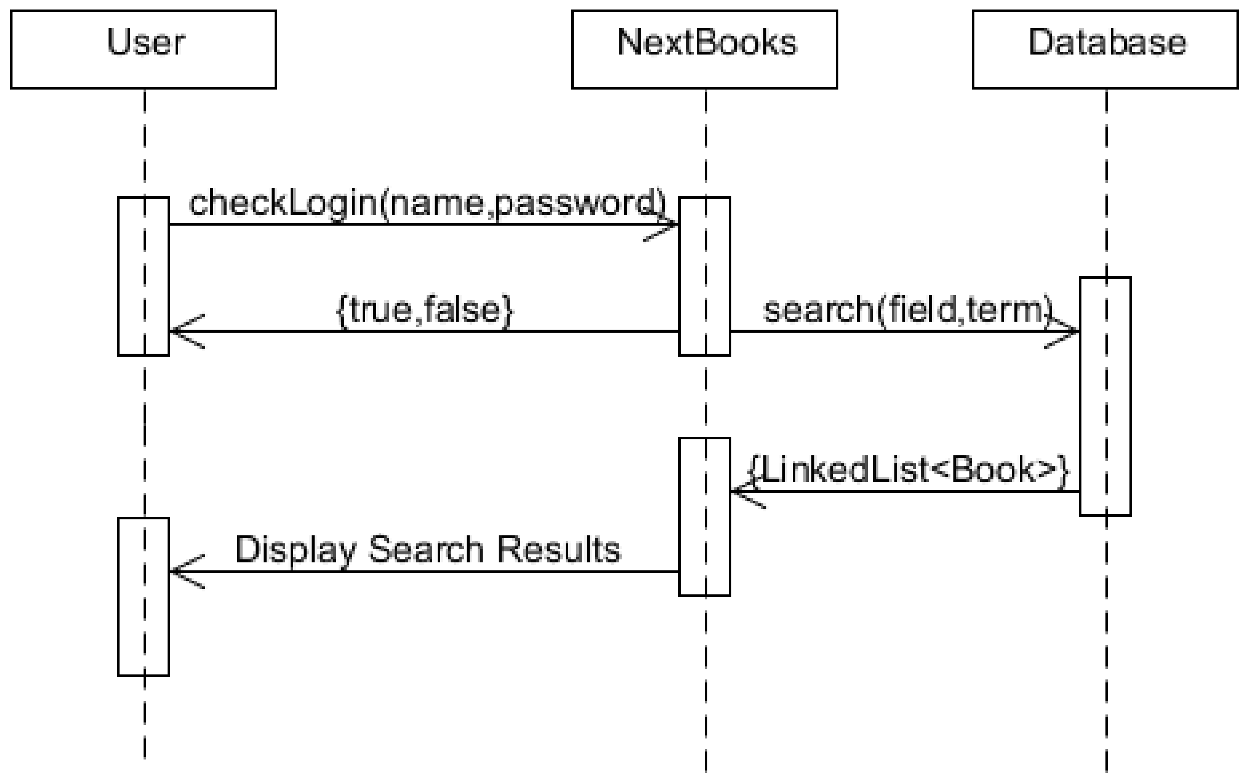
Remove From Cart



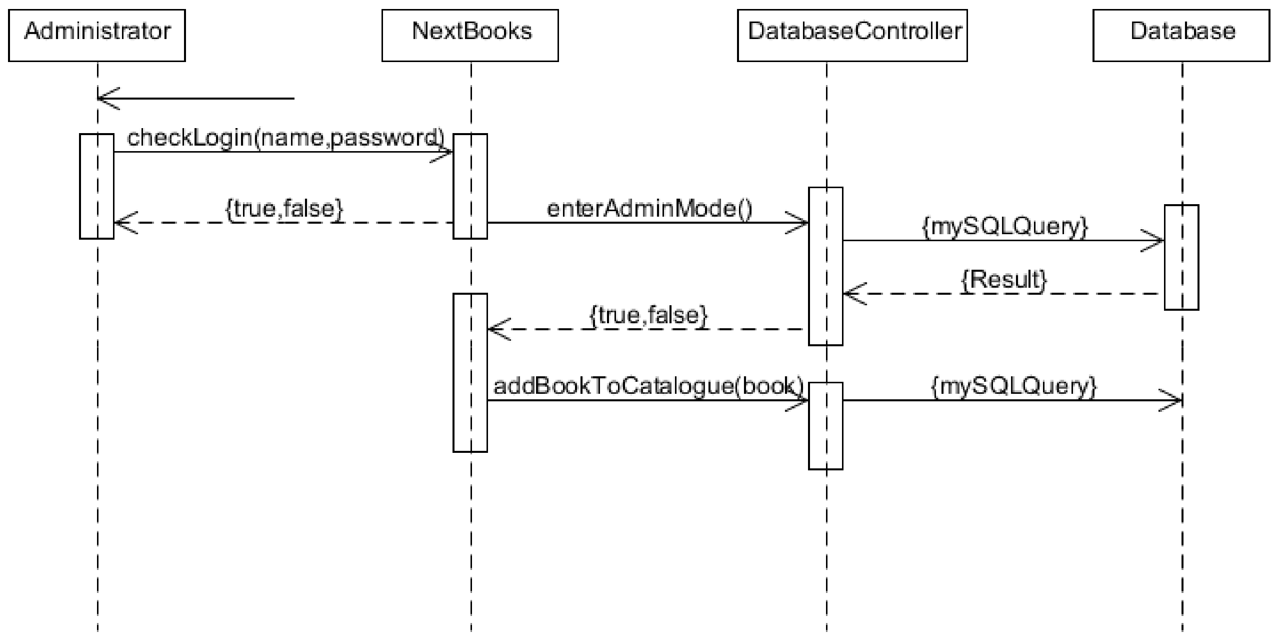
Reading .PDF



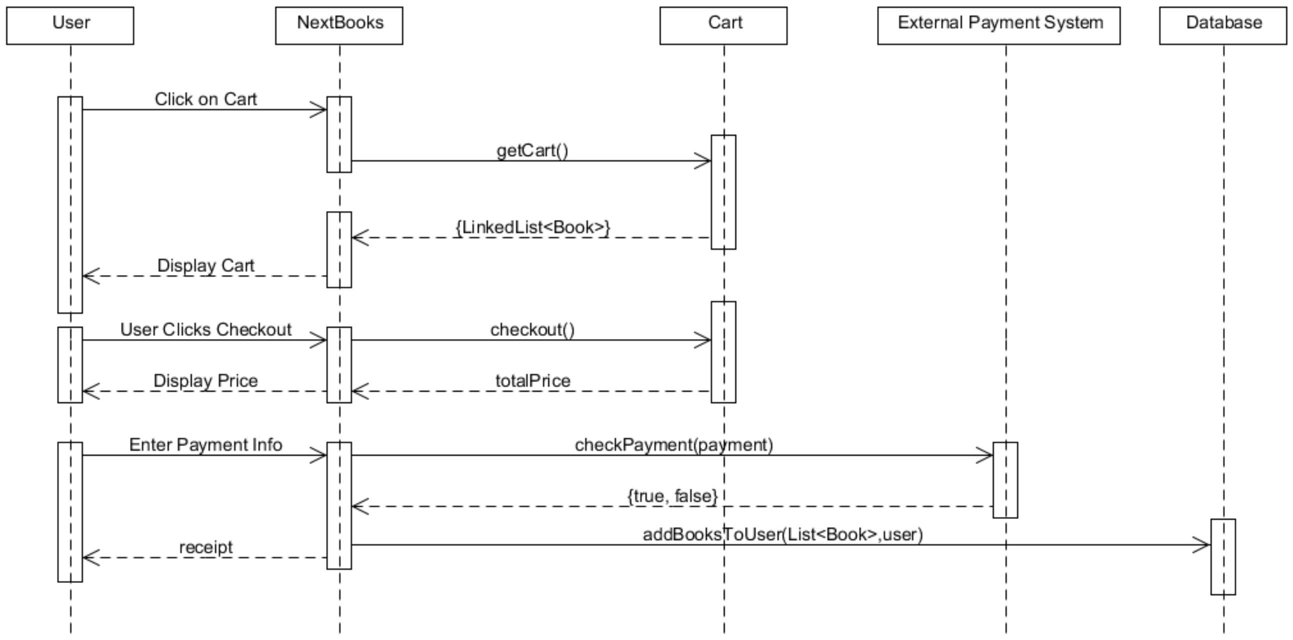
Searching



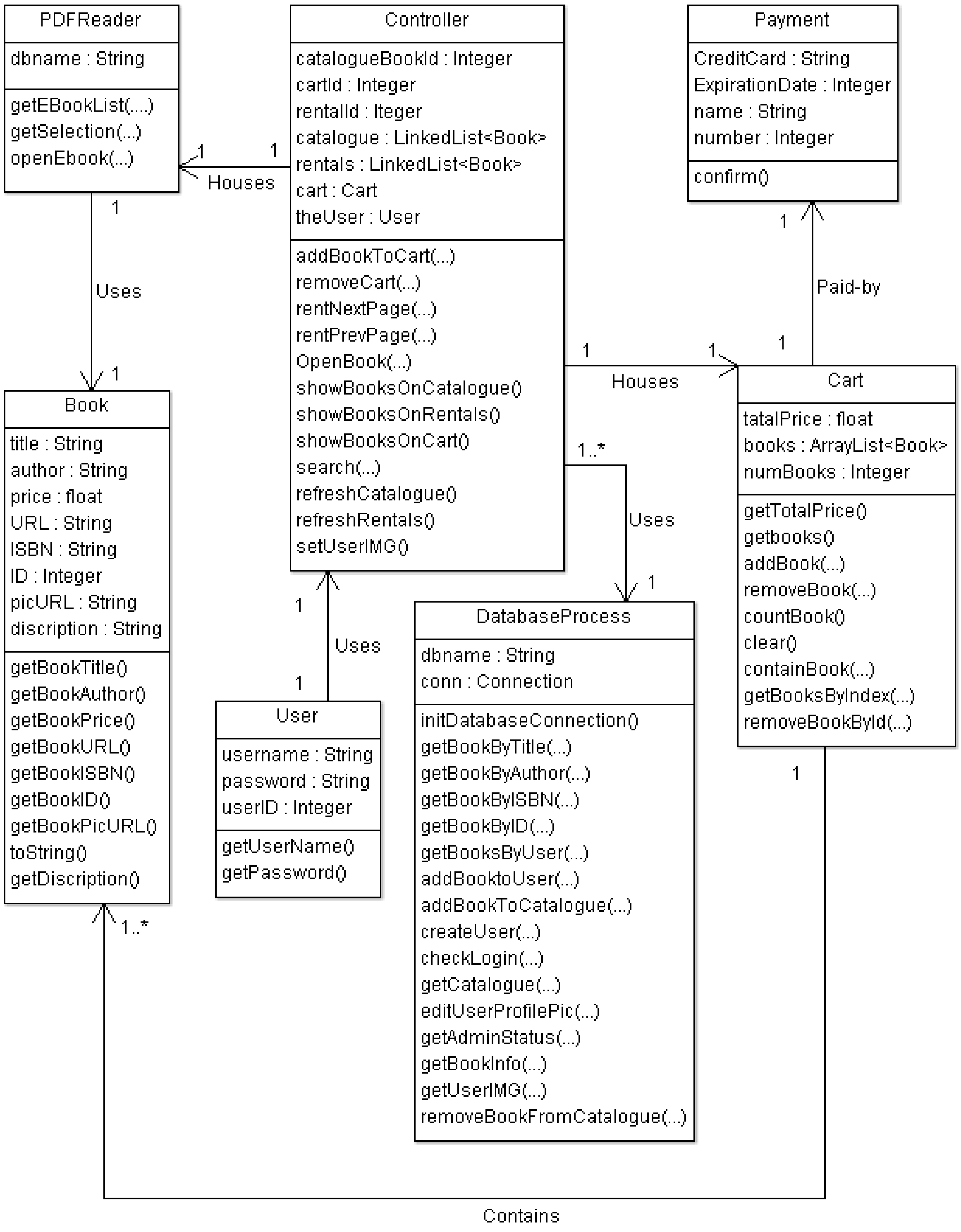
Administrator Adding Books



Payment


User
NextBooks
Cart
External Payment System
Database
Click on Cart
getCart()
{LinkedList<Book>}
Display Cart
User Clicks Checkout
checkout()
Display Price
totalPrice
Enter Payment Info
checkPayment(payment)
{true, false}
addBooksToUser(List<Book>,user)
receipt

3.4 Class Diagram



3.5 Unit Testing

We used JUnit testing to ensure the success of our DatabaseController methods. The source code of our tests is inside DatabaseProcessJUnitTest.java. We tested that creating a user was possible, creating an admin was possible and we tested each method of searching (ISBN, author, title). We've also tested adding books to the user and adding books to the catalogue. JUnit was very helpful for us to ensure the success of our system.

4 Reengineering

Authors Note: Jiayuan took on the role as lead developer of our front end system. He has since dropped the course and moved back to China. Since his code is fairly dirty, this has created problems with our re-engineering of the front end. Thus, we are focusing our refactoring and reengineering on the backend,database interface.

4.1 Code Smells

Smelly Controller

Our controller class is bloated. It suffers from Low Cohesion because it has way too much functionality. It functions as a GUI class and as a GRASP Controller.
We should have moved most of the GUI elements to the Nextbook Class and then just used the Controller class as a controller.
It is out of our time budget to accomplish this now.

Clones

Auto-Generated Clones
Lines 101-132 of newUser
Lines 64-95 of Receipt
Lines 163-194 of Nextbook
Lines 107 – 138 of Pay
Lines 77 – 108 of Confirm
These are clones due to Development Strategy. These code fragments were generated by NetBeans GUI design tools. We are going to leave them in due to Lack of Ownship, as the GUI code was done by Jiayuan who is no longer with us.

Database Controller Clones
Overall we are happy with the Database Controller class. It was written by Colin and we think he did a pretty good job. The clones in Database Controller are all necessary loops that simply iterate through the results from the database query.

Book Constructor Clone
Lines 21-31, 37-46 of Book
At first we didn't have "ids" for each book, then we added in an id field for purposes of database storage, we added a new constructor as well. The constuctor's both have different purposes. The DB creates books with Ids, (real books in DB), but the GUI creates books just for purposes of displaying them, and they have no Ids. We could refactor this by having the smaller constructor call the larger one.

Controller Class Clones
We have many clones in the controller class. There is two reasons for there existence. One is that we used Netbeans' integrated GUI designer to design the basics of our GUI. Second is we left refactoring of this system for too long. When we started adding in new

elements to our GUI we simply copy and pasted the old elements and altered the code to suit our needs. The reason our code is bad is because of:

showBooksOnRentals() *Lines 1273-1379*
showBooksOnCatalogue() *Lines 1126-1271*
showBooksOnCart() *Lines 1381 - 1497*

Each method refreshes the appropriate list (rentals, cart, catalogue), and then displays it in the page. They exist like this because of poor initial design. We started out with just a catalogue, and at the time it made sense to have a method to show the catalogue, but it wasn't very extendable to the future aspects of our system. We needed some kind of "tick" or "update" to keep our program refreshing. Instead we started with these 3 methods to refresh their individual pages. It's a design that we have learned from, but mostly we've learned what not to do.

It would be more appropriate to have this method replace all three:
showBooksOnList(NextBookList books)*

GUI Creation Clones
Class **Controller**, *InitComponents(), Lines 131-904*
There are many subblocks of clones inside of *InitComponents()*. This is due to a Re Use Strategy where we established one working component (eg, a button) and created many more in a similar way. There isn't too much refactoring that could be done here, as in Java GUIs you will always have many Panels, Buttons and Labels all with similar traits except for small changes.

Action Listener Clones
Lines 1014-1022, 1029-1037,1073-1080 , 1082-1090, 1093-1104
of Controller: Next and Prev Buttons
We have cloned action listeners for the "NEXT" and "PREV" buttons for the rentals page, catalogue page and the cart page. Each action listener simply increments (or decrements) an integer corresponding to the location we are looking in the rentals list, cart list or catalogue list.
*We could replace all these action listeners by refactoring a *LinkedList<Book>* to be a *NextBookList*, which will contain a *LinkedList<Book>* as well as an integer to know what the current item is.
Then showBooksOnList would become *showBooksOnList(NextBookList list)*
and each action listener would simply increment the ID in the current *NextBookList*.

Lines 544-551, 559-566 , 574-581 , 590-596 of Controller: Add To Cart Buttons
Lines 608-615, 623-630, 638-645, 653-660 of Controller: Remove From Cart Buttons
These clones are for the "add to cart" buttons on the catalogue page and the "remove from cart" buttons on the cart page. They could be killed by creating a new method which decides which book to add to the cart and which to remove from based on an integer passed in, but no matter how you do Action Listeners you always have at least a small clone. As it stands we simply directly tell the first button to add the first book and so on. This refactoring adds unnecessary complexity. We are fine with leaving these clones in.

4.2 Refactoring

Cart

Removed method “getBooks”. This method was not being used. *Lines: N/A*
Removed field “id”. This field wasn't being used. *Line: 12*

User

Removed “userID” field. This field was not actually required. The ID is stored in the DB. *Line: 6*

CartProcess

This class has been removed. It was supposed to be a controller for the cart, but was unnecessary. *Lines: N/A*

DatabaseController

This class has become a *Singleton*. It made sense to be a *Singleton* because it doesn't have any instance specific memory. We added a getInstance() method and updated every reference to DatabaseController. *Lines: 28-36*

Book

The constructor without Ids now calls the constructor with an ID of 0. *Lines 23-25*

4.3 Gang of Four Design Patterns

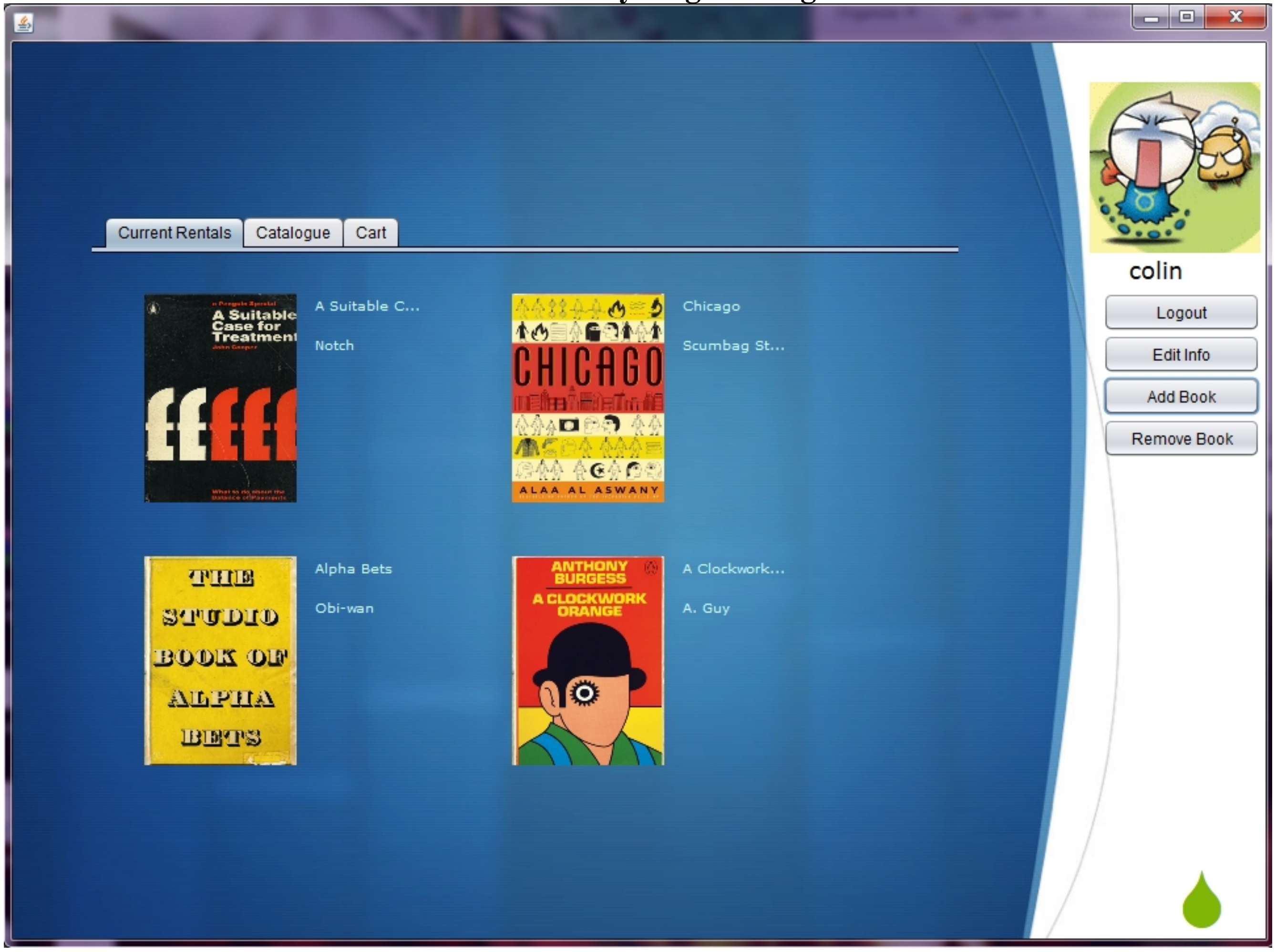
DatabaseController

This class is a *Singleton*, *Pure-Fabrication Controller* which we use to interface with our database. It is extremely easy to use because of these design patterns. It is easy for a client to instantly have access to it because of it's *Pure-Fabrication* attributes. It is efficient in memory usage because it is a *Singleton*, and it hides all the database knowledge from the client with it's *Controller* traits.

Controller

This class was meant to serve as simply a controller to control the GUI. However it has since become bloated and serves as both a controller and as the GUI itself.

5.4 Usability Engineering



Nextbooks' main screen contains your Current Rentals, Catalogue and Cart Administrator functions: Add and Remove Book are seen on the right.

Some systems force users to delve through multiple windows and buttons, but Nextbooks has all it's functionality conveniently located all on one page. This lowers *memorization*. The Catalogue Tab, Cart Tab and Rentals Tab all have very similar button placements, picture placements and functionality, which appeases c*onsistency*. The remaining tasks (Logout, Edit [User] Info, Add Book and Remove Book], are pictured below the user's name and profile picture on the right side of the window.

6 Project Plan, Budget Justification and Performance Evaluation

List of Tasks			Group Member	Comments
	Abstract		Jason(jwk881)	
1.	Introduction		Jason	
2.	Requirements and Early Design			
	2.1	Summary Use Cases	50% Jason/ Luke(ljr458)	
	2.2	Fully-dressed Use Cases	Jason	
	2.3	Use Case Diagram	Colin(cml220)	
	2.4	Domain Model	Luke	
	2.5	Glossary	Jason	
	2.6	Supplementary Specification	Jason	
	2.7	System Sequence Diagrams	Luke	
	2.8	Operation Contracts	Luke	
	2.9	Obtaining User Feedback	Luke	
3.	Updated Design and Testing			
	3.1	System Operations	Luke	
	3.2	Sequence or Communication Diagrams with GRASP Patterns	Luke/Jason	
	3.3	Class Diagram	Luke	
	3.4	Unit Testing	Colin	Colin tested DatabaseController
4.	Reengineering			
	4.1	Code Smells	50% Jason 50% Colin	Colin fixed alot of problems
	4.2	Refactoring	75% Colin 25% Jason	Jason took over Jiayuan's early code
	4.3	Gang of Four Design Patterns	50% Colin 25% Jason 25% Jiayuan (jih748)	
5.	Complete Implementation and Product Delivery			
	5.1	Naming Conventions	75% Jason 25% Colin	
	5.2	Commenting	80% Colin 20% Jason	
	5.3	Pretty-printing of the source	Jason	
	5.4	Usability Engineering	50% Jason/	

			Luke	
	5.5	**Complete Implementation**	**30% Jiayuan 30% Jason 40% Colin**	**Jiayuan/Jason front end Colin backend**
	5.6	**User Manual**	**Colin**	
	6.	**Project plan, Budget Justification and Performance Evaluation**	**Jason**	
	7.	**Conclusion**	**Jason**	
		Acknowledgements	**Jason**	
		References	**Jason**	
		Total % Contribution	**Jason 35% Colin 35% Luke 20% Jiayuan 10%**	

Group Member	**Tasks Responsible For**	**% Contributions if not done alone, and then say who helped and how much**	**Comments**
jwk881/Knight	Delegation of work & Project Planning	90%, 10% other members	
	Took over Jiayuans front end GUI code after he left to China	Did about 60% of the work, then Colin did the other 40%	
	Documentation	Did ~75% of all the documentation, Luke and Colin did the rest	
	Total for Knight	30 hours	

cml220/Larson	Database Interface	Did all of DatabaseController class	
	Admin Functionlity	Added the admin functionality	
	PDF Reader	Got the PDF reader and showed us how to use it	
	Misc Programming	Did about 40% of the functionality that was added after Jiayuan left	
	Total for Larson	30 Hours	

jih748/Hu	GUI Design	Created the GUI for the first milestone, then tried to add more functionality but left for China	
	Total for Hu	20 Hours	
ljr458/Robinson	Diagrams	Most of the diagrams	
	Documentation	50% of the documentation	
	Code	Helped Debug	
	Total for Robinson	25 Hours	
	Total hours	105 Hours	

Comparison To Budget Estimates

Overall we feel like we had a fairly accurate estimate of the time required to actually implement the functionality. It took about 2 hours for Jiayuan to do the initial GUI design, and for Colin to create the initial Database. We had similar accuracy for our Catalogue Prototype and our Rentals (profile) prototype. We did not account for the time to create the documentation in our initial estimates, and this took about 40% of the total time.

Notable failures are how ugly our code is. We know that our code is ugly but we have learned from it. Excuses are inexcusable, but Jiayuan leaving for China halfway through the project was a huge setback. Up until that point no one but him had worked on the GUI, and it forced the other members (mostly Colin and Jason) to decipher Jiayuans code. His code was created with the Netbeans GUI Designer, so it was barely even readable. It is in a better state now than it was then, but it still suffers from Low Cohesion and High Coupling.

7. Conclusion

Nextbook achieved all the crucial functionality that we wanted it to without our budgeted time. We had to cut out Page Notes as we were unable to deduce it in time, but we knew that such a decision was possible. Our group, The League, worked well together. Jason would typically break down the milestone into work units for each member to do. Jiayuan worked on the GUI and created the look and feel we see today. Colin did a great job on the DatabaseController and helped a ton with the coding. Luke was available to help with the documentation. We suffered a significant setback when Jiayuan left for China. His code was hard to work with because we did not pair program enough while he was still around. We had to start refactoring his code partway through implementation, and the code could still use some more refactoring. We feel like it was our GUI design that really set us apart from the other groups. We got the second most votes for “best project” and we feel like it was because of the good look of our system.

Acknowledgements

Oracle for making many nice Java Libraries
http://mysite.science.uottawa.ca/rsmith43/Zombies.pdf is the temporary .PDF we are linking to.

References

[1] PDF Renderer: http://java.net/projects/pdf-renderer/sources/svn/show/trunk/src/com/sun/pdfview?rev=140