CSCI-5448 00AD

BOOKWISE - PROJECT PART 2 EXTRA CREDIT

Name: Aayush Grover

Title: BOOKWISE (An android app for exchanging and selling textbooks)

Project Summary:

An Android App where students can post their textbooks for sale; users(students,buyers) would be able to search for a book, buy it or exchange it for a book that the other user(seller) requires. Books will categorized for different subjects and will have information of condition, edition, picture of cover page and location. All data will stored in a database.

USE CASE CONSIDERED: RESERVE BOOKS

Full Use Case List to complete the considered use-case:

Use Case ID	Primary Actor	Use Cases
UC-1.1.1	General User	Register
UC-1.1.2	General User	Sign In
UC-1.1.3	General User	Provide valid email address
UC-1.1.4	General User	Create Valid Password
UC-1.1.5	General User	Provide Valid Password
UC-1.1.6	General User	Provide contact information
UC-1.2.1	Buyer	Search For Books
UC-1.2.2	Buyer,Seller	Reserve Books

Use case

Use Case ID:	UC-1.1.1		
Use Case Name:	Register		
Created By:	Aayush Grover	**Last Updated By:**	Aayush Grover
Date Created:	Nov 6, 2015	**Last Revision Date:**	Nov 6, 2015
Actors:	General User (Any User)		
Description:	Allows a user to register for a new account.		
Frequency of Use:	Once per User		
Preconditions:	User has not signed up yet.		
Postconditions:	User has signed up and holds a valid account.		
Normal Flow:	i. User visit the register page. ii. User provide valid @colorado.edu E-mail address iii. User create valid password (at least 8 digits) iv. Provide contact information (Mandatory/optional) v. User click on register button		
Exceptions:	i. E-mail address is invalid ii. Password is invalid iii. Mandatory contact information not provided iv. Account Already exist		
Includes:	Provide Valid Email Address Create Valid Password Provide Contact		
Special Requirements:	None		
Assumptions:	User has internet access		
Notes and Issues:	None		

Use Case ID:	UC-1.1.2		
Use Case Name:	Sign In		
Created By:	Aayush Grover	**Last Updated By:**	Aayush Grover
Date Created:	Nov 6, 2015	**Last Revision Date:**	Nov 6, 2015
Actors:	General User (Any User)		
Description:	This use case allows a user to sign in with his/her account		
Frequency of Use:	Every time switch account or sign in to a new device		
Preconditions:	User was logged out		

Postconditions:	User is signed in
Normal Flow:	i. User visit the sign in page ii. User provide valid @colorado.edu E-mail address iii. User provide valid password iv. User click on sign in button
Exceptions:	i. E-mail address is invalid ii. Password is invalid
Includes:	Provide Valid E-mail Address Provide Valid
Special Requirements:	None
Assumptions:	User has internet access
Notes and Issues:	None

Use Case ID:	UC-1.1.3		
Use Case Name:	Provide Valid E-mail Address		
Created By:	Aayush Grover	**Last Updated By:**	Aayush Grover
Date Created:	Nov 6, 2015	**Last Revision Date:**	Nov 6, 2015
Actors:	General User (Any User)		
Description:	This use case allow a user to provide a valid E-mail address		
Frequency of Use:	Every time user wants to register or sign in		
Preconditions:	Email address not provided		
Postconditions:	Email address provided		
Normal Flow:	i. User click on E-mail address input field ii. User type in E-mail address		
Exceptions:	None		
Includes:	None		
Special Requirements:	None		
Assumptions:	User has internet access		
Notes and Issues:	None		

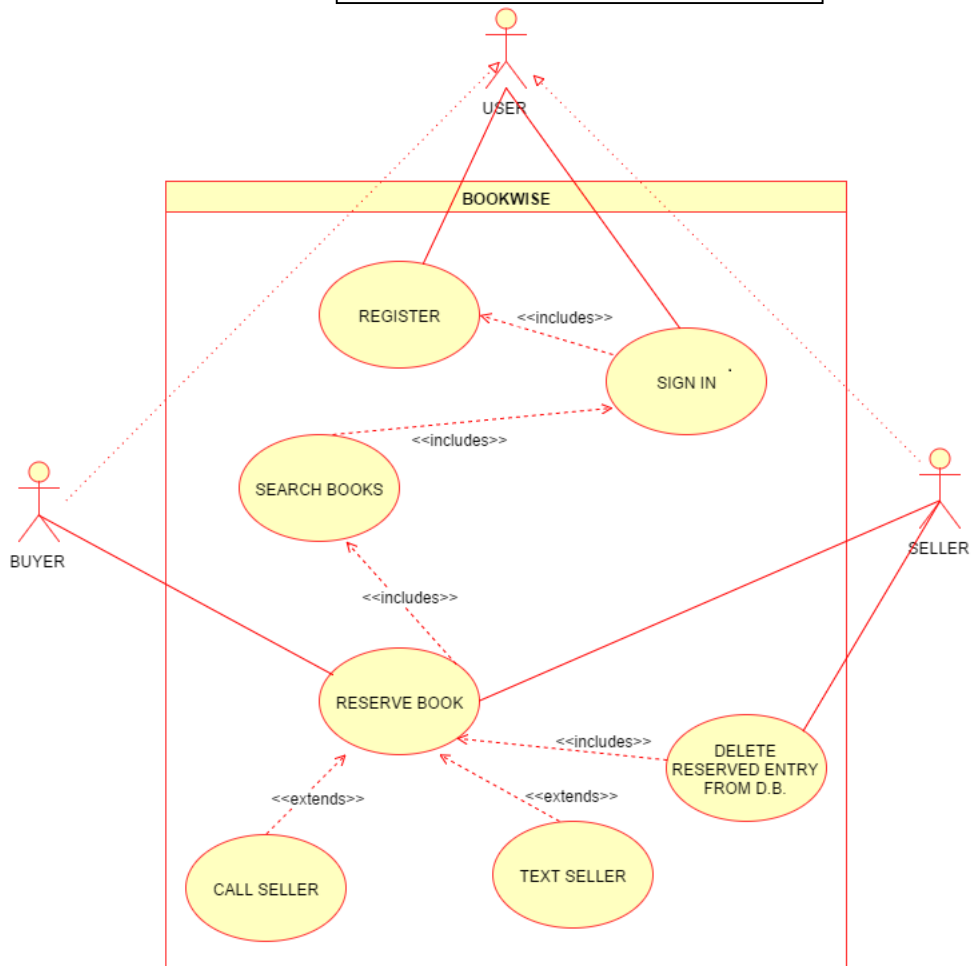|	
Use Case ID:	UC-1.1.4
Use Case Name:	Create Valid Password
Created By: Aayush Grover	**Last Updated By:** Aayush Grover
Date Created: Nov 6, 2015	**Last Revision Date:** Nov 6, 2015
Actors:	General User (Any User)
Description:	This use case allow a user to create a valid password
Frequency of Use:	Every time user wants to register
Preconditions:	Password not exist
Postconditions:	Password created
Normal Flow:	i. User click on the new password input field ii. User type in a new password
Exceptions:	None
Includes:	None
Special Requirements:	None
Assumptions:	User has internet access
Notes and Issues:	None
Use Case ID:	UC-1.1.5
Use Case Name:	Provide Valid Password
Created By: Aayush Grover	**Last Updated By:** Aayush Grover
Date Created: Nov 6, 2015	**Last Revision Date:** Nov 6, 2015
Actors:	General User (Any User)
Description:	This use case allow a user to provide a valid password
Frequency of Use:	Every time user wants to sign in
Preconditions:	Password not provided
Postconditions:	Password provided
Normal Flow:	i. User click on the password input field ii. User type in the password

Exceptions:	None	
Includes:	None	
Special Requirements:	None	
Assumptions:	User has internet access	
Notes and Issues:	None	

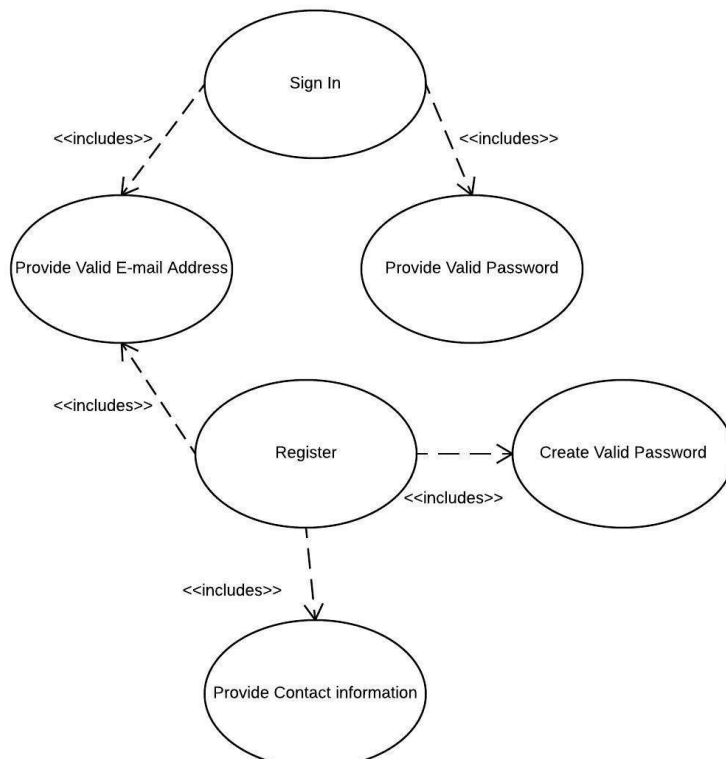|			
Use Case ID:	UC-1.1.6		
Use Case Name:	Provide Contact Information		
Created By:	Aayush Grover	**Last Updated By:**	Aayush Grover
Date Created:	Nov 6, 2015	**Last Revision Date:**	Nov 6, 2015
Actors:	General User (Any User)		
Description:	This use case allow a user to provide his/her contact information		
Frequency of Use:	Every time user wants to register		
Preconditions:	Contact information (mandatory/optional) not provided		
Postconditions:	Contact information (mandatory/optional) provided		
Normal Flow:	i. User click on the contact information input field ii. User type in the contact information		

Use Case ID:	UC-1.2.1		
Use Case Name:	Search Books		
Created By:	Aayush Grover	Last Updated By:	Aayush Grover
Date Created:	Nov 6, 2015	Last Revision Date:	Nov 6, 2015
Actors:	Buyer		
Description:	This use case allow a Buyer to search for book he/she wants to buy		
Frequency of Use:	Every time Buyer wants to buy a book		
Preconditions:	Buyer was on search page		
Postconditions:	Buyer has a list of results		
Normal Flow:	i. Buyer visit the search page ii. Buyer provide title/author/edition iii. Buyer select sort with distance/price option iv. Buyer click on search		
Exceptions:	None		
Includes:	User has signed-in.		
Special Requirements:	None		
Assumptions:	User has internet access		
Notes and Issues:	None		

Use Case ID:	UC-1.2.2		
Use Case Name:	Reserve Books		
Created By:	Aayush Grover	**Last Updated By:**	Aayush Grover
Date Created:	Nov 6, 2015	**Last Revision Date:**	Nov 6, 2015
Actors:	Buyer, Seller		
Description:	This use case allows a buyer to reserve a book he/she wants		
Frequency of Use:	Every time a buyer finds a book, he/she wants to buy		
Preconditions:	Buyer is done with searching the books and has a list of results, but does not have reservation on any book.		
Postconditions:	Buyer has reserved the book and the seller has been notified.		
Normal Flow:	1. Buyer clicks on the reserve button. 2. Seller gets notified of the reservation. 3. Corresponding book entry gets deleted from the database.		
Exceptions:	None		
Includes:	User has performed the Search operation		
Extends:	CALL SELLER TEXT SELLER		
Special Requirements:	None		
Assumptions:	User has internet access		
Notes and Issues:	None		

USE CASE DIAGRAM:

Sub-Diagram

ACTIVITY DIAGRAM

RESERVE BOOKS

BUYER	BOOKWISE	SELLER

START

SHOW BOOKS POSTED BY ALL SELLERS

SELECT A PARTICULAR BOOK

YES

NO

END

NOTIFY SELLER AND RESERVE THE BOOK

TRANSACTION SUCCESSFUL?

YES

NO

DELETE POST

END

END
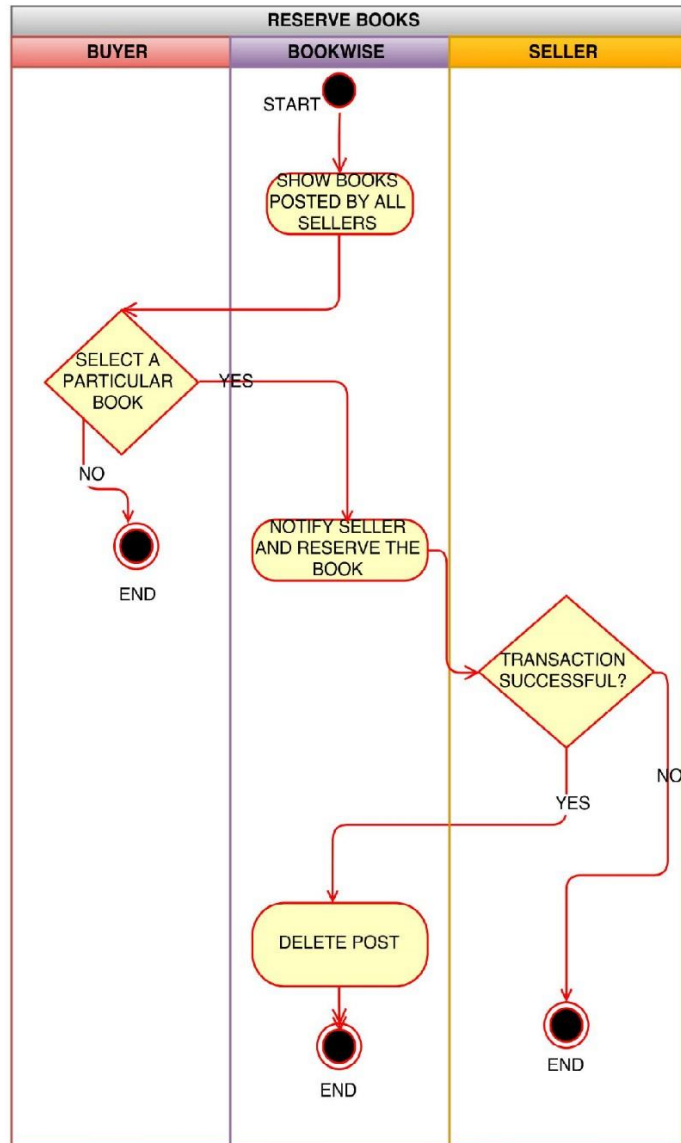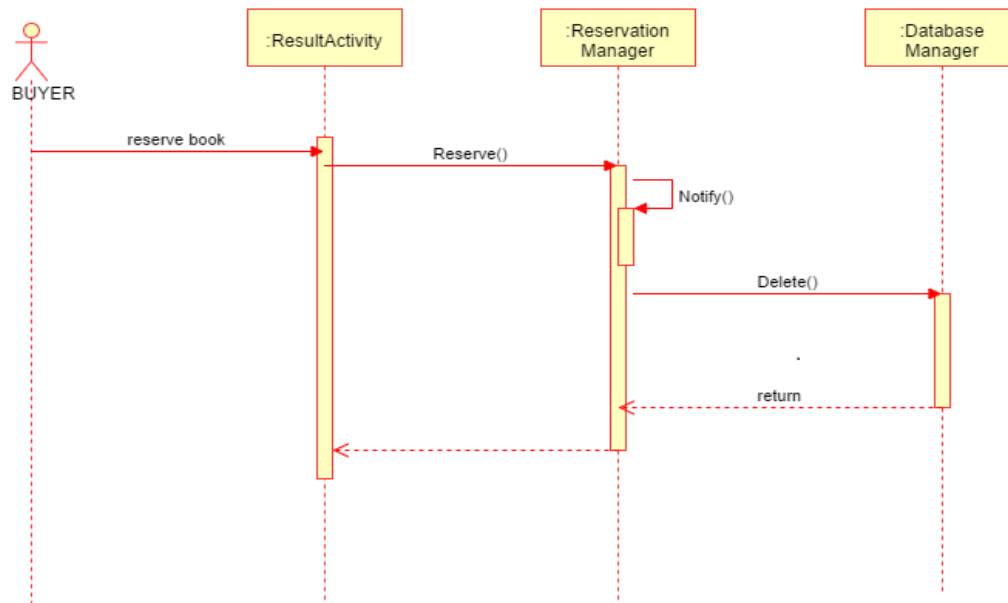
UI MOCKUPS:
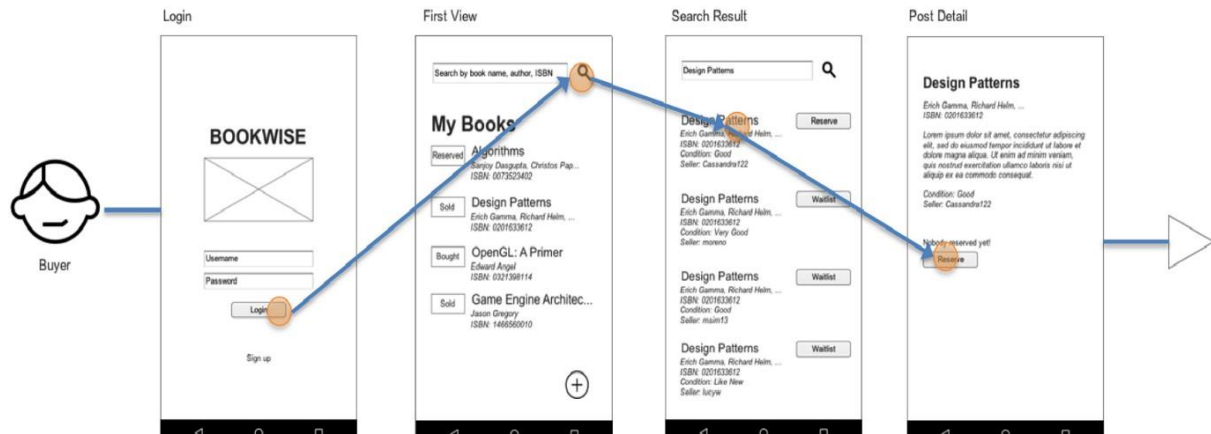
SEQUENCE DIAGRAM FOR RESERVING BOOKS:

CLASS DIAGRAM

android.app::Activity

#onCreate(state:Bundle) : void
#onResume() : void
+findViewById(id:int) : View
+getIntent() : Intent
#onPause() : void

Reservation Manager

-title : String
-db : DatabaseManager

+ Reserve(title : String):boolean
+Notify() : void

ResultActivity

-resultlayout : RelativeLayout
-resultlist : ListView
-alert : AlertDialog
-alertOK : Button
-alertCancel : Button
-rm : ReservationManager

#onCreate(SavedInstanceState : Bundle) : void
#onResume() : void
#onPaunse() : void
+showAlertDialog() : void
+onClick(dialog : DialogInterface, which : int) : void
+onItemClick(parent : AdapterView<?>, view:View,position : int,id:long) : void

DatabaseManager

-ctx : Context
-reg_URL:String
-login_URL:String
-bookadd_URL:String
-bookSearch_URL:String
-url:URL
-httpurlconnection: HttpURLConnection
-bufferedWriter : BufferedWriter
-data: String
-IS : InputStream
-line: String
-response : String
-alertdialog : AlertDialog

+DatabaseManager(ctx : Context)
#onPreExecute() : void
#doInBackground(...params : String) : String
#onProgressUpdate(...values : String) : void
#onPostExecute(result : String) : void
+Delete() : void

Extends

deletes
entry

1

1

1

1..*