kodisnowhere

SHELFBUTLER

Software Design Description

v1.1

18.01.2013

Changelog

*A: Added, M: Modified, D: Deleted

Version number	Date	Section	A,M,D*	Title or Brief Description
Version 1.0	02/12/12			Original.
Version 1.1	18/01/13	Section 2.0	M	Second paragraph is modified for clear specification.
		Section 3.1	M	Component diagram is updated.
		Section 3.2.1	A	A brief explanation is added to the diagram.
		Section 3.2.2	A	A brief explanation is added to the diagram.
		Section 3.2.3	A	A brief explanation is added to the diagram.
		Section 3.2.4	A	A brief explanation is added to the diagram.
		Section 3.2.4	M	Diagram is updated to fix a mistake.
		Section 3.2.5	A	A brief explanation is added to the diagram.
		Section 3.2.6	A	A brief explanation is added to the diagram.
		Section 3.2.7	A	A brief explanation is added to the diagram.
		Section 3.2.8	A	A brief explanation is added to the diagram.
		Section 3.2.9	A	A brief explanation is added to the diagram.
		Section 3.2.10	A	A brief explanation is added to the diagram.
		Section 3.2.11	A	A brief explanation is added to the diagram.
		Section 3.2.12	A	A brief explanation is added to the diagram.
		Section 3.2.13	A	A brief explanation is added to the diagram.
		Section 3.2.14	A	A brief explanation is added to the diagram.
		Section 3.2.15	A	A brief explanation is added to the diagram.

Version number	Date	Section	A,M,D*	Title or Brief Description
		Section 3.4	A	External interfaces section added.
		Section 3.4	A	External interface diagram is added.
		Section 4.1	M	Cardinality description is added to diagram.
		Section 4.1.1	M	Diagram explanation is aded.
		Section 5.16	A	Webservices are explained in detail.
		Section 7	M	Webservices are added to related use cases.

1 Introduction

1.1 Purpose

This software design document describes the architecture and system design of ShelfButler project. It gives a general development description for the code writing process. The intended audience of this document is developers and testers. The document is prepared according to the Software Requirement Specification document of the ShelfButler project. [1]

1.2 Document Scope

This document gives the design description for ShelfButler project. Detailed design descriptions and basic structure of the project are explained in order to be a guide for implementation.

1.3 Overview

First general description of the ShelfButler system including its functionality and matters related to the overall system and its design will be provided in System Overview section. This will provide the basis for the brief description of ShelfButler project.

After that, in the third chapter architectural design and decomposition description, and design rationale are explained. This section includes diagram of major subsystems are their interactions. UML sequence diagrams of the components are provided.

The fourth chapter includes Data Design section which gives information about data structures and data types used in ShelfButler project. This section includes ER diagram of the database, description of data storage, system entities with their attributes and relationships and diagram of the major classes.

In the fifth section, Component Design includes detailed descriptions. This section is the most important section of this document, because description of components and relationship between components are scrutinized in detail. Since general system will be constructed by connecting and mixing some components; details of components like attributes, methods, views will be explained in this section.

Sixth section, Human Interface Design includes overview of the user interface and screen images. It contains information about the functionality of the system from user perspective will take place.

Seventh section contains requirements matrix which maps software requirements in the Software Requirement Specification (SRS) [1] document with the matching components described. In the end, there is the conclusion section which will give a short brief and a general explanation.

1.4 Reference Material

[1] ShelfButler Software Specification Document, October 2012

2 System Overview

ShelfButler is a project that includes a mobile application and a web interface for the book collector users. The main goal of the project is to provide an environment for book collectors to manage their collections. It makes it easy to organize the items by keeping location of all books in the library, adding or removing books, searching books locally or on the internet, finding reviews/ratings, sharing opinions, getting recommendations about the books or recommending to someone else, reviewing and rating them and keeping track of the items that are lent to others. It also supports online shopping option for the books by redirecting the user to Amazon.com or Idefix.com. Furthermore, users will be able to save their favorites (as a favorites list), the books they want to read (as a readlist) and the books they wish to have (as a wishlist).

It is an easy to access and user-friendly application having a web interface in addition to mobile application, which makes it reachable from anywhere with an internet connection. Web interface will also provide all the functionalities that the mobile application has and moreover there will be a synchronize option for the synchronization between mobile application and the web interface.

The system will be an application with a graphical user interface and a database. After each operation as input, the changes shall reflect to the database in a way that will add, remove or edit the related fields, the system shall deal with the database issues i.e. searching or retrieving data from database, itself. It will be an independent project, only using some APIs with LGPL.

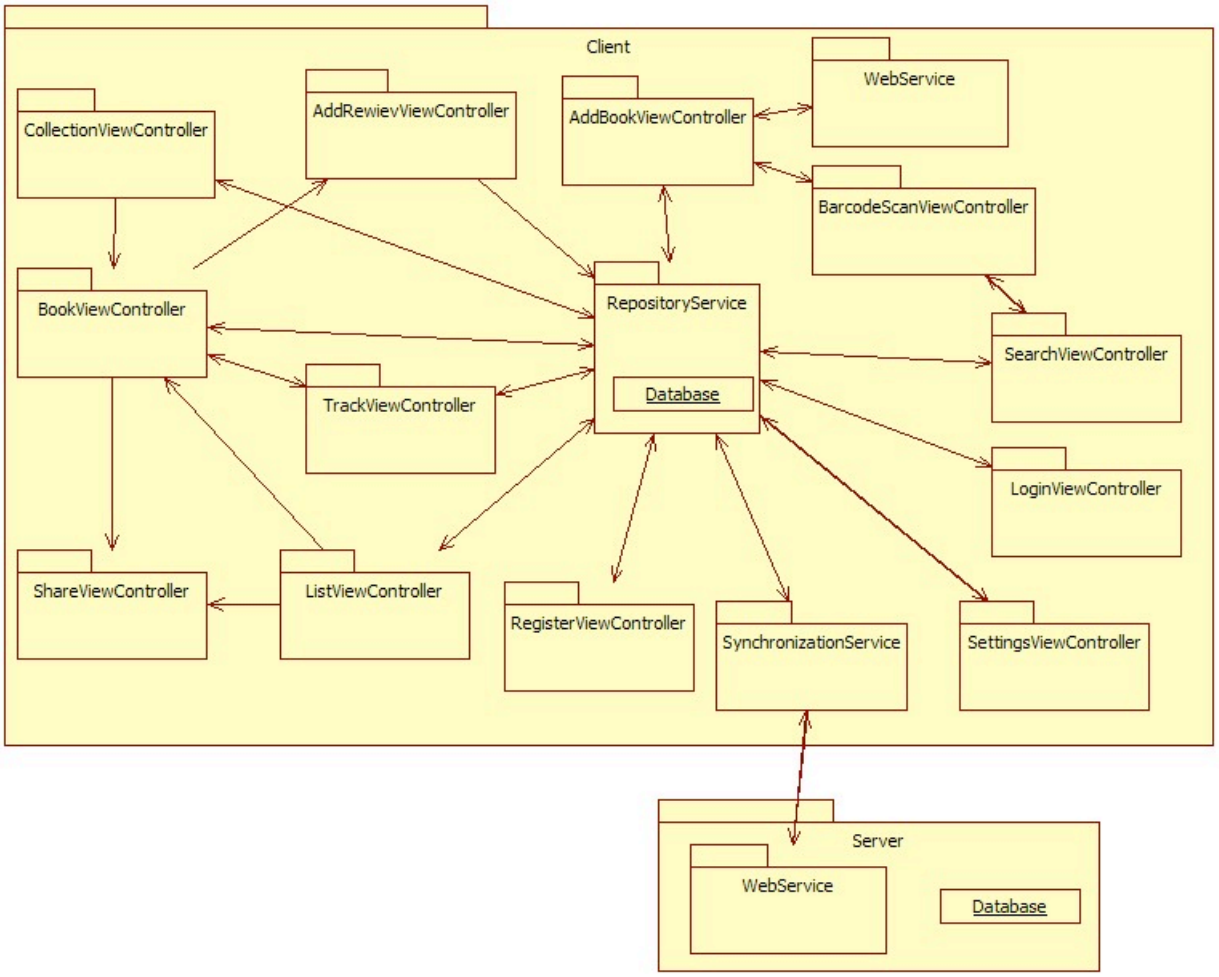
3 System Architecture

A general description of the software system architecture of ShelfButler is presented in the following sections.

3.1 Architectural Design

The architecture of Shelfbutler adopts model-view-controller pattern. The details and the reason why this pattern was chosen is explained in Section 3.3.

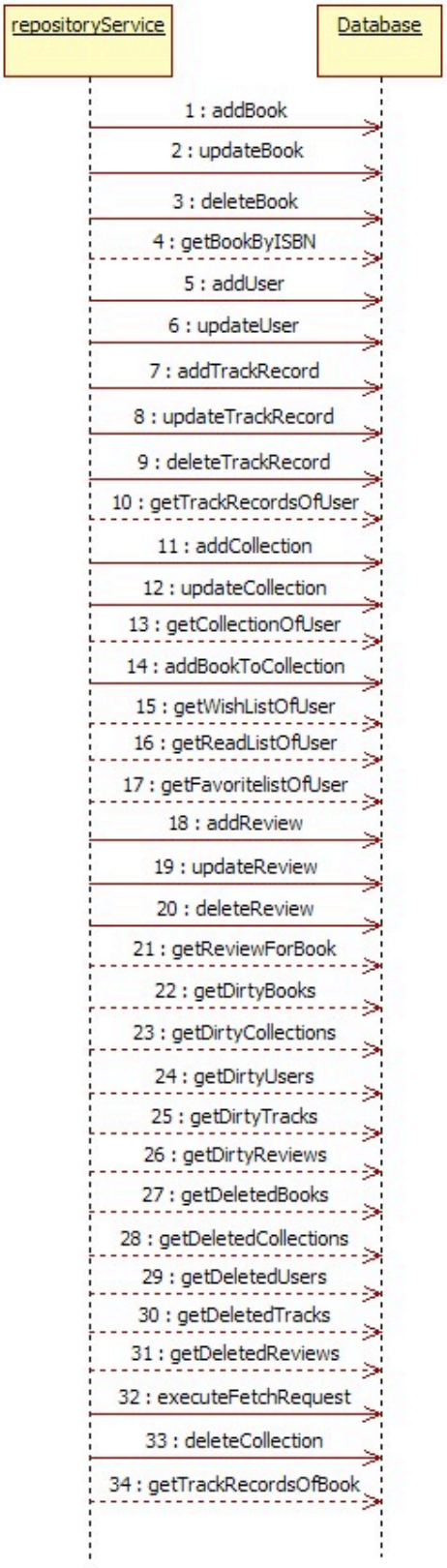
Diagram below shows the components of the system and their interconnections with each other. Description of each component is explained in Section 5 in detail.



3.2 Decomposition Description

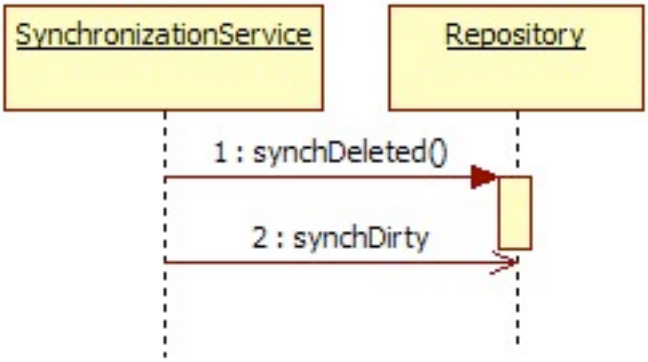
Components and their subsystems with sequence diagrams are presented below:

3.2.1 Repository Service



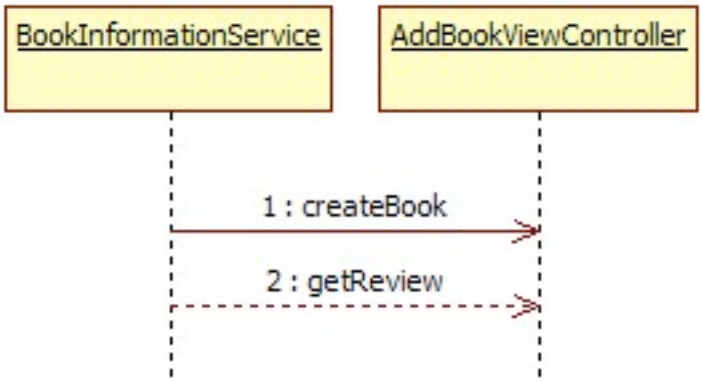
Repository service has the functions, which can be seen from the diagram, that communicate directly with database.

3.2.2 BookInformation Service



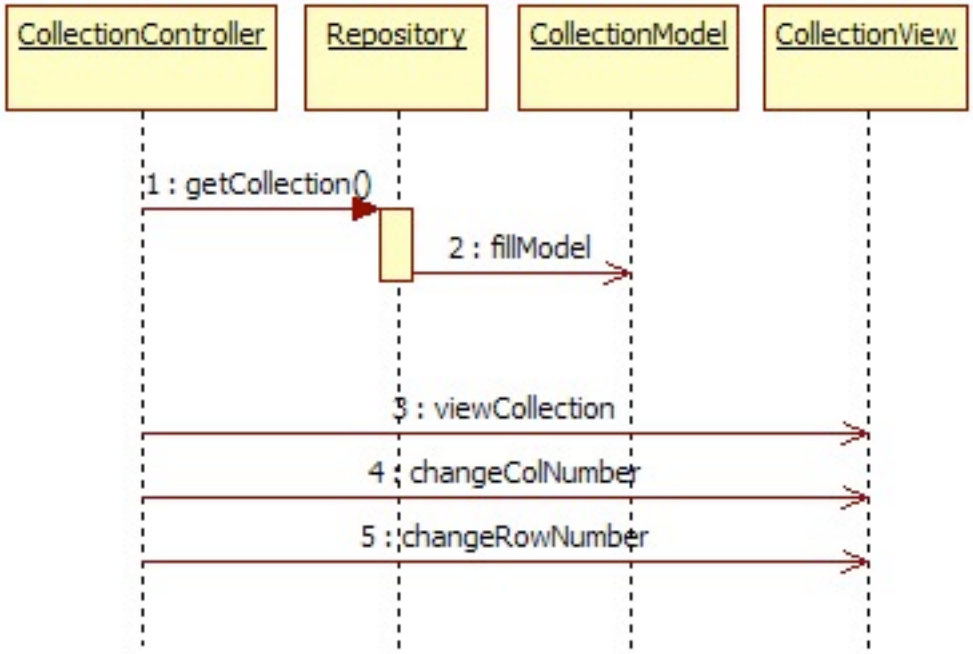
BookInformation Service has the function createBook and getReview which communicate with AddBookViewController.

3.2.3 Synchronization Service



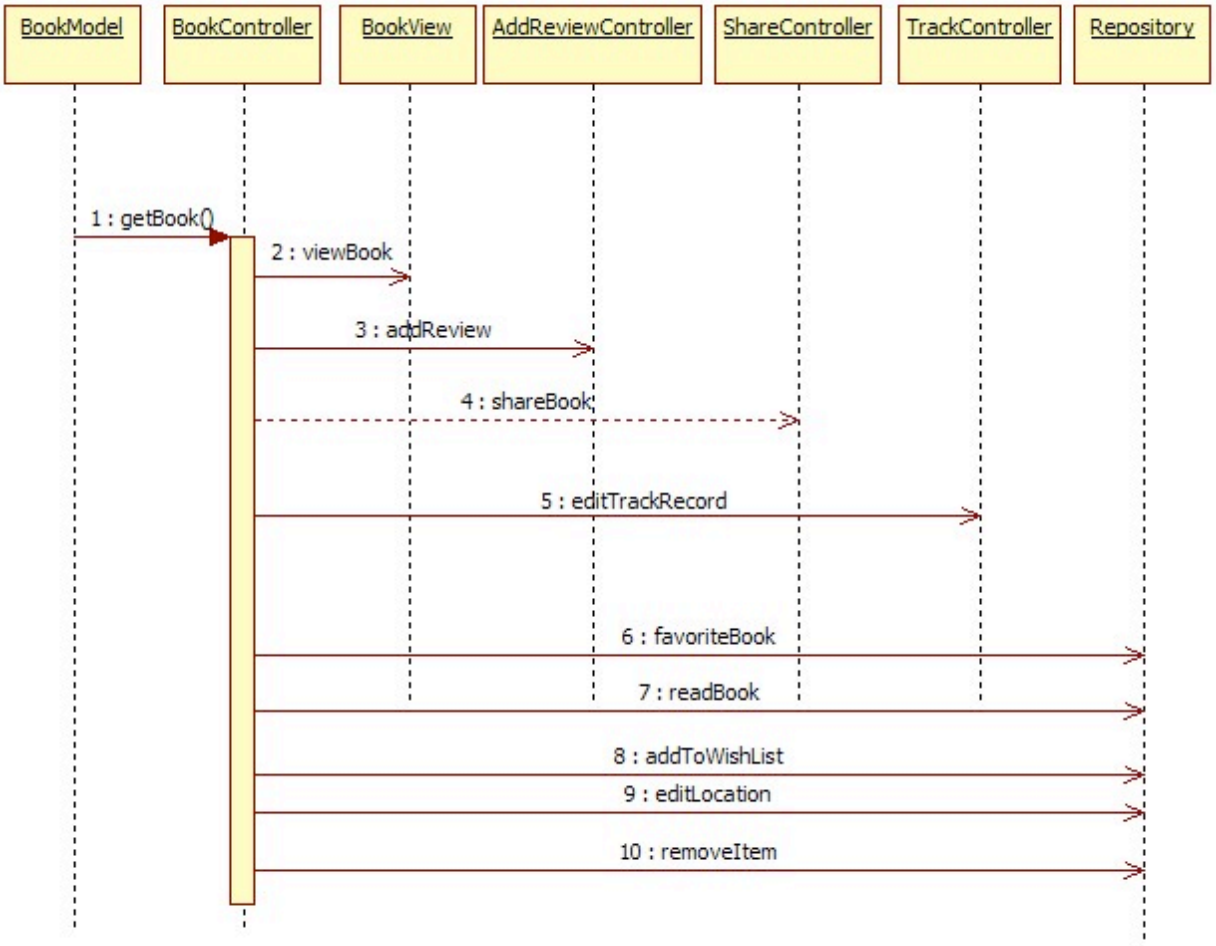
SynchronizationService has functions synchDeleted and synchDirty, which are useful for checking if the system is synchronized or not, that communicate with repository.

3.2.4 CollectionViewController



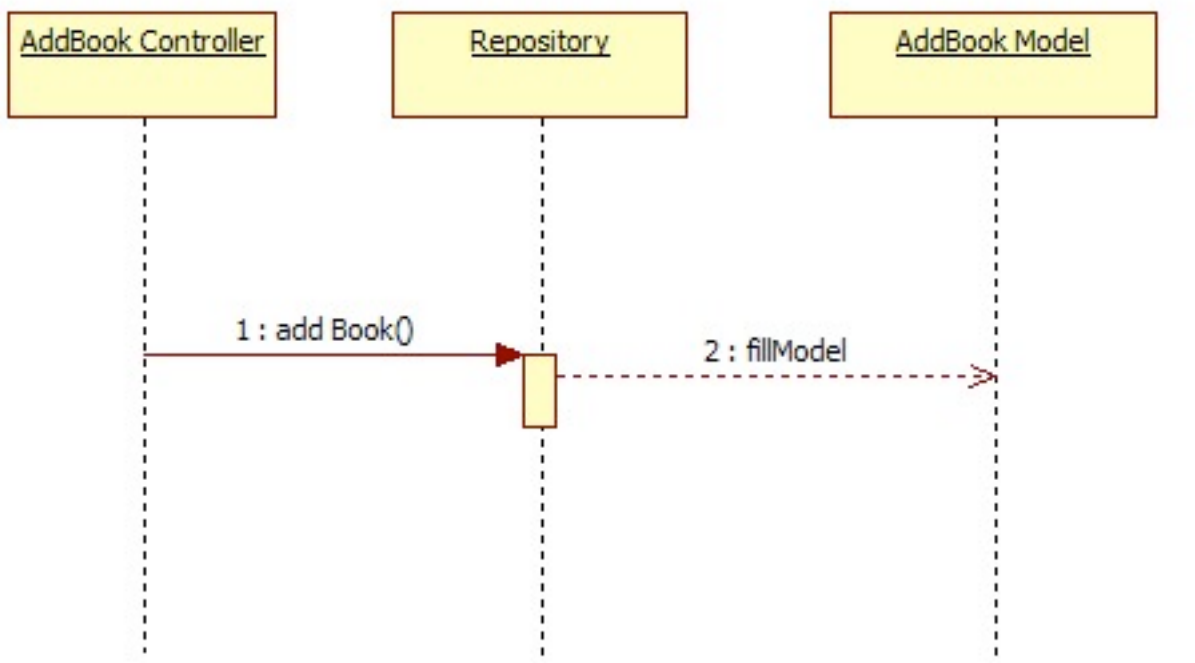
The relating functions between the Model View and Controller parts and the function communicating with repository are given in the diagram.

3.2.5 BookViewController



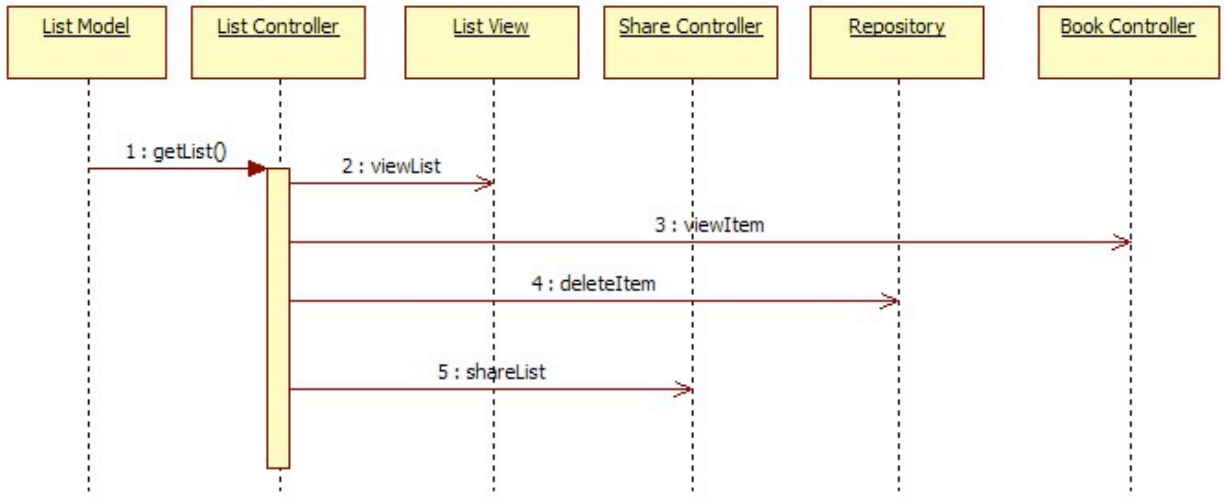
The relating functions in between, functions related to the repository and Controllers that are communicated or used are given in the diagram.

3.2.6 AddBookViewController



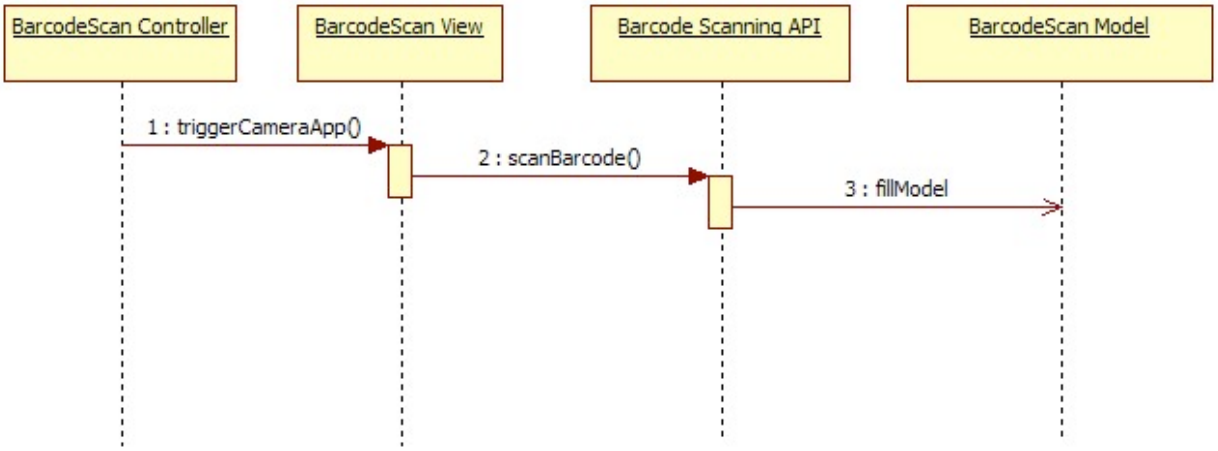
The relating functions between the Model View and Controller parts and the function communicating with repository are given in the diagram.

3.2.7 ListViewController



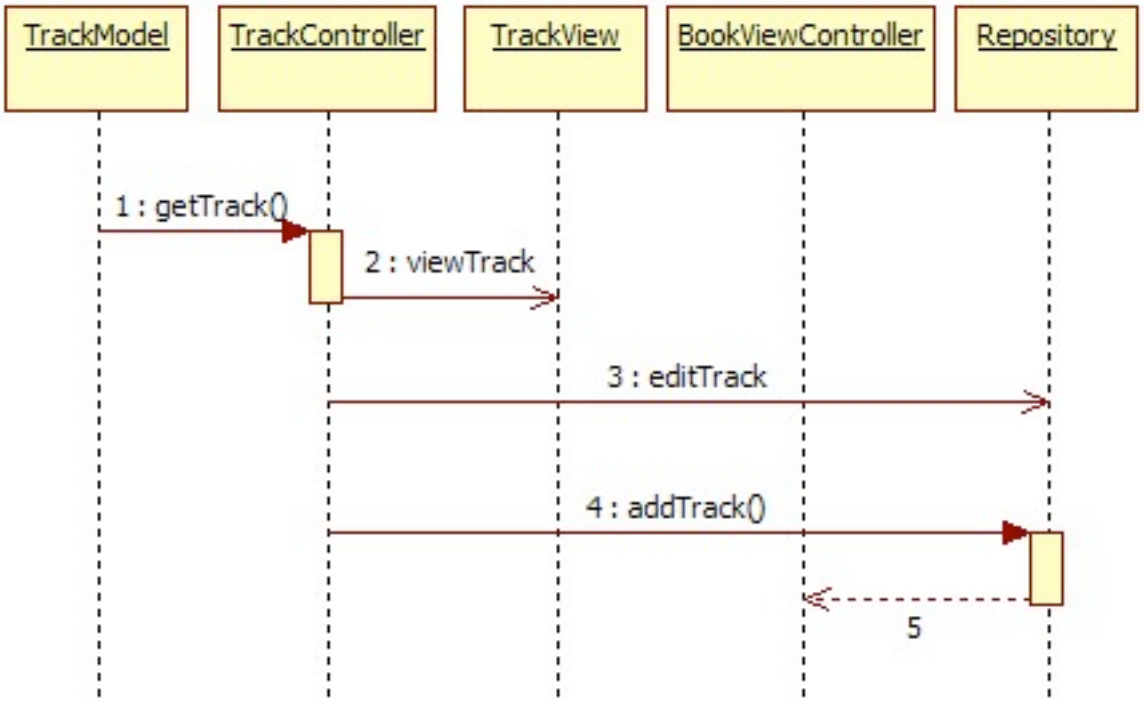
The relating functions in between, functions related to the repository and Controllers that are communicated or used are given in the diagram.

3.2.8 BarcodeScanViewController



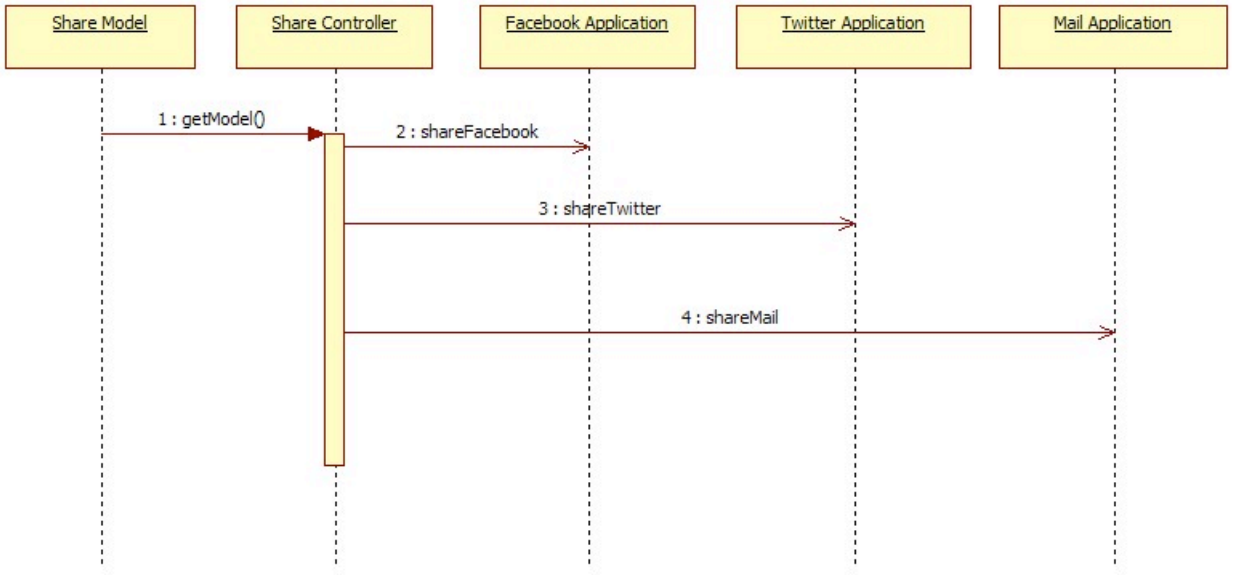
The relating functions between the Model View and Controller parts, related API and communication with it are given in the diagram.

3.2.9 TrackViewController



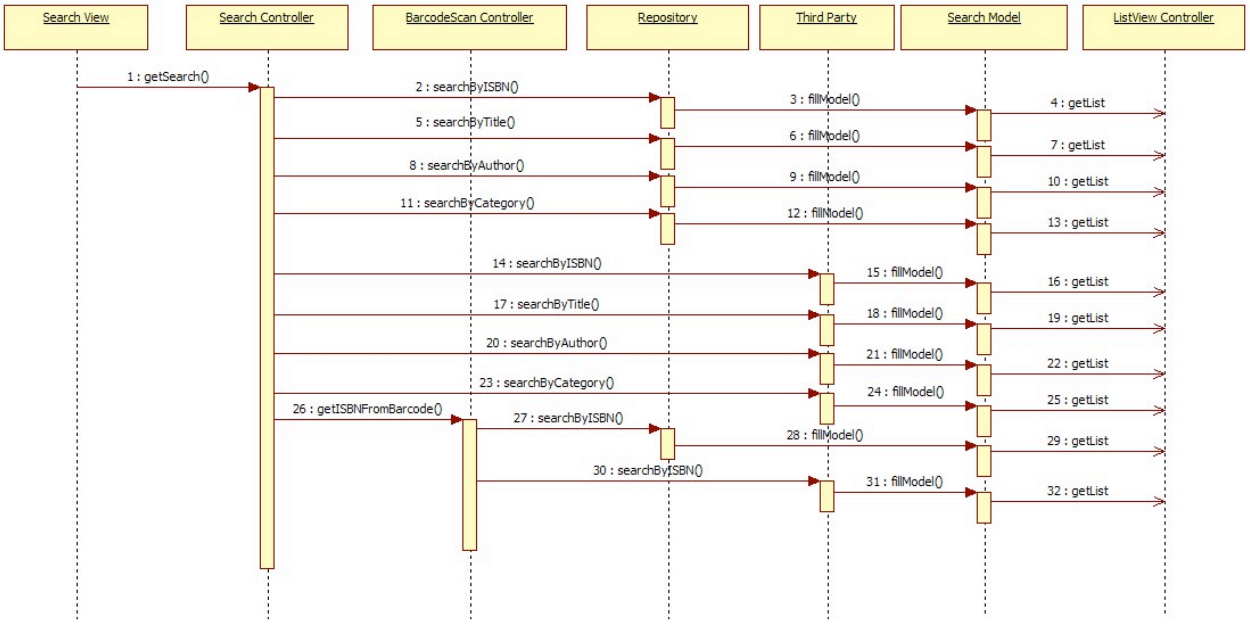
The relating functions in between, functions related to the repository and Controllers that are communicated or used are given in the diagram.

3.2.10 ShareViewController



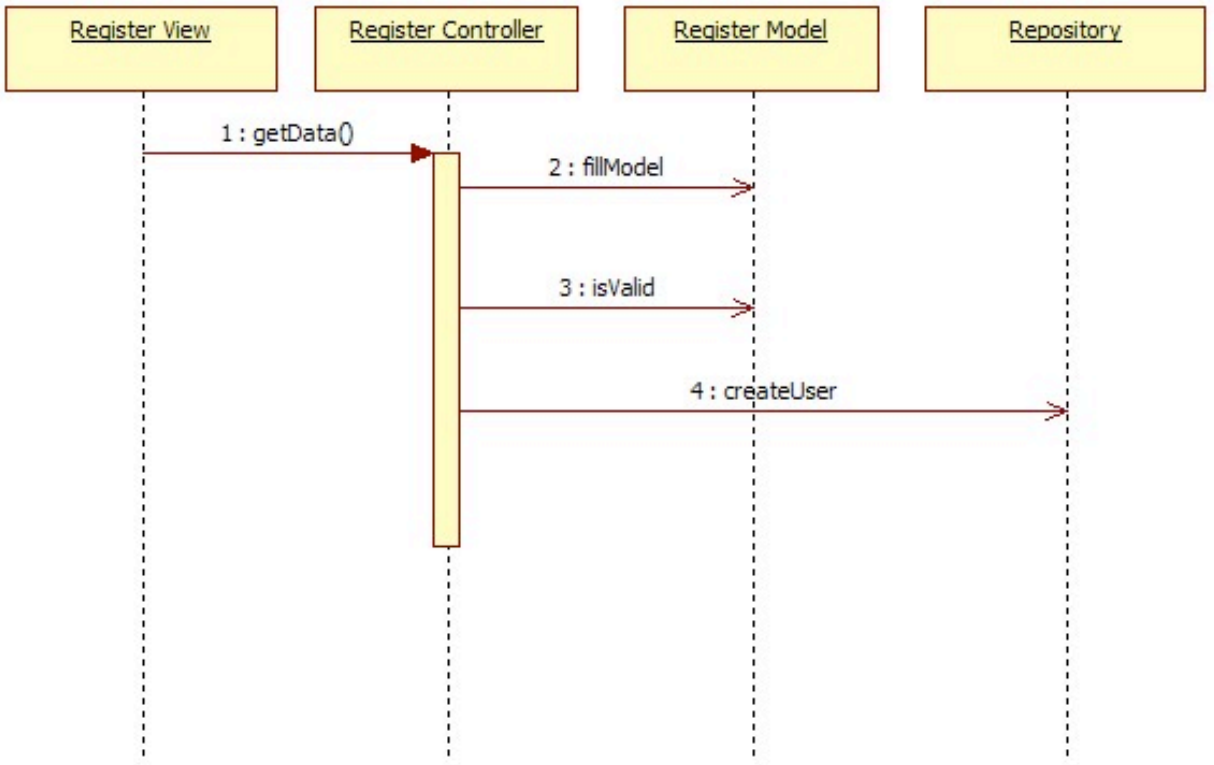
The relating functions in between; functions related to the external applications that are communicated or used are given in the diagram.

3.2.11 SearchViewController



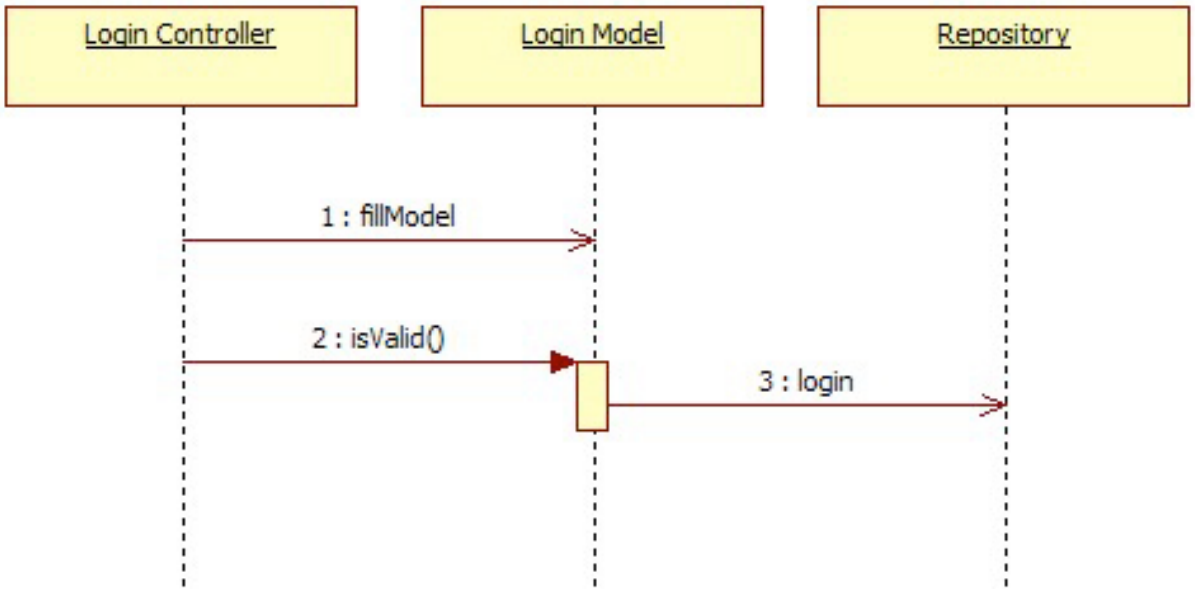
The relating functions in between, functions related to the repository and Controllers that are communicated or used are given in the diagram.

3.2.12 RegisterViewController



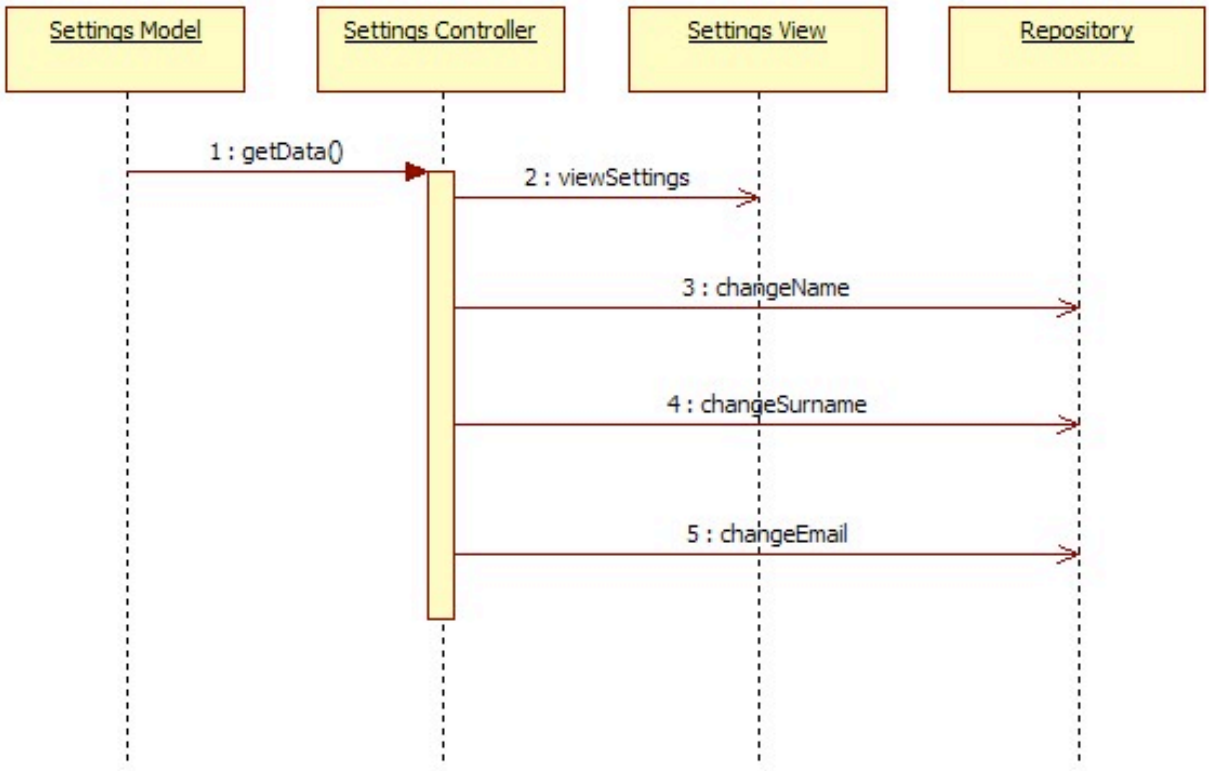
The relating functions between the Model View and Controller parts and the function communicating with repository are given in the diagram.

3.2.13 LoginViewController



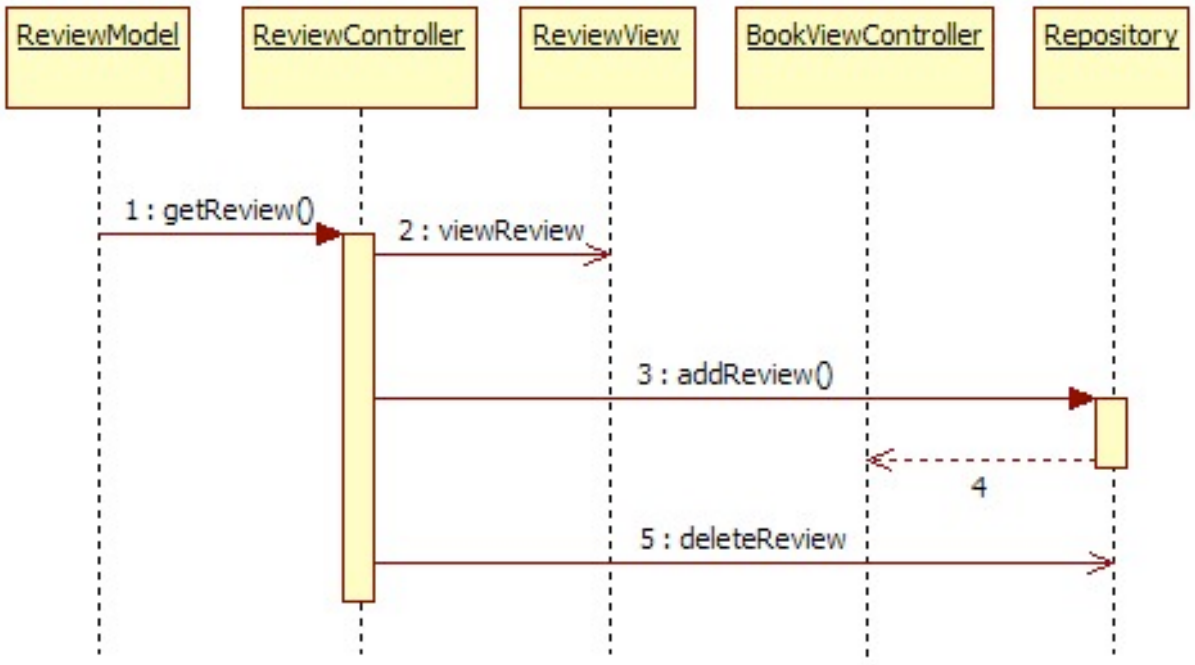
The relating functions between the Model View and Controller parts and the function communicating with repository are given in the diagram.

3.2.14 SettingsViewController



The relating functions between the Model View and Controller parts and the function communicating with repository are given in the diagram.

3.2.15 AddReviewViewController



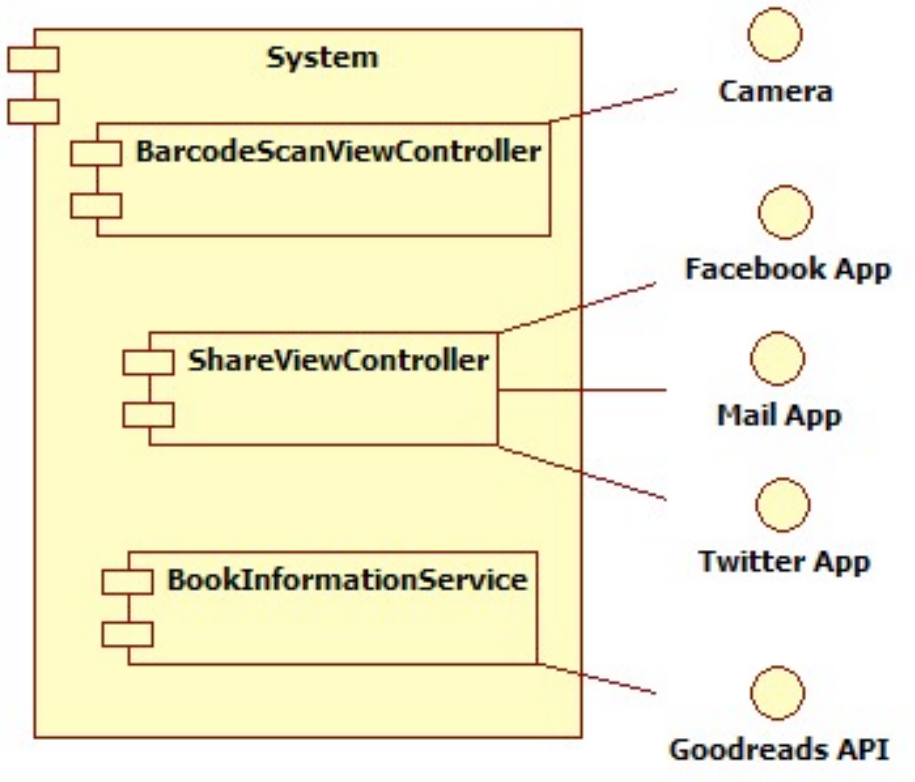
The relating functions between the Model View and Controller parts and the function communicating with repository are given in the diagram.

3.3 Design Rationale

In ShelfButler project, Model View Controller architecture is used. MVC pattern separates the representation of information from the user's interaction with it. Models represent knowledge. A model could be a single object or could be complex structure of objects. Views are the representations of models. Controllers are the links between a user and the system. The controller receives input from the user, translates it into appropriate messages and pass these messages on to views.

Since this application is targeted for iOS devices, Cocoa framework in iOS SDK shall be used. This framework is mainly based on the MVC architecture. Thus, this architecture is to be used while implementing this project.

3.4 External Interfaces

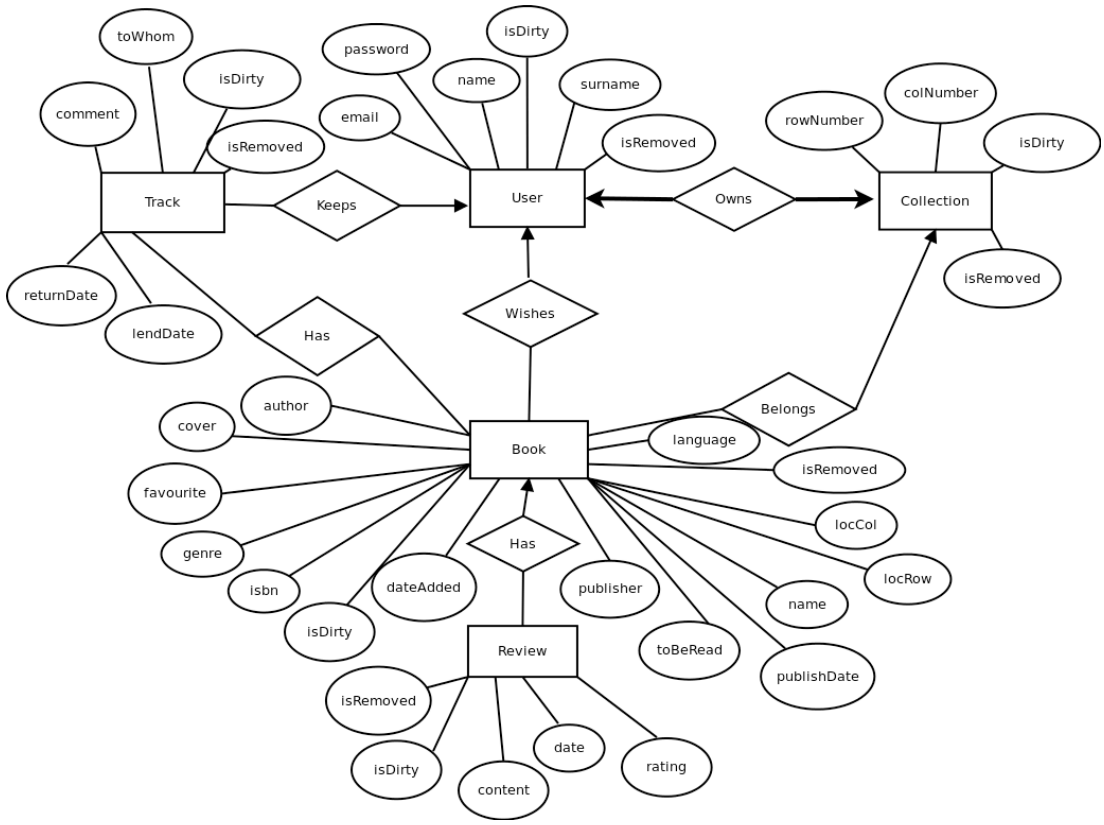


ShelfButler has communication with different external interfaces including; Facebook, Twitter, Goodreads, Mail Application and the device’s camera for barcode scanning option.

4 Data Design

4.1 Data Description

All of the information is kept in the databases which can only be accessed by Repository services.

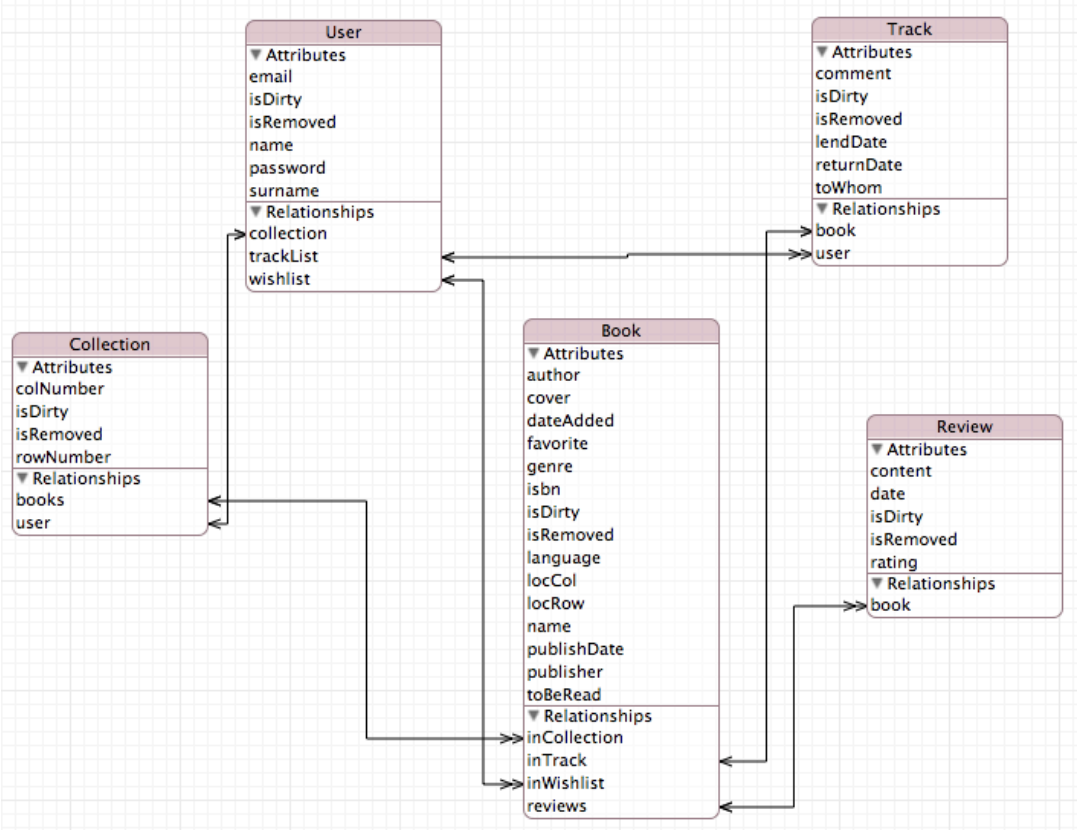


Attributes are drawn as ovals and are connected with a line to exactly one entity or relationship set.

A thick line indicates each entity in the entity set is involved in exactly one relationship.

An arrow from entity set to relationship set indicates a key constraint: each entity of the entity set can participate in at most one relationship in the relationship set.

4.1.1 Mobile Application Data Description



The diagram represents both entity relationships and classes of the application, since they are auto-generated from ER-Diagram.

This section provides information of entities and components in the mobile application of this project. The database of the mobile application will be implemented using Xcode and xcdatamodel as it is used for creating databases for programs written in objective C.

4.1.1.1 Book Entity

Book is the major entity of the whole database. Since it is a book collection management project, almost all entities and components of the project uses this entity. The attributes and relations associated with book are as follows:

Name	Type	Data Type	Description
name	attribute	String	Name of the book.
author	attribute	String	Name of the author of the book.
cover	attribute	String	File path to the cover picture.

Name	Type	Data Type	Description
dateAdded	attribute	Date	The date that the book was added to the collection.
isFavourite	attribute	Boolean	Whether the book is in favorite list or not.
genre	attribute	String	Genre of the book.
isbn	attribute	String	ISBN of the book, which uniquely identifies the book.
language	attribute	String	Language of the book.
locCol	attribute	Integer	The column that the book resides in the collection.
locRow	attribute	Integer	The row that the book resides in the collection.
publishDate	attribute	Date	The date that the book is published.
publisher	attribute	String	Name of the publisher of the book.
toBeRead	attribute	Boolean	Whether the book is in readlist or not.
inCollection	relationship	-	The relationship that indicates whether the book is associated with a collection or not. It is a many to one relationship.
inTrack	relationship	-	The relationship that indicates whether the book is lent to someone. It is a one to many relationship.

Name	Type	Data Type	Description
inWishlist	relationship	-	The relationship that indicates whether the book is in wishlist of a sure. It is a one to one relationship.
reviews	relationship	-	The relationship that indicates the reviews of the book. It is one to many relationship.
isDirty	attribute	Boolean	Whether the entry is synch with cloud or not.
isRemoved	attribute	Boolean	Whether the entry is deleted or not.

4.1.1.2 User Entity

Users are the ones they own the collections. The attributes and relations associated with user are as follows:

Name	Type	Data Type	Description
name	attribute	String	Name of the user.
surname	attribute	String	Surname of the user.
email	attribute	String	E-mail of the user.
password	attribute	String	Password of the user.
collection	relationship	-	The relationship that determines the collection that belongs to the user. It is a one to one relationship.

Name	Type	Data Type	Description
trackList	relationship	-	The relationship that determines the tracklist of the user. That is which book the user owns is lent to others. It is a one to many relationship.
wishlist	relationship	-	The relationship that relates a wishlist to the user. It is a one to many relationship.
isDirty	attribute	Boolean	Whether the entry is synch with cloud or not.
isRemoved	attribute	Boolean	Whether the entry is deleted or not.

4.1.1.3 Collection Entity

Collections represent the libraries that belong to users. The attributes and relations associated with collection are as follows:

Name	Type	Data Type	Description
colNumber	attribute	Integer	The number of columns in the library. That is introduced by the user.
rowNumber	attribute	Integer	The number of rows in the library. That is introduced by the user.
books	relationship	-	The relationship that indicates which books belong to a specific library. It is a one to many relationship.

Name	Type	Data Type	Description
user	relationship	-	The relationship that determines the collection belongs to which user. It is one to one relationship.
isDirty	attribute	Boolean	Whether the entry is synch with cloud or not.
isRemoved	attribute	Boolean	Whether the entry is deleted or not.

4.1.1.4 Track Entity

The information about lent books are kept in the track entity. The attributes and relations associated with collection are as follows:

Name	Type	Data Type	Description
comment	attribute	String	Comments of the user about the lending.
lendDate	attribute	Date	The date which the book is lent.
returnDate	attribute	Date	The date which the book is returned.
toWhom	attribute	String	The person who lent the book.
book	relationship	-	The relationship that indicates which book is related to track. It is a one to one relationship.
user	relationship	-	The relationship that associates the lent books with user. It is a one to one relationship.

Name	Type	Data Type	Description
isDirty	attribute	Boolean	Whether the entry is synch with cloud or not.
isRemoved	attribute	Boolean	Whether the entry is deleted or not.

4.1.1.5 Review Entity

The information about reviews are kept in the review entity. The attributes and relations associated with collection are as follows:

Name	Type	Data Type	Description
content	attribute	String	Content of the review.
date	attribute	Date	The date which review is written.
rating	attribute	Integer	The rating of the book.
book	relationship	-	The relationship that indicates which book is related to review. It is a one to one relationship.
isDirty	attribute	Boolean	Whether the entry is synch with cloud or not.
isRemoved	attribute	Boolean	Whether the entry is deleted or not.

4.1.2 Cloud Data Description

Database for web interface of the application is stored in the cloud. Entities and relationships are same as mentioned above in the mobile part. Mobile and cloud databases are kept synchronized via Synchronization service.

The only difference from mobile part is that cloud database will be implemented using SQL.

4.2 Data Dictionary

Name	Type	Referred Section
Book	Entity	4.1.1.1
Collection	Entity	4.1.1.3
Review	Entity	4.1.1.5
Track	Entity	4.1.1.4
User	Entity	4.1.1.2

5 Component Design

5.1 Repository Service

This component is an abstraction over actual database. It provides all methods (namely insert, update, delete, fetch) for database operations. Any other component wants to access database, shell use the methods implement by Repository Service.

This component is basically is a singleton class, so that the same instance can be accessed anywhere in the application.

5.1.1 Repository Attributes

Repository class has one attribute that is a database connection.

Field Name	Data Type	Description
database	Database connection	Created at the initialization of Repository object. Keeps a connection to the database. Connection closed when the last reference to Repository object is destructed.

5.1.2 Repository Methods

Repository class provides methods for any database operation.

Method Name	Return Type	Description
addBook	Book	Takes a Book object as an argument. Inserts it to database. Returns the object inserted. NULL, if insertion fails.
updateBook	Void	Takes old and new entries as arguments. Replaces the old with new one in the database.

Method Name	Return Type	Description
deleteBook	Void	Takes the Book object to be deleted as an argument. Removes corresponding entry from database.
getBookByISBN	Book	Takes ISBN number as argument. Fetches and returns the requested entry as a Book object. NULL, if not found.
addUser	User	Takes required information to register as arguments. Inserts a new user to database. Returns the object inserted. NULL, if insertion fails.
updateUser	Void	Takes old and new entries as arguments. Replaces the old with new one in the database.
addTrackRecord	Track	Takes tracking information and a Book as arguments. Insert new track record for the given item. Returns the Track object.
updateTrackRecord	Void	Takes old and new entries as arguments. Replaces the old with new one in the database.
deleteTrackRecord	Void	Takes the Track object to be deleted as an argument. Removes corresponding entry from database.
getTrackRecordsOfBook	Track array	Takes a Book object as an argument. Fetches and returns the Track list of this book.
getTrackRecordsOfUser	Track array	Takes a User object as an argument. Fetches an returns the Track list of all books owned by this user.

Method Name	Return Type	Description
addCollection	Collection	Takes row and column numbers as arguments. Creates and insert a new collection to database.
updateCollection	Void	Takes old and new entries as arguments. Replaces the old with new one in the database.
deleteCollection	Void	Takes the Collection object to be deleted as an argument. Removes corresponding entry from database.
getCollectionOfUser	Collection	Takes a User object as an argument. Fetches an return the Collection owned by this user.
addBookToCollection	Void	Takes a Book and a Collection object as arguments. Adds book to collection in the database.
getWishlistOfUser	Book array	Takes a User object as an argument. Fetches and returns all the books in the wishlist of this user.
getReadlistOfUser	Book array	Takes a User object as an argument. Fetches and returns all the books in the readlist of this user.
getFavoritelistOfUser	Book array	Takes a User object as an argument. Fetches and returns all the books in the readlist of this user.
addReview	Review	Takes a Review object as an argument. Inserts it to database. Returns the object inserted. NULL, if insertion fails.

Method Name	Return Type	Description
updateReview	Void	Takes old and new entries as arguments. Replaces the old with new one in the database.
deleteReview	Void	Takes the Review object to be deleted as an argument. Removes corresponding entry from database.
getReviewForBook	Review array	Takes a Book object as an argument. Fetches and returns all the reviews for this item.
getDirtyBooks	Book array	Fetches and returns all dirty book entries.
getDirtyCollections	Collection array	Fetches and returns all dirty collection entries.
getDirtyUsers	User array	Fetches and returns all dirty user entries.
getDirtyTracks	Track array	Fetches and returns all dirty track entries.
getDirtyReviews	Review array	Fetches and returns all dirty review entries.
getDeletedBooks	Book array	Fetches and returns all books marked as deleted.
getDeletedCollections	Collection array	Fetches and returns all collections marked as deleted.
getDeletedUsers	User array	Fetches and returns all users marked as deleted.
getDeletedTracks	Track array	Fetches and returns all tracks marked as deleted.
getDeletedReviews	Review array	Fetches and returns all reviews marked as deleted.

Method Name	Return Type	Description
executeFetchRequest	Array	Takes a fetch query as an argument. Executes and returns the results. (For more general usage.)

5.2 BookInformation Service

This component is responsible for providing book metadata from the given ISBN. Since ShelfButler application will not have an huge ISBN database, this service will connect to remote services, that is Goodreads API.

It is designed to be a class with no attributes. Since no data stored in the object, all methods are to be implemented as class methods, not instance methods.

Because of the fact that the system uses online services, it requires a network connection for this service to work.

5.2.1 BookInformation Attributes

Currently, BookInformation class does not have any attribute to store data since it is planned to user 3rd party APIs. It may have an ISBN database in the future.

5.2.2 BookInformation Methods

BookInformation class provides methods to get any type of metadata for a given ISBN. All methods are class methods, so that they can be used without creating an instance of this class.

Method Name	Return Type	Description
createBook	Book	Takes the ISBN as an argument. Passes the ISBN to Goodreads API. Processes the result and fills a Book object to be returned.
getReview	String array	Takes the ISBN as an argument. Returns the reviews via Goodreads API.

Method Name	Return Type	Description
getRecommendations	Book array	Takes the ISBN as an argument. Get recommendations for the given book and returns a list of books.

5.3 Synchronization Service

This component is responsible for the synchronization between cloud and local databases. This service only triggered by Repository Service when an operation happens on the local database.

5.3.1 Synchronization Attributes

Synchronization class has one attribute.

Field Name	Data Type	Description
isSynch	Boolean	Attribute to check whether cloud and local databases are synch. Initialized at the application launch.

5.3.2 Synchronization Methods

Synchronization class provides one method.

Method Name	Return Type	Description
synchDirty	Boolean	Gets all dirty entries from Repository Service and synchronizes local and cloud databases. Returns True if successful.
synchDeleted	Boolean	Gets all deleted entries from Repository Service and synchronizes local and cloud databases. Returns True if successful.

5.4 CollectionViewController

This component is the one that shows the users their collection and enables them to manage collection settings.

5.4.1 Collection Model

This is the structure where collection data resides after being fetched from the database.

Field Name	Data Type	Description
collection	Collection object	Keeps the collection data of the user.

5.4.2 Collection View

Collection view is what shows the users their collection. They see and manage their collection through this view.

5.4.3 Collection Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
viewCollection	Collection	At the time this component initialized, it fetches the Collection object from Repository service and passes it to the view.
changeColNumber	Void	It sends a request to Repository service to change the column number of collection.
changeRowNumber	Void	It sends a request to Repository service to change the row number of collection.

5.5 BookViewController

This component is responsible for viewing a book item with all required information. It uses the interfaces provided by Track and Repository services.

5.5.1 Book Model

Model of this component is Book object.

Field Name	Data Type	Description
book	Book	Attribute to keep the current item viewed. It is filled by the parent view that sends a request to this component.

5.5.2 Book View

The view shows all information included in a Book object. Besides, it provides user interface elements to add new tracking record, shop online and remove item from the collection.

5.5.3 Book Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
viewBook	Book	Passes the model item to its view.
editTrackRecord	Void	Triggers TrackViewController and passes the model to it.
removeItem	Void	Sends a request to Repository Service to remove current item from user's collection.
favoriteBook	Void	Marks current book as favorited using Repository Service.
readBook	Void	Marks current book as read using Repository Service.

Method Name	Return Type	Description
addToWishlist	Void	Adds current book to user's wishlist using Repository Service.
addReview	Void	Triggers AddReviewViewController and passes the model to it.
editLocation	Void	Updates the physical location of current book with new values. Passes new object to Repository Service to update the entry in database.
shareBook	Void	Triggers ShareViewController and passes the model to it.

5.6 AddBookViewController

This component is the one to be used for adding books to the collection.

5.6.1 AddBook Model

This is the structure where the data of the book to be added is kept.

Field Name	Data Type	Description
book	Book	Keeps the Book object that is to be added to the collection.

5.6.2 AddBook View

Collection view is what shows the users the book that is about to be added to the collection.

5.6.3 AddBook Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
addBook	Void	It sends a request to the repository service to add the book to the collection. The dateAdded, isFavourite, toBeRead and inTrack attributes of the Book object is initialized before the request.

5.7 ListViewController

This component is base component for all list views namely; favorite list, read list and wish list. It provides basic functionalities to manage and display lists.

5.7.1 List Model

Model is a structure as a list contains Book objects.

Field Name	Data Type	Description
bookList	Book array	Keeps the books in a list.

5.7.2 List View

It is a table view that shows the items in the corresponding list to the user. User interacts with the system through this view.

5.7.3 List Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
viewList	Book array	At the time this component initialized, it fetches list from Repository service and passes it to the view.

Method Name	Return Type	Description
deleteItem	Void	It sends a request to Repository service to delete the selected item.
viewItem	Book	It sends a request to the BookViewController to display the selected item.
shareList	Void	Sends the list to Share component with desired social network.

5.8 BarcodeScanViewController

This component provides an interface to user to scan the barcode, processes scanned image and returns ISBN.

5.8.1 BarcodeScan Model

Model of this component is a string to keep ISBN data.

Field Name	Data Type	Description
isbn	String	Attribute to keep the value of ISBN.

5.8.2 BarcodeScan View

BarcodeScan opens the camera interface of the mobile device and provides user interface elements in order to help user to scan a barcode properly.

5.8.3 BarcodeScan Controller

Controller is responsible to send a request to Camera application and pass it to current view. BarcodeScan controller has an barcode scanning library integrated and provides methods to use it. Methods are listed below:

Method Name	Return Type	Description
scanBarcode	String	Takes the image data as an arguments. Passes it to barcode scanning library. Returns the ISBN number by processing the data coming from internal library.

5.9 TrackViewController

This component provides an interface to the user which includes lending a book or receiving a book back.

5.9.1 Track Model

Model of this component includes a Track object.

Field Name	Data Type	Description
lendRecord	Track	Keeps the Track object including the information of the lent book.

5.9.2 Track View

The view shows all information included in that Track object. Other than that, it includes user interface elements such as lending and receiving book.

5.9.3 Track Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
viewTrack	Track	It passes the model to its view.

Method Name	Return Type	Description
editTrack	Track	It creates a new Track object and sends both of the old and new Track objects to the RepositoryService for updating.
addTrack	Track	It creates a new Track object for the information of the lent book.

5.10 ShareViewController

This component provides an interface to share the book or the list of books (wishlist, readlist, favorites).

5.10.1 Share Model

Model of this component includes a book and a list which are desired to be shared.

Field Name	Data Type	Description
sharedBook	Book	Keeps the Book object that is desired to be shared.
sharedList	Book array	Keeps the list of the books that is desired to be shared.

5.10.2 Share View

The view shows three panels for sharing options, Facebook, Twitter, and Mail.

5.10.3 Share Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
shareFacebook	Void	It opens the Facebook application and redirects the user to share the book or list on his wall.
shareTwitter	Void	It opens the Twitter application and redirects the user to share the book or list on his page.
shareMail	Void	It opens the Mail application and redirects the user to send the book or list as a text to another person.

5.11 SearchViewController

This component provides a search interface to the user to search a book locally (in his own collection) or online. The interface provides searching via barcode scanning or by ISBN of the requested book.

5.11.1 Search Model

Model of this component includes two structures as lists containing Book objects and a boolean to determine if the search will be on local or online.

Field Name	Data Type	Description
localBookList	Book array	Keeps the list returned by searching the book locally.
onlineBookList	Book array	Keeps the list returned by searching the book online.
isLocal	Boolean	Keeps the information for making the search either local or online. If the variable is ‘true’ the search will be made locally. If the variable is ‘false’ the search will be made online.

5.11.2 Search View

There are four views. In the main view user is asked to select either barcode search or ISBN search. After that the user is asked to select either local or online.

5.11.3 Search Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
searchByISBN	Book array	It makes a search of the book by ISBN and sends a request to ListViewController to display the list of the book(s) found.
getISBNfromBarcode	String	It sends a request to Barcode scan and returns the ISBN of the desired book. Then calls searchByISBN method.
searchByTitle	Book array	It makes a search of the book by title and sends a request to ListViewController to display the list of the book(s) found.
searchByAuthor	Book array	It makes a search of the book by author's name and sends a request to ListViewController to display the list of the book(s) found.
searchByCategory	Book array	It makes a search of the book by category and sends a request to ListViewController to display the list of the book(s) found.

5.12 RegisterViewController

This component is responsible for registering a new user to the system. It basically asks required information to the user for registration; name, surname, unique e-mail address and a password.

5.12.1 Register Model

Fields of the model are as follows;

Field Name	Data Type	Description
id	Integer	This is a private field. Each user has a unique id automatically assigned by database.
name	String	Keeps the name of the user.
surname	String	Keeps the surname of the user.
mail	String	Keeps the e-mail address of the user.

5.12.2 Register View

The view is a GUI to take Register Model data from the user. For each field mentioned above except id, the register method of Register Controller takes input through this view.

5.12.3 Register Controller

The methods of the controller are as follows:

Method Name	Return Type	Description
register	Void	Takes the data of fields from Register View and sets the fields.
isValid	Boolean	Checks the fields. Returns true if they are valid. Returns false and sends information about the missing fields to the Register View they are not valid.

Method Name	Return Type	Description
createUser	Void	It sends the Register Model fields to Repository service to create a user.

5.13 LoginViewController

This component is responsible for logging a new user to the system. It asks required information to the user for logging; mail address and password.

5.13.1 Login Model

Fields of the model are as follows;

Field Name	Data Type	Description
mail	String	Keeps the e-mail address of the user.
password	String	Keeps the password of the user.

5.13.2 Login View

The view is a GUI to take Login Model data from the user. For each field mentioned above, the login method of Login Controller takes input through this view.

5.13.3 Login Controller

The methods of the controller are as follows:

Method Name	Return Type	Description
login	Void	Takes the data of fields from Login View and sets the fields.
isValid	Boolean	Sends a request to the Repository service to check the input password.

Method Name	Return Type	Description
forgotPassword	Void	Sends a randomly generated password to user's e-mail address.

5.14 SettingsViewController

This component is the one that the users see their account settings and manage them.

5.14.1 Settings Model

This is a structure where the attributes of the user object resides after fetching from the database.

Field Name	Data Type	Description
user	User	Keeps the account information of the user.

5.14.2 Settings View

It is a view that shows the account data. Users interact with the system and manage their settings through this view.

5.14.3 Settings Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
viewSettings	User	At the time this component initialized, it fetches the User object from Repository service and passes it to the view.
changePassword	Void	It sends a request to Repository service to change the password of the user.

Method Name	Return Type	Description
changeName	Void	It sends a request to the Repository service to change the name of the user.
changeSurname	Void	It sends a request to the Repository service to change the surname of the user.
logout	Void	Ends current session of the user. Triggers LoginViewController.

5.15 AddReviewViewController

This component is responsible for managing reviews of a book.

5.15.1 Review Model

Model of this component is Review object.

Field Name	Data Type	Description
review	Review	Keeps the Review object including the information of the review.

5.15.2 Review View

It is a view that shows the user interface elements to get inputs such as rating, review text and date.

5.15.3 Review Controller

All functionalities of this component are handled by the controller. Methods are listed below:

Method Name	Return Type	Description
viewReview	Review	It passes the model to its view.

Method Name	Return Type	Description
addReview	Review	Sends created Review object to Repository Service to be inserted to database.
deleteReview	Void	It sends a request to Repository Service to delete corresponding entry.

5.16 WebServices

In this section all web services used for synchronization purposes are explained in detail with their parameters and return values.

Method Name	Return Type	Description
addBook	Boolean	Takes the serialized JSON string of Book object as an argument. Deserialize the input to the Book and inserts it to database. Returns true if the operation is successful, false otherwise.
updateBook	Boolean	Takes the serialized JSON string of Book object as an argument. Deserialize the input to the Book and updates the corresponding one in database. Returns true if the operation is successful, false otherwise.
deleteBook	Boolean	Takes the serialized JSON string of Book object as an argument. Deserialize the input to the Book and removes corresponding entry from database. Returns true if the operation is successful, false otherwise.

Method Name	Return Type	Description
addUser	Boolean	Takes the serialized JSON string of User object as an argument. Deserialize the input to the Book and inserts it to database. Returns true if the operation is successful, false otherwise.
updateUser	Boolean	Takes the serialized JSON string of User object as an argument. Deserialize the input to the Book and updates the corresponding one in database. Returns true if the operation is successful, false otherwise.
addCollection	Boolean	Takes the serialized JSON string of Collection object as an argument. Deserialize the input to the Book and inserts it to database. Returns true if the operation is successful, false otherwise.
updateCollection	Boolean	Takes the serialized JSON string of Collection object as an argument. Deserialize the input to the Book and updates the corresponding one in database. Returns true if the operation is successful, false otherwise.

Method Name	Return Type	Description
deleteCollection	Boolean	Takes the serialized JSON string of Collection object as an argument. Deserialize the input to the Book and removes corresponding entry from database. Returns true if the operation is successful, false otherwise.
addReview	Boolean	Takes the serialized JSON string of Review object as an argument. Deserialize the input to the Book and inserts it to database. Returns true if the operation is successful, false otherwise.
updateReview	Boolean	Takes the serialized JSON string of Review object as an argument. Deserialize the input to the Book and updates the corresponding one in database. Returns true if the operation is successful, false otherwise.
deleteReview	Boolean	Takes the serialized JSON string of Review object as an argument. Deserialize the input to the Book and removes corresponding entry from database. Returns true if the operation is successful, false otherwise.
addTrackRecord	Boolean	Takes the serialized JSON string of Track object as an argument. Deserialize the input to the Track and inserts it to database. Returns true if the operation is successful, false otherwise.

Method Name	Return Type	Description
updateTrackRecord	Boolean	Takes the serialized JSON string of Track object as an argument. Deserialize the input to the Track and updates the corresponding one in database. Returns true if the operation is successful, false otherwise.
deleteTrackRecord	Boolean	Takes the serialized JSON string of Track object as an argument. Deserialize the input to the Track and removes corresponding entry from database. Returns true if the operation is successful, false otherwise.

6 Human Interface Design

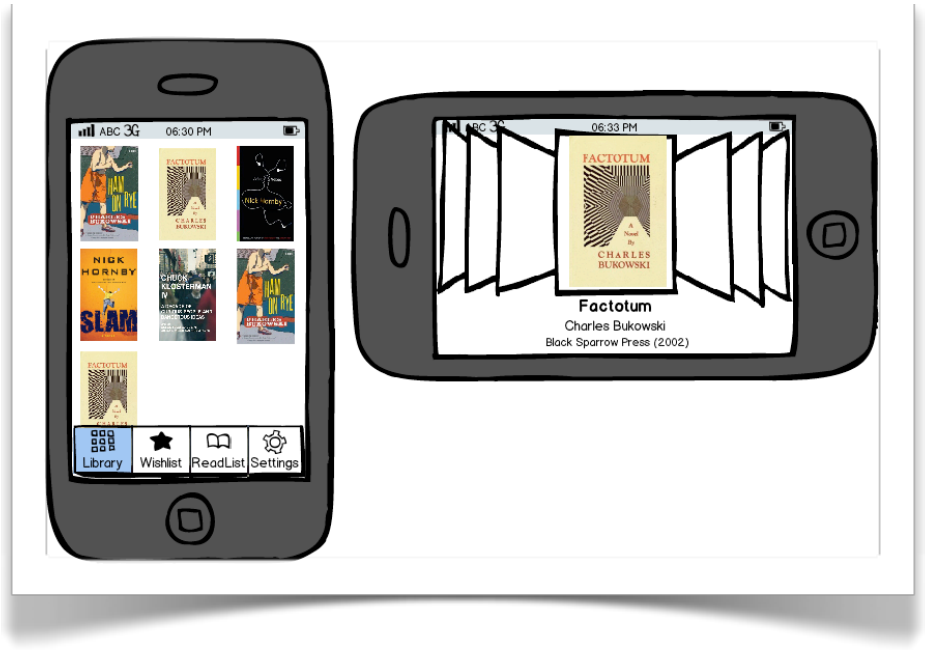
6.1 Overview of User Interface

Primary concern for user interface to be easy to use. Application welcomes users with a screen to login or register. After a successful login, users can access their collection, lists or setting with a tabbed menu.

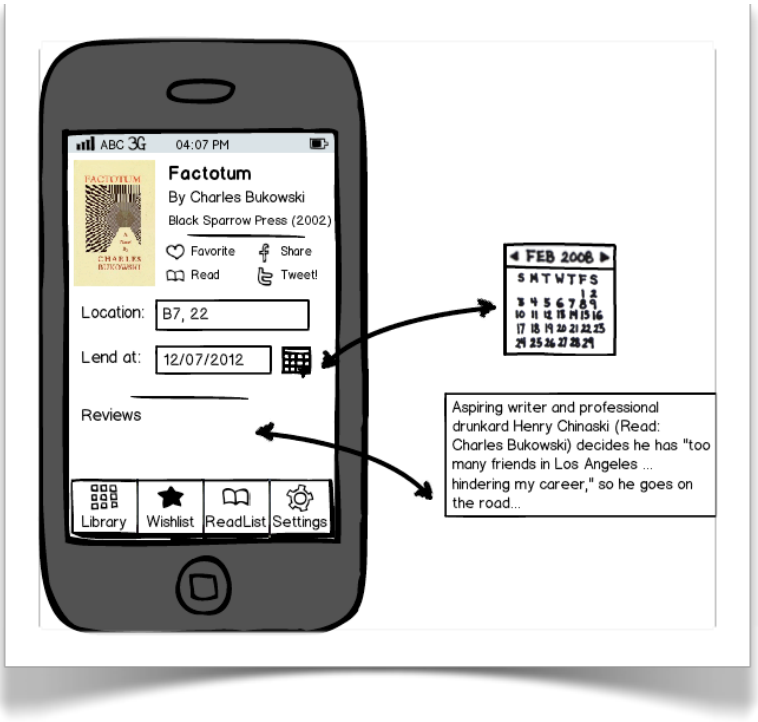
Login Screen: A simple login screen for users to enter e-mail and password.

Register Screen: A simple screen with a form requesting e-mail, name and password only. Registration form will not ask too much information not to disturb users.

6.2 Screen Images



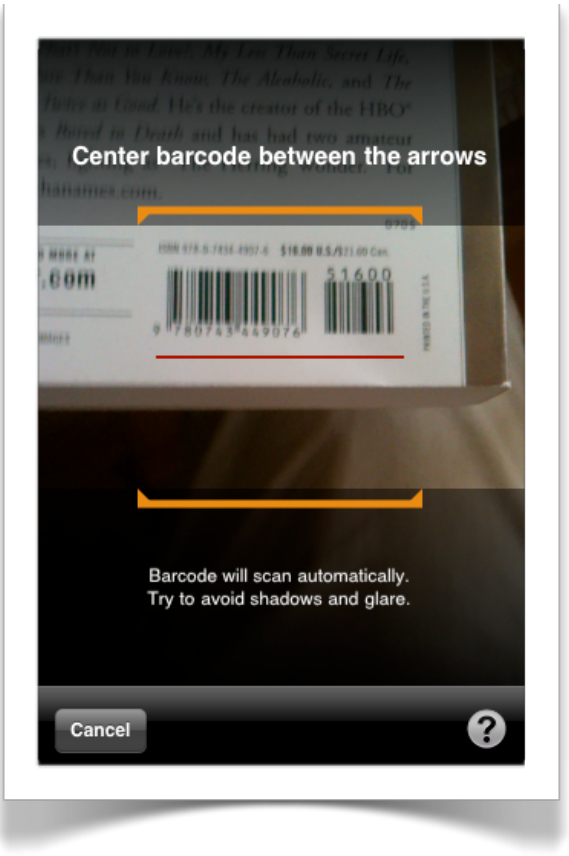
Collection View



Book View

Search View

List View



Barcode Scan View

6.3 Screen Objects and Actions

All screen objects will be compatible to default iOS type user interface elements so that users can feel more comfortable while using the application.

7 Requirements Matrix

Use Case	Matching Component
4.1.1. Add Book By ISBN	5.6. AddBookViewController 5.16 Webservices
4.1.2. Add Book By Barcode	5.6. AddBookViewController 5.16 Webservices
4.1.3. Remove Book	5.5. BookViewController 5.16 Webservices
4.2.1. Add to Wishlist	5.5. BookViewController 5.16 Webservices
4.2.2. Remove from Wishlist	5.7. ListViewController 5.16 Webservices
4.2.3. Display Wishlist	5.7. ListViewController
4.2.4. Share Wishlist	5.10. ShareViewController
4.3.1. Add to Favorites	5.5. BookViewController 5.16 Webservices
4.3.2. Remove from Favorites	5.7. ListViewController 5.16 Webservices
4.3.3. Display Favorites	5.7. ListViewController
4.3.4. Share Favorites	5.10. ShareViewController
4.4.1. Add to Readlist	5.5. BookViewController 5.16 Webservices
4.4.2. Remove from Readlist	5.7. ListViewController 5.16 Webservices
4.4.3. Display Read Readlist	5.7. ListViewController
4.4.4. Display ToBeRead Readlist	5.7. ListViewController
4.4.5. Share Readlist	5.10. ShareViewController
4.5.1. Register	5.12. RegisterViewController 5.16 Webservices
4.5.2. Login	5.13. LoginViewController
4.5.3. Forgot Password	5.13. LoginViewController 5.16 Webservices

Use Case	Matching Component
4.5.4. Logout	5.14. SettingsViewController
4.5.5. Change Password	5.14. SettingsViewController 5.16 Webservices
4.5.6. Synchronize	5.3. SynchronizationService 5.16 Webservices
4.6.1. Lend	5.9. TrackViewController 5.16 Webservices
4.6.2. Receive	5.9. TrackViewController 5.16 Webservices
4.7.1. Search with a Keyword	5.11. SearchViewContoller
4.7.2. Search with Barcode Scan	5.11. SearchViewController
4.8.1. Rate	5.5. BookViewController 5.16 Webservices
4.8.2. Review	5.5. BookViewController 5.16 Webservices
4.9.1. Get Recommendation	5.2. BookInformationService
4.9.2. Recommend Book	5.2. BookInformationService
4.10.1. Shop Online	5.5. BookViewController