

**2.2 Design Specification**

**Kindler**

Team README.md

I-Ning Chang

Andrew Jiang

Sehajpreet Kand

Eduardo Marin

Kuan-Yi Wu

**Table of Contents**

1. **Project Title and Authors ……………………………… 3**
2. **Preface …………………………………………………... 3**
3. **Introduction …………………………………………….. 3**
4. **Architecture Design…….………………………………. 3**
5. **Detailed Design………………………………………….12**
6. **Project Title and Authors**
   1. **Team Number:** 18
   2. **Team Name:** README.md
   3. **Team Members:**

Team Member USC ID Number Email

I-Ning Chang 9142146887 iningcha@usc.edu

Andrew Jiang 5770082146 andrewji@usc.edu

Sehajpreet Kand 2732926164 kand@usc.edu

Eduardo Marin 3842524453 eduarddm@usc.edu

Kuan-Yi Wu 1503666162 kuanyiwu@usc.edu

1. **Preface**

This software document is intended to serve as a record for the detailed breakdown of the high-level architecture required to deploy the mobile application Kindler. The document is divided into sections of introduction, architecture design, and detailed design of class diagrams and sequence diagrams. The primary function of the document is to inform and receive approval from the client. It will also serve as an important document to guide the developers through the implementation of the application.

1. **Introduction**

Kindler intends to provide users with an easy way to access a variety of books directly from their phones. Kindler achieves this by allowing users to perform five main functions: browse books, search books, like books (put in wishlist), post books (put in My List) and exchange books. Browsing books allows users to view books the Kindler community has to offer. Search books allows users to search for books they already have in mind. Liking books allows users to show they are interested in making exchanges with other members. Posting books allows users to show the Kindler community their book collection. And exchanging books allows two users to borrow books from one another. Unlike its competitors, Kindler gives users the best of both worlds by allowing users to access a large collection of books, for free, while offering functionality similar to apps like Goodreads.

1. **Architecture Design**
   1. **Major Components of the System**

* **User Interface:** The first layer that interacts with the user. This is going to be the outer appearance that the client sees and can interact with.
* **Presentation/Transition Logic:** This is going to be the logic in which buttons and icons should transition to a different UI.
* **Activity/Logic:** Activities in this layer run after user makes some interaction with the presentational/UI level.
  + **Searching Logic:** runs when user chooses to search for a book
  + **Authentication:** runs when user wants to log in, out/sign in, out
  + **Getting Messages:** Fetching messages that user receives
  + **Rendering Data:** Fetching from database, and rendering result
  + **Local Data:** Local Data is stored in this layer to assist in basic logic and rendering of views (such as strings, arrays, and pictures stored).
* **Database:** Queries made from the activity layer. This layer will only run to search in the whole Kindler, and a callback function will be called to return the inquired information back to the activity layer.
  1. **Components to Requirements**
* **User Interface:**
  + Will display pages
  + Will allow user to interact with the features provided by different pages
    - All pages
      * Click on buttons that appear on page
    - Login and Registration Page
      * Enter username and password
      * Enter short bio
      * Enter a book owned by user
      * Enter a book user wants
    - User Profile Page
      * Tap on profile picture to enlarge picture
      * Scroll through list of books user owns
      * Scroll through list of books user wants
      * Enter new bio for current user
      * Enter new name for current user
      * Change profile picture
    - Discover Page
      * Swipe right to add book list of books user wants
      * Swipe left to remove book from discover page
    - Search Page
      * Select search filter
      * Enter search term
      * Add book to list of books user owns
      * Add book to list of books user wants
    - User’s “My Books” page
      * Allow user to remove books they own from the exchange-matching algorithm
      * Allow user to indicate that a book has already been exchanged or traded
      * Allow user to remove book from list of books user owns
    - User’s “Wishlist” page
      * Allow user to remove books they want from the exchange-matching algorithm
      * Allow user to remove book from list of books user wants
    - User’s Exchange
      * Allow user to indicate that they want to trade with another user
      * Allow user to send message to another user with whom they wish to trade
* **Presentation/Transition Logic:**
  + Allows user to navigate between different pages of the application
    - Login and Registration Page
      * When user presses login or signup button
        + User is redirected to Discover page if the login or sign up information is valid
        + User is redirected back to Login and Registration page if login or sign up information is invalid
    - Discover Page
      * When user presses “My Books” button
        + User is redirected to “My Books” page
      * When user presses “Wishlist” button
        + User is redirected to “Wishlist” page
      * When user presses “Profile” button
        + User is redirected to “Profile” page
      * When user presses “Search” button
        + User is redirected to “Search” page
      * When user presses “Exchange” button
        + User is redirected to Exchange page
    - All other pages
      * When user presses “Discover” button
        + User is redirected to “Discover” page
  + Updates page view when user presses on a button or performs an action
    - Login and Registration Page
      * When user presses login or sign up button
        + Page displays loading bar while user information is validated
    - Profile Page
      * When user presses bio edit button
        + A text form will appear with current bio
      * When user presses name edit button
        + A text form will appear with current name
      * When user presses profile picture edit button
        + User profile picture will change to user’s current facebook profile picture
    - Discover Page
      * When user swipes either left or right
        + Page will display a new book
    - Search Page
      * When user presses search button
        + Page will display all books that return from query
    - “My Books” Page
      * When user presses remove book button
        + Page will no longer show removed book
    - “Wishlist” Page
      * When user presses remove book button
        + Page will no longer show removed book
    - Exchange Page
      * When presses on a conversation
        + Page will display conversation
      * When send message button is pressed
        + Text in text form will be added to conversation
        + Text in text form will be cleared after it is added to conversation
* **Activity/Logic**
  + Retrieves user input from User Interface layer and edits Local Data layer
  + Retrieves data from Local Data layer and sends data up to User Interface Layer
  + **Searching Logic**
    - Retrieves a pair of strings from User Interface layer which contains a search term and search type
      * Performs search in Local Data layer
    - Retrieves an array of strings from Local Data layer with query results and sends array of strings to User Interface layer
  + **Authentication**
    - Retrieves an array of strings from User Interface layer where the last string is the authentication type i.e. either “login” or “signup”
      * If authentication type is login
        + First string in array is username and second string in array is password
        + Search for username password pair in Local Data layer

If username password pair is not found in Local Data layer make User Interface layer redirect to Login and Registration Page while displaying appropriate error message

If username password pair is found in Local Data layer make User Interface layer redirect to Discover Page

* + - * If authentication type is signup
        + First string in array is username, second string in array is password, and third string in array is user bio
        + Search for username password pair in Local Data layer

If username password pair is not found in Local Data layer add new user to Local Data layer and make User Interface layer redirect to Discover Page

If username and password pair is found in Local Data layer make User Interface layer redirect to Login and Registration Page while displaying appropriate error message

* + **Getting Messages**
    - Retrieves an array of strings from User Interface layer where last string is the message search type i.e. either “all” or “one”
      * If message search type is all
        + First string in array is user ID of current user
        + Search for conversations that belong to user ID in Local Data layer
        + Return an array of conversations that were found to User Interface layer and display conversations in Exchange Page
      * If message search type is one
        + First string in array is user ID of current user and second string is user ID of contact
        + Search for conversation between current user and contact in Local Data layer using user ID’s
        + Return conversation between current user and contact to User Interface layer and display conversation in Exchange Page
  + **Rendering Data**
    - Retrieves an array of strings from User Interface layer where last string identifies page i.e. “mybooks”, “wishlist”, or “discover”
      * If page is mybooks
        + Second to last string in array list identifies action i.e. either “add”, “remove”, or “get”
        + First string in array list identifies User ID
        + Second string in array list identifies Book ID

If action is add

Add book that belongs to Book ID to the book list that belongs to the user with User ID

If action is remove

Remove book that belongs to Book ID from the book list that belongs to the user with User ID

If action is get

Search for book list that belongs to user with User ID and return to User Interface layer

* + - * If page is wishlist
        + Second to last string in array list identifies action i.e. either “add”, “remove”, or “get”
        + First string in array list identifies User ID
        + Second string in array list identifies Book ID

If action is add

Add book that belongs to Book ID to the wishlist that belongs to the user with User ID

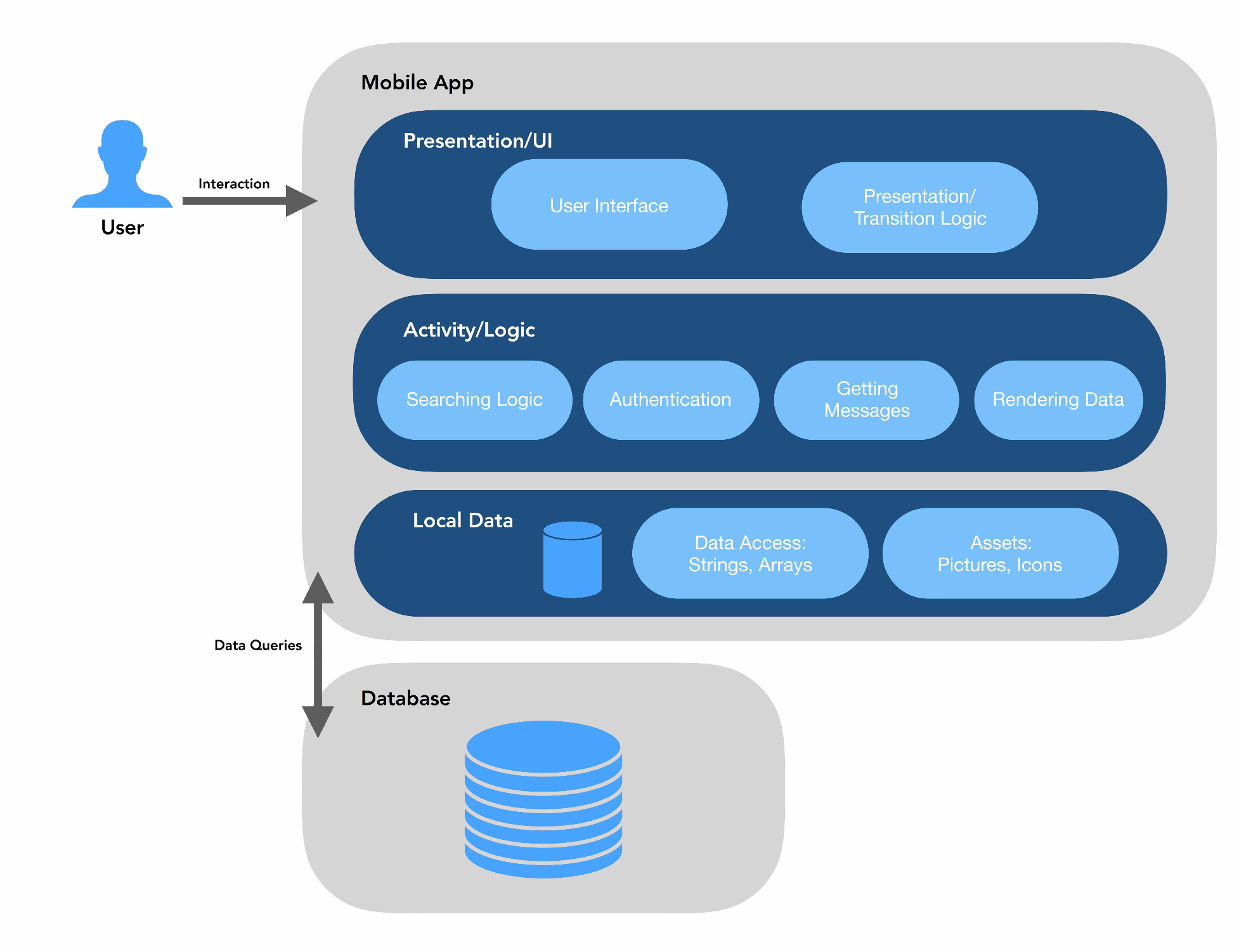
If action is remove

Remove book that belongs to Book ID from the wishlist that belongs to the user with User ID

If action is get

Search for wishlist that belongs to user with User ID and return to User Interface layer

* + - * If page is discover
        + First string in array list identifies User ID
        + Search Local Data for all books that user has not seen and are not a part of user’s “My Books” and “Wishlist”
        + Return results to User Interface layer for display
* **Local Data**
  + Stores information that belongs to application i.e. assets, icons, pictures
  + Stores information that belongs to current user i.e. username, password, book lists, etc.
  + Retrieves data from Database Layer when information is not found and sends data up to Activity Logic Layer
* **Database**
  + Searches for queries and returns results to Local Data layer
  + Remove data from database
  + Add data to database
  1. **Block Diagram**

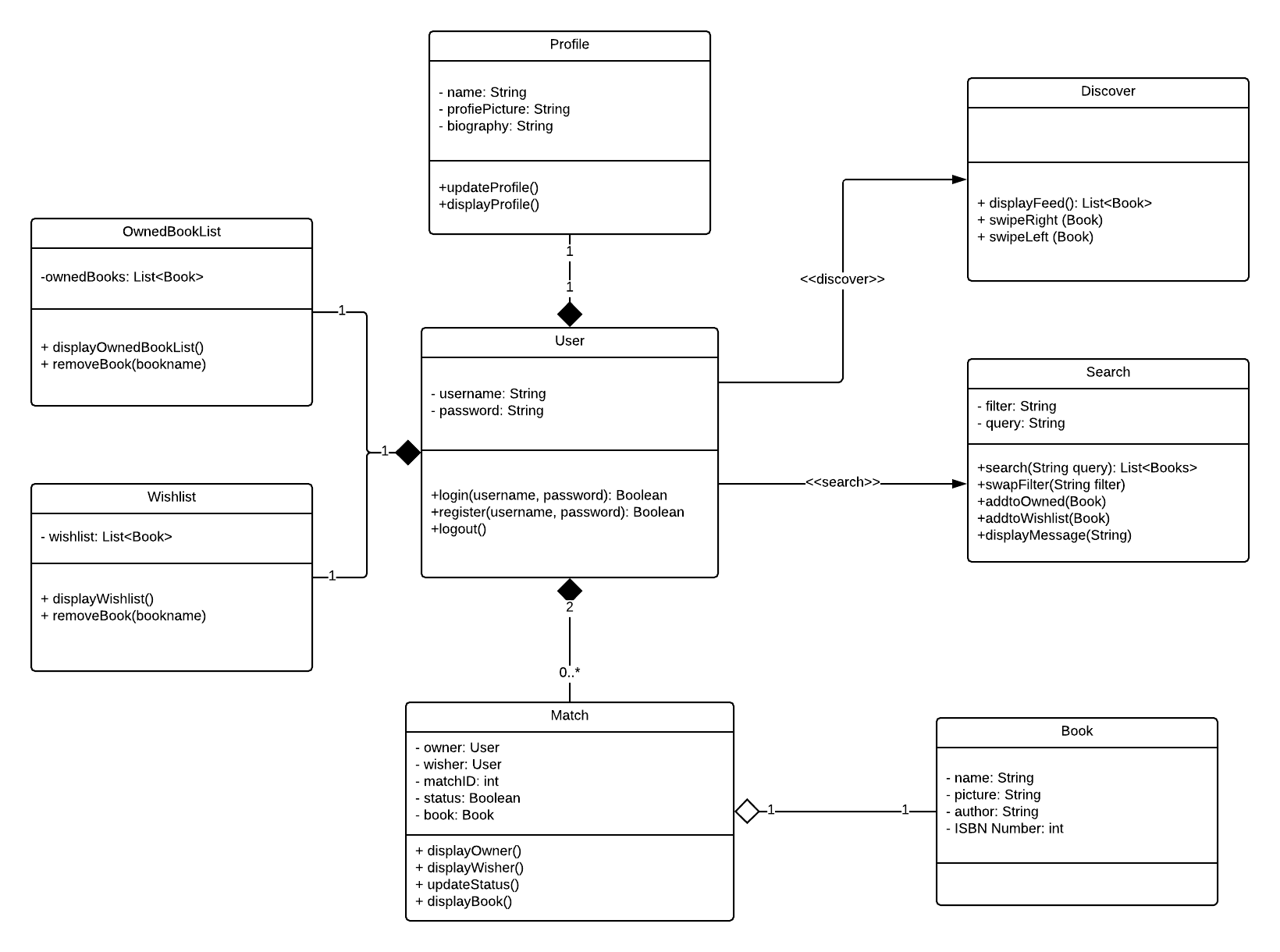
****

* 1. **Rationale for the Architectural Configuration**

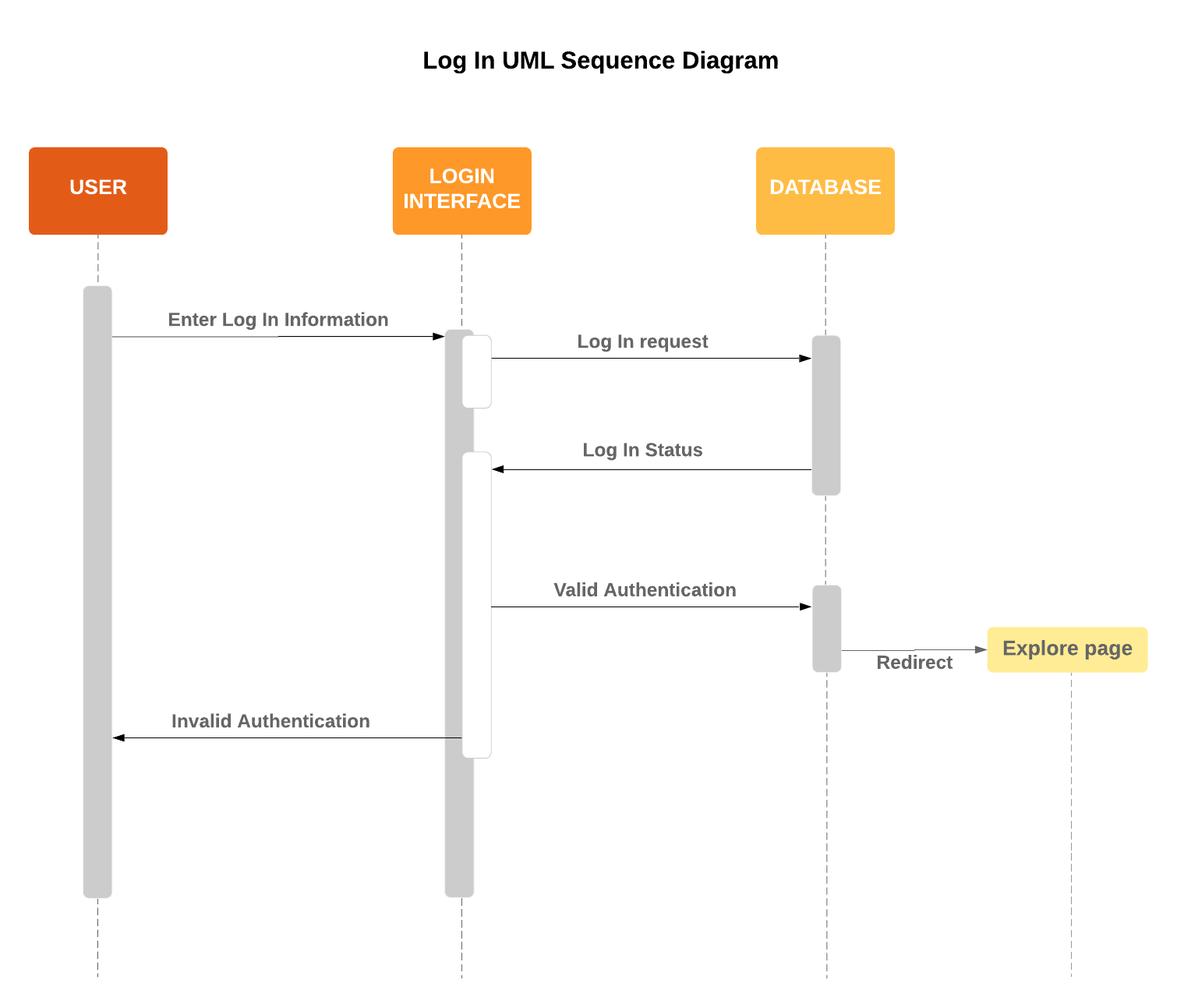
We chose to configure our application using a layered architecture because we want our app to be maintainable throughout the software engineering process. Since each layer only interacts with the layer above and the layer below any and all changes that occur throughout our development process will affect at most two other layers instead of the entire application. In addition to the above mentioned, we also decided to use a layered architecture because using a layered architecture helps improve the security of our application. Since each layer only interacts with adjacent layers, users cannot directly access our database so our system is more secure.

We selected our components by viewing the requirements our app needed to meet. We noticed that our app needed a method for the user to receive and input information so it seemed like a good choice to make a component solely dedicated to the User Interface. Then we noticed that our app only allows the user to input login/signup information, a book search, or a message, so we decided to make a component that received all the types of user input from the user interface and returned the appropriate data. Since each type of user input was used to perform a different type of data search in our app we decided to group all those different searches into the Activity/Logic component. Next, we decided to add a Local Data component to store information that is relevant to the current user in order to prevent unnecessary calls to the database. This is especially important because most of our user’s actions result in a search. Lastly we decided to make the database its own component because it allows the Local Data component to retrieve and update its information.

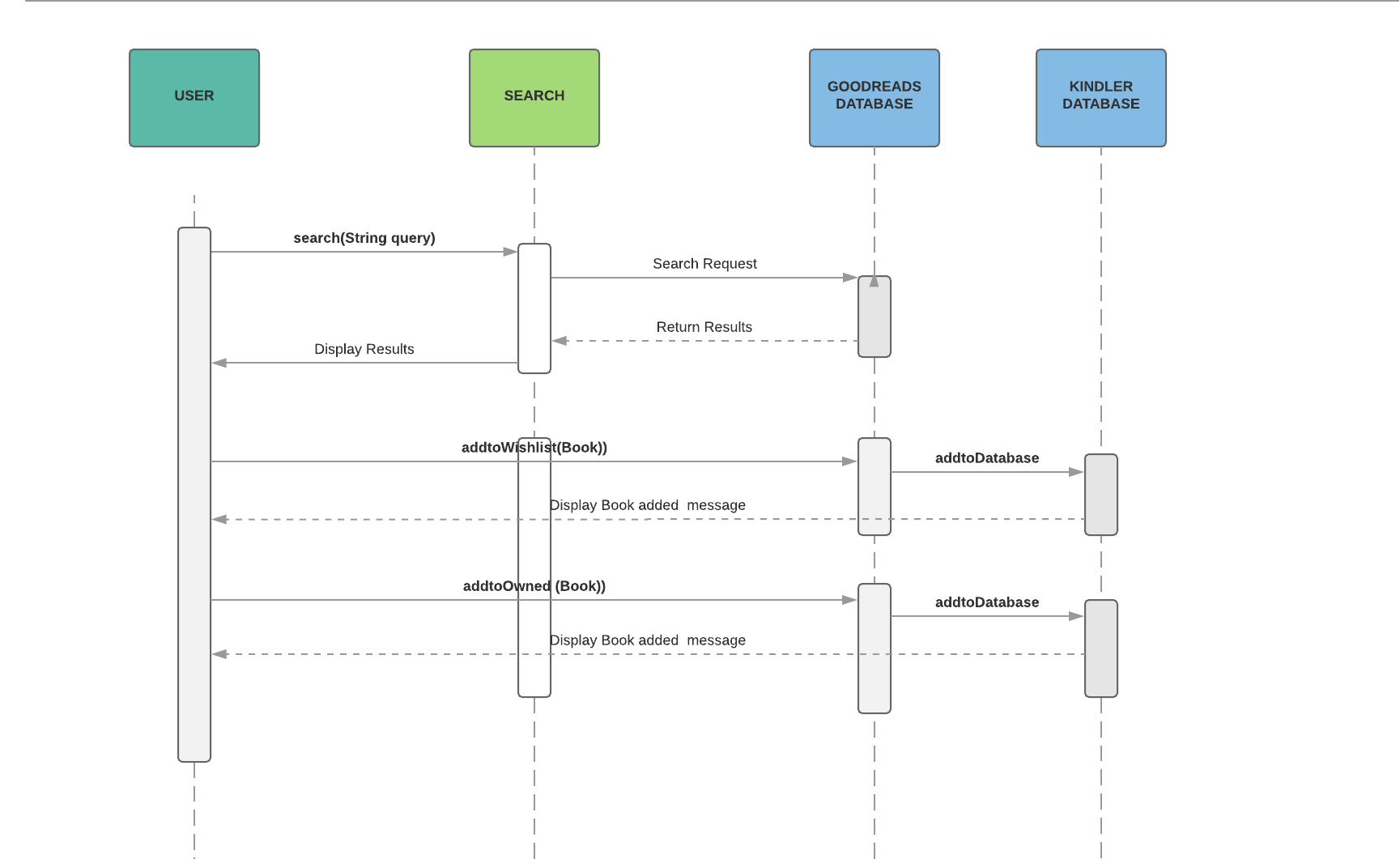
1. **Detailed Design**
   1. **UML Class Diagram**



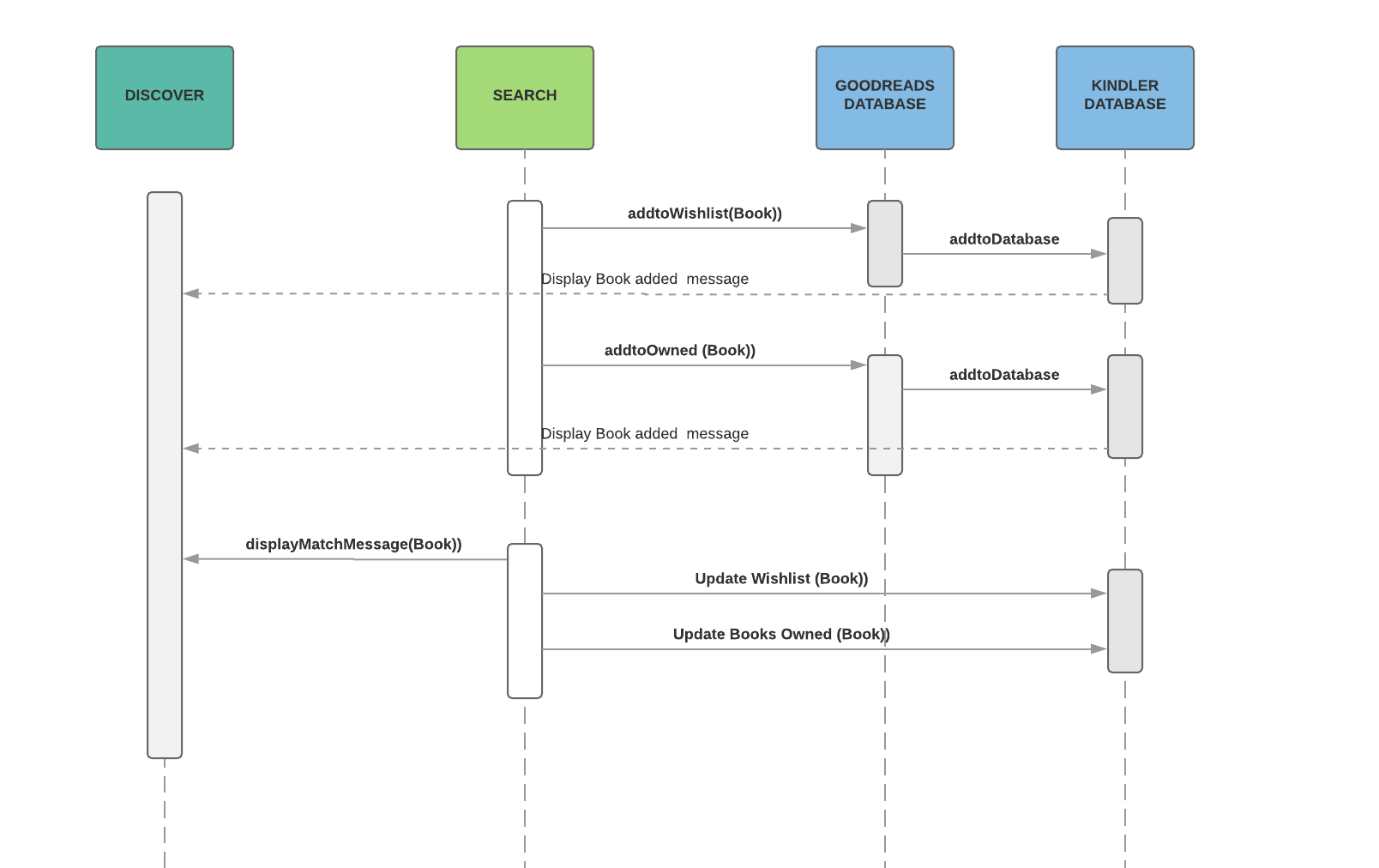
* All the classes listed here belong to the mobile app layer of the architecture design.
* User class corresponds to both UI and Activity sub-layers as the user is authenticated through this class.
* Search class corresponds to the Activity sub-layer, and in particular, the Searching Logic component.
* Match class corresponds to the Activity sub-layer, and in particular, the getting messages component.
* OwnedBookList, WishList, Profile, Discover, and the Book class all corresponds to the Activity sub-layer in the rendering data component. This component works with either local data or makes api call to the database.
  1. **UML Sequence Diagrams**



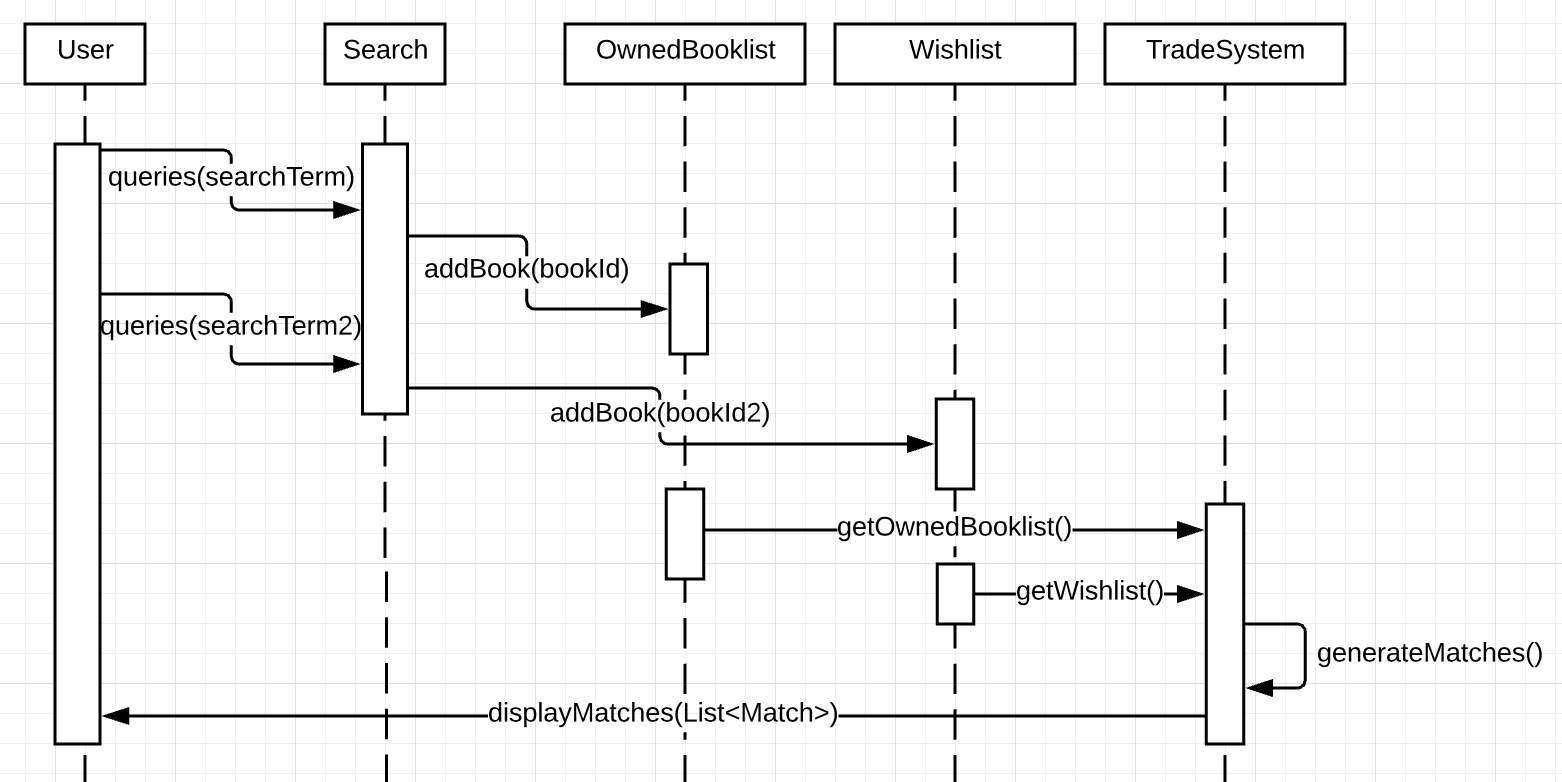
**Search UML Sequence Diagram**



**Match UML Sequence Diagram**



**Adding Books UML Sequence Diagram**



**Trading Books UML Sequence Diagram**

