

**Khulna University of Engineering & Technology**

**Department of Computer Science and Engineering**

**Project Report on CSE 3218**

**Course Title:** Mobile Computing Laboratory

**Project Title:** BookIsh

| **Submitted by:** | **Submitted to:** |
| --- | --- |
| Name:Nusrat Jahan Jonaki  Roll: 1907067  Name: Nishat Tasnim Mahi  Roll: 1907072  Name: Sumiya Islam Barsha  Roll: 1907076  Name: Samin Saiara  Roll: 1907078  Name: Anisa Walida  Roll: 1907087 | **Most. Kaniz Fatema Isha** Lecturer  Computer Science and Engineering  **Argha Chandra Dhar** Lecturer  Computer Science and Engineering |

**Introduction**

“BookIsh” is an iOS app that is mainly a reader’s diary for bookworms.It can be useful for booklovers if they have a list of the books they have already read or they want to read or the books they are currently reading. This app also enables users to find new books according to their favorite author or genre. to have an overview of his/her favorite book or books to be read.

The first feature implemented on this app is Email Authentication of Firebase Database. Then the user’s info was stored and retrieved from firebase realtime database. The wishlist option enables the user to keep track of the books they want to read. Another feature implemented was getting the list of books, which was done by calling iTunes API. The use can also search for desired book in this app.

**Methodology**

First, the system has to be logged into. On the first page (Home page) there will appear a search box where the users can search for books of their interest. After getting the desired book, the user will finally see the complete description of the book. In that description section, there will be the title, author’s name, book cover, publication year, date and rating. This task was done by calling the API, here the API was called from iTunes. The Search API allows to place search fields in app to search for content within the iTunes Store and Apple Books Store. Moreover, we’ve used firebase as our database. Besides, we have two options. One is the bookshelf option and the other one is the wish-list option. The main purpose of the bookshelf option is to track down the books that the user already read, and the wish-list option enables the users to have a record of the book that they want to read. These lists can be dynamically changed using the database. Swift has been used to develop the project. Swift is a programming language developed by Apple Inc. for iOS and OS X development. Swift adopts the best of C and Objective-C, without the constraints of C compatibility.

**Book Struct -**

struct Book: Identifiable, Decodable, Hashable {

var id: Int

var title: String

var author: String

var coverImageURL: URL?

var averageRating: Double?

var genres: [String]?

var description: String?

var releaseDate: String?

...

}

The Book struct represents a book in the BookIsh2.0 application. It includes properties such as id, title, author, coverImageURL, averageRating, genres, description, and releaseDate. The struct conforms to the Identifiable, Decodable, and Hashable protocols, allowing it to be used in SwiftUI views and easily decoded from external data sources. The CodingKeys enum is used to map struct properties to corresponding keys in the decoding process, following the format expected from an iTunes API response. The struct is a fundamental part of the app's data model, capturing essential information about books for display and interaction within the application.

**BookFetcher Class -**

**class BookFetcher: ObservableObject {**

**@Published var books = [Book]()**

**func searchBooks(searchTerm: String) {**

**...**

**}**

**func loadBooks() {**

**leturl=URL(string:"https://itunes.apple.com/search?term=books&entity=ebook&limit=50")**

**...**

**}**

**...**

**}**

The BookFetcher class is a data manager responsible for fetching and storing book information from the iTunes API. It features a @Published array of Book objects for dynamic updates in SwiftUI views. The searchBooks method queries the API based on a search term, while loadBooks fetches a default set. Fetched data is decoded into a BooksResponse struct, and the books array is updated on the main thread for seamless integration with SwiftUI views, reflecting the application's book data.

**AuthorBook Fetcher -**

class AuthorBookFetcher: ObservableObject {

@Published var books = [AuthorBook]()

...

func searchBooks(byAuthor author: String) {

var urlComps = URLComponents(string: "https://itunes.apple.com/search")!

...

}

}

The AuthorBookFetcher class manages the asynchronous retrieval and storage of book data from the iTunes API for a specific author. The class, marked as ObservableObject, features a @Published array of AuthorBook objects for automatic updates in SwiftUI views. The searchBooks(byAuthor:) method constructs a URL for author-specific book searches, utilizes Combine's URLSession.shared.dataTaskPublisher for asynchronous data fetching, and updates the books array upon successful decoding of the response on the main thread. The class employs Combine's publishers and subscribers to streamline asynchronous operations and provide real-time updates to SwiftUI views.

**CurrentReadViewModel -**

**class CurrentReadViewModel: ObservableObject {**

**@Published var books = [Book]()**

**func fetchCurrentreadBooks(userID: String) {**

**db.child("users").child(userID).child("currentread").observe(.value, with:**

**...**

**}**

**...**

**}**

The CurrentReadViewModel is a SwiftUI ObservableObject managing the fetching of books from a user's "currentread" list in Firebase, using a DatabaseReference. It observes changes to the database and updates the books array accordingly. The CurrentReadViewModel displays the fetched books in a List with each book's title and author, and is initialized with an instance of CurrentReadViewModel.

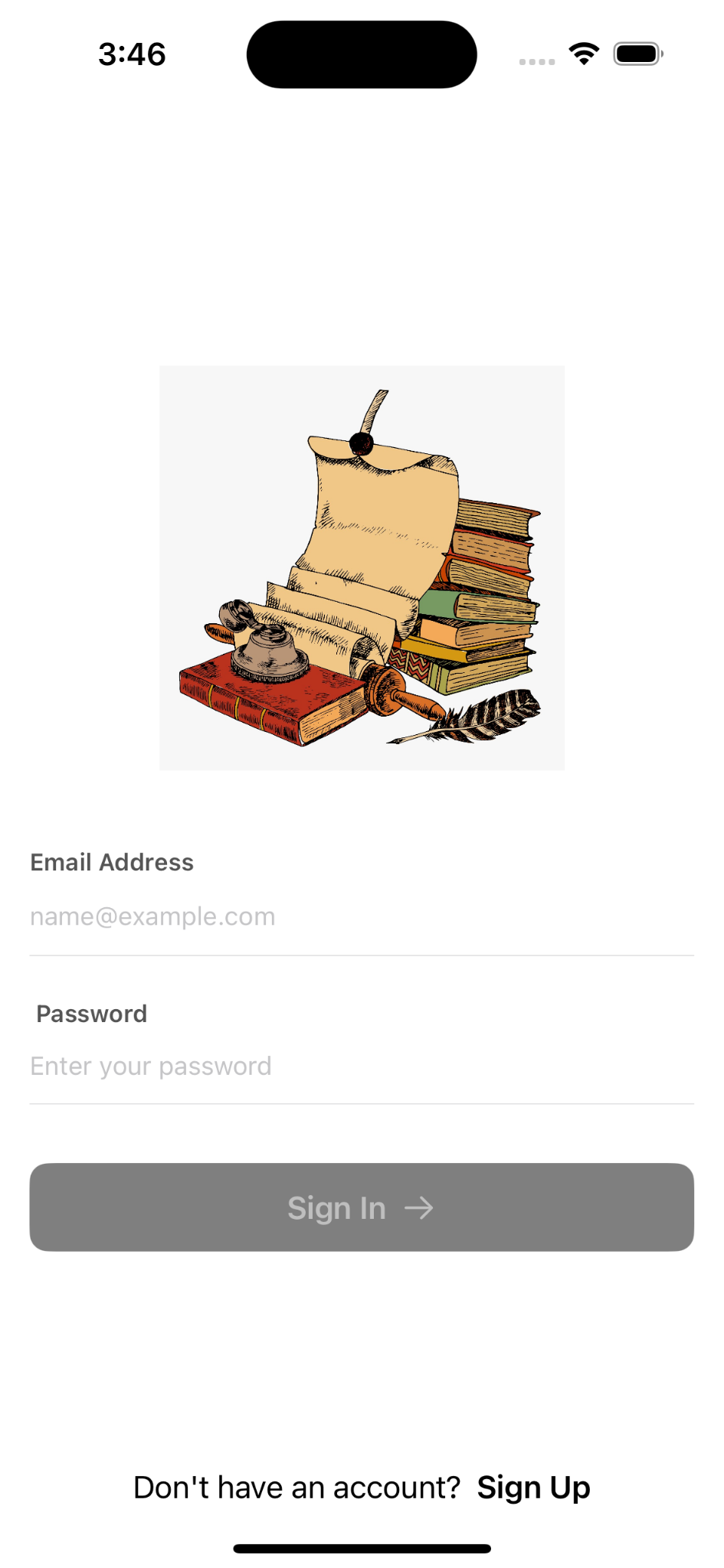
Same goes for HaveReadViewModel and WishListViewModel files.

**Result**

The detailed implementation of the iOS application named “BookIsh” is stated below-

**Login**

Once the application is loaded, user will see the login page, and then after inserting their email and password on the respective fields, they will log in to the main application, and will be redirected to the homepage. If they are not registered yet, they can sign up by clicking below.

 Figure 1: Login Page

**Register**

The user will be redirected to the register page. In the register page, the user will have to enter their email ID , name password and confirm password which is at least six characters long to register their account. Their account information will be saved on the Firebase.

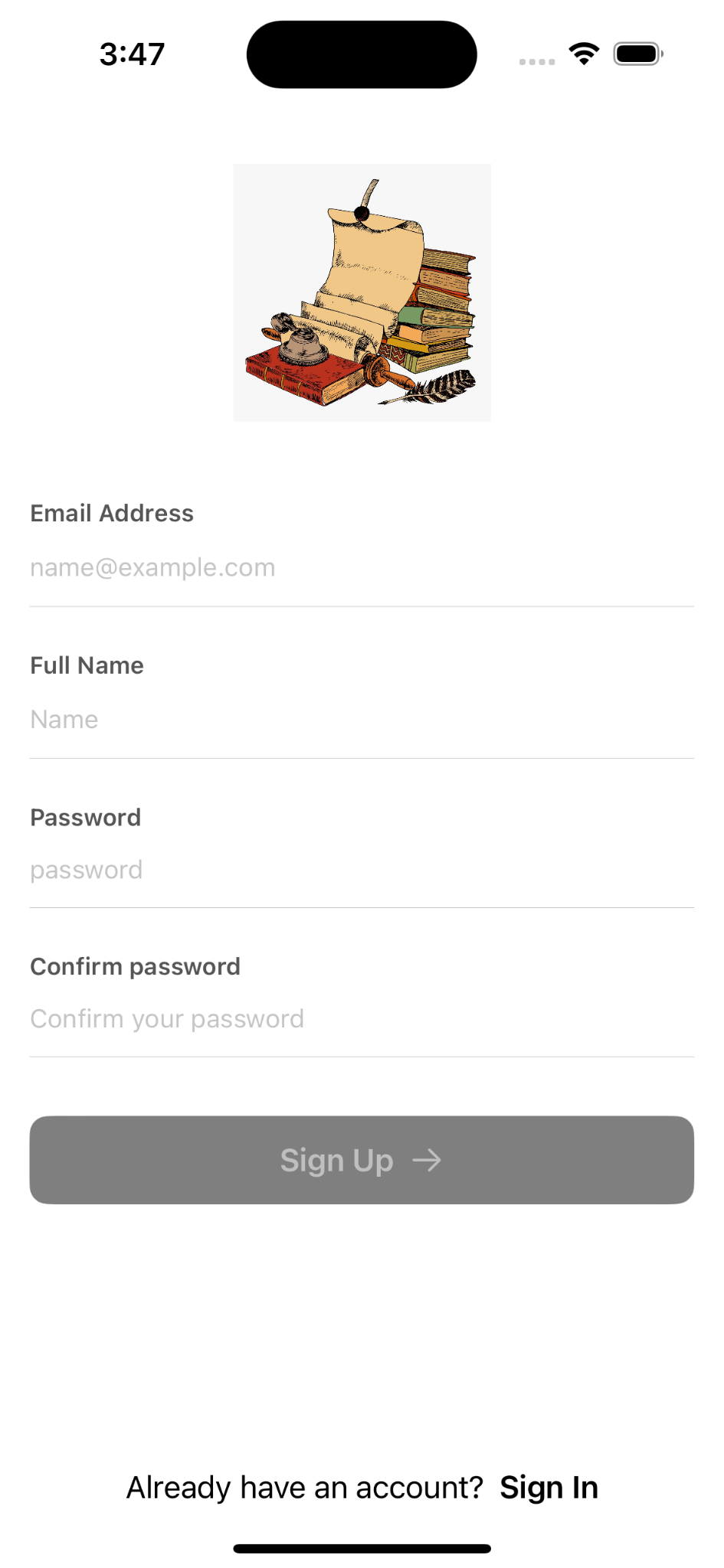


Figure 2: Sign Up Page

**Homepage**

After login, the user will be redirected to the homepage where they can search for the book they were looking for. There is a bottom navigation bar containing four fields - home, author, status and genre. On the top right there is profile option.

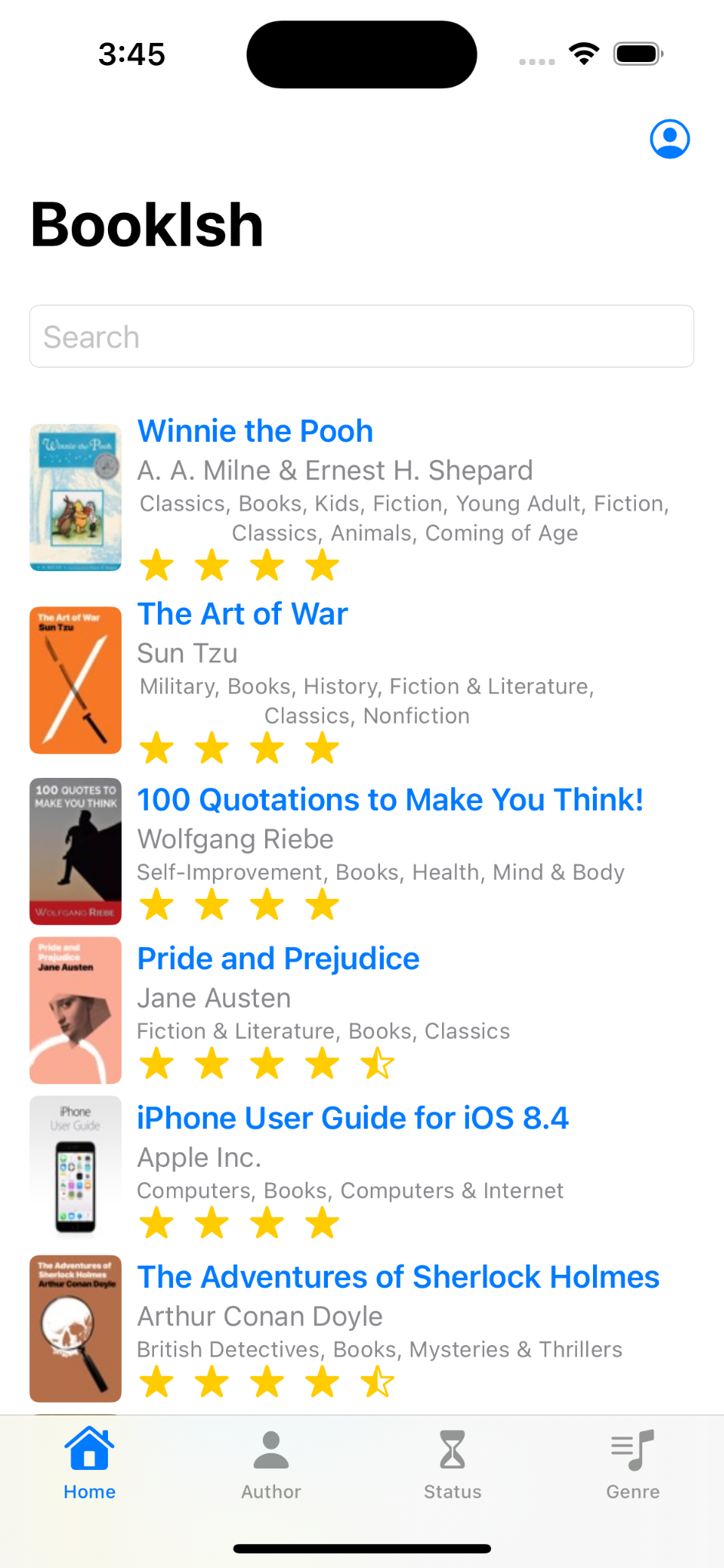


Figure 3: Homepage

**Book details**

Choosing the book, the user then will be redirected to the book’s details page where information about the book such as book name, author name, rating, book description, etc. will be displayed. On the details page, the user will be able to add the book to “Have read” as a book they have already read, and to “Wishlist” , the books they want to read in the future and lastly “Currently Reading”, the books they are reading right now.

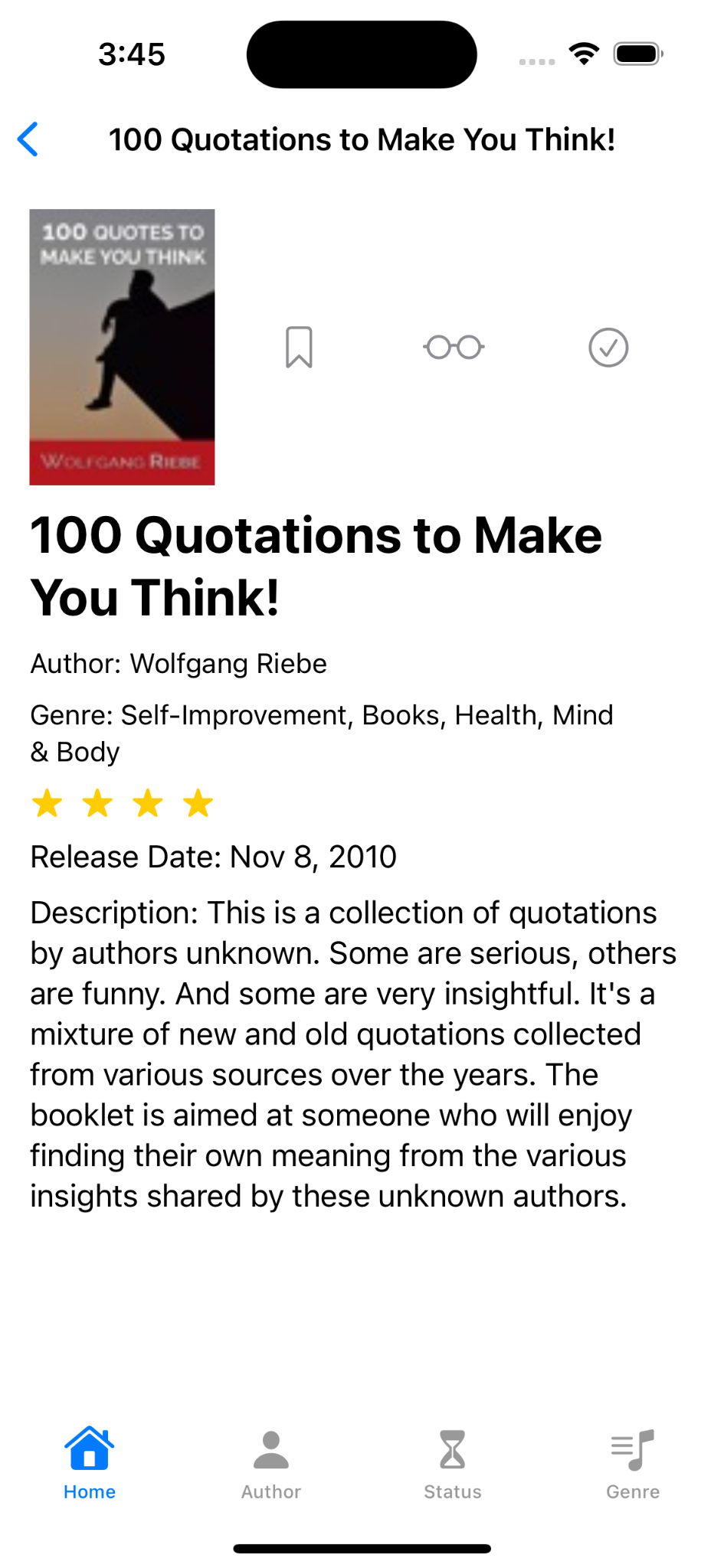
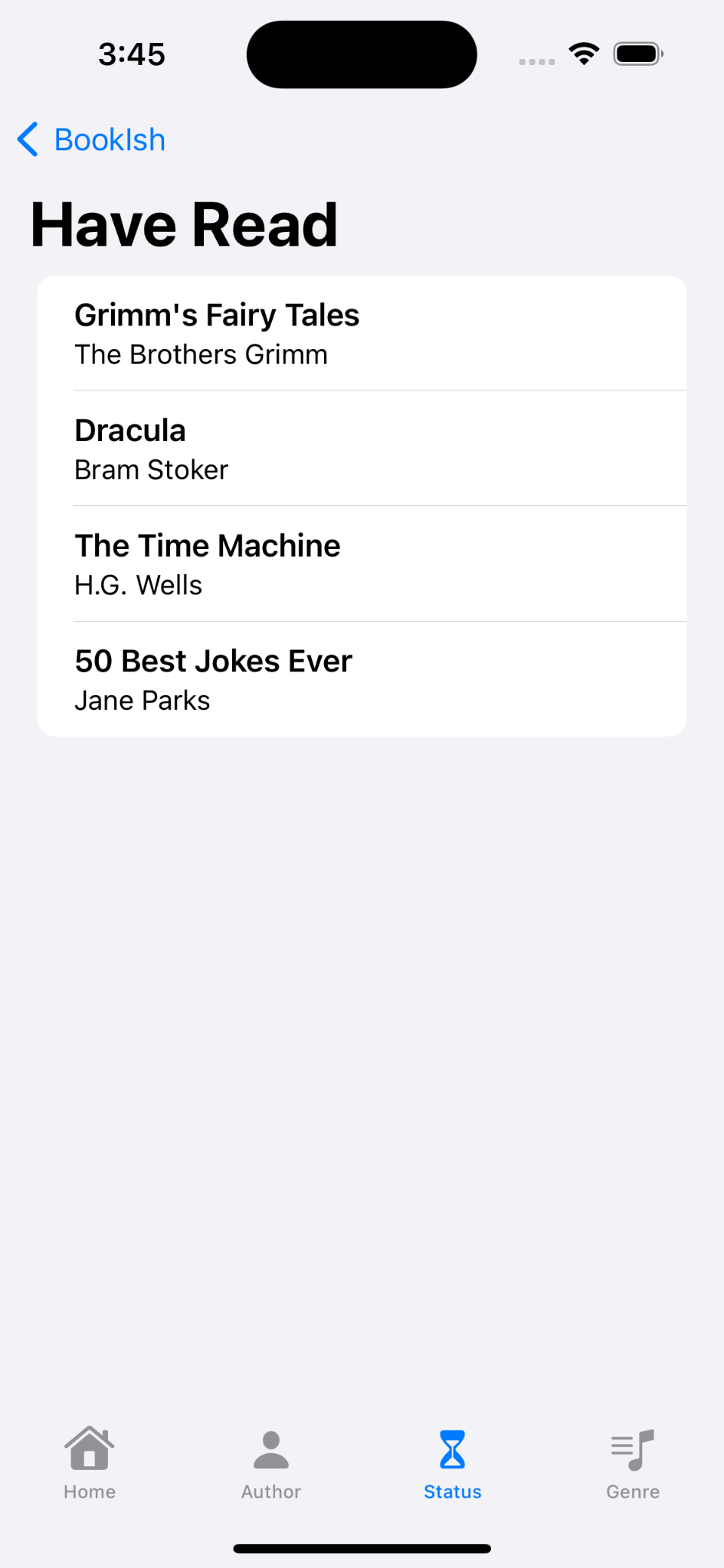
** **

Figure 4: Book Details and Have Read Status

**Genre**

On selecting the Genre, there will be a list of different genres.User will be able to see lists of books according to the selected genre.



Figure 5:Genre List

**Author**

User can search by their favorite author.

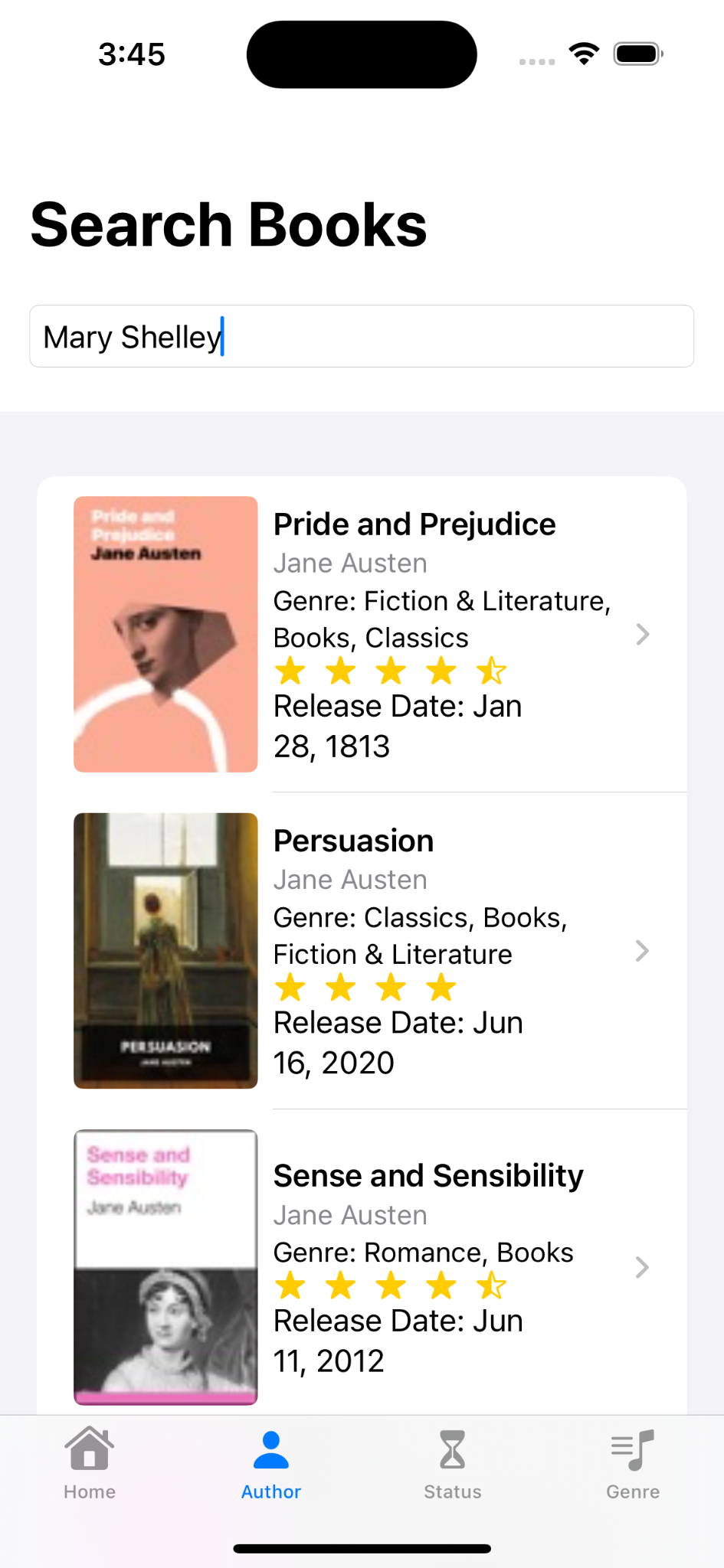


Figure 6: Author

**Discussion**

The project is all about quenching the thirst of book lovers. It ensures the users get an overview of the book and have a record of already read books and books to be read. We faced quite a lot of difficulties while developing the project. First of all, the X-code interface is not user-friendly for people who use windows. Moreover, connecting the app to the firebase database is challenging. On the other hand, designing UI is a complex task keeping all the constraints in mind, and finally, we faced difficulties when we were trying to develop using storyboard. And we had to switch to swiftUI having no option left.

**Conclusion and Future Work**

In conclusion we can say that we faced many issues as beginners in making iOS applications. Although we had added most of the features we wanted to, we were unable to add user review dynamically. In future, a community feature can be added to make the app more diverse and useful.

**References**

* The Basics - The Swift Programming Language

(<https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html>)

* Swift Tutorial (<https://www.tutorialspoint.com/swift/index.htm>)
* [COMPLETE User Login / Sign Up App | Swift UI + Firebase | Async / Await](https://youtu.be/QJHmhLGv-_0)
* https://firebase.google.com/docs/database