"BOOK SHARING"

A Smart Web Application to Communicate Reader

Prepared by:

Mahmud Al Saeed

ID: 151-35-1121

B.Sc. in Software Engineering

Daffodil International University

MD. Asif Adham

ID: 151-35-1044

B.Sc. in Software Engineering

Daffodil International University

Supervised By

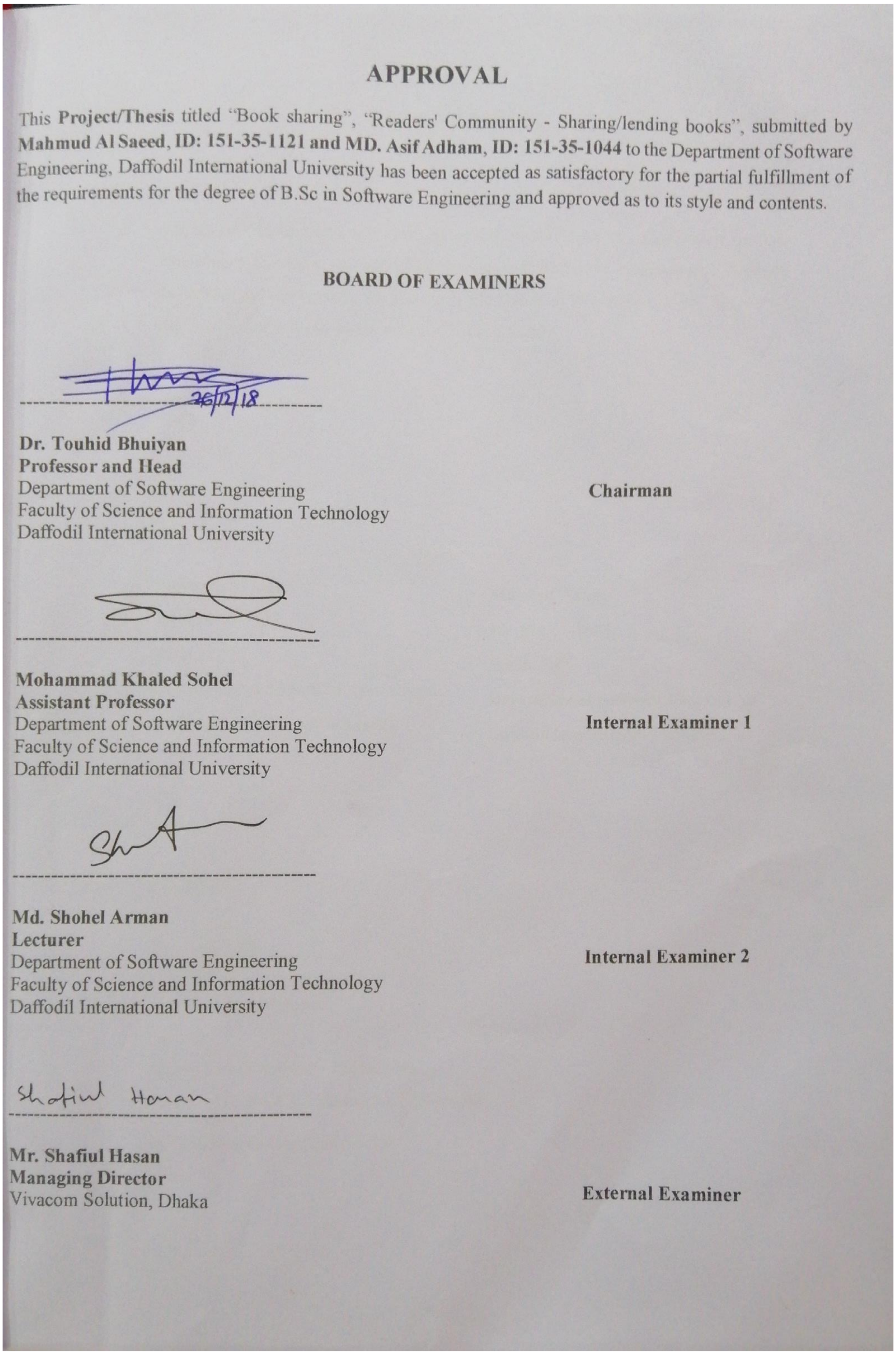
Md. Mushfiqur Rahman

Lecturer

Department of Software Engineering

Faculty of Science & Information Technology

Daffodil International University



APPROVAL

This **Project/Thesis** titled "Book sharing", "Readers' Community - Sharing/lending books", submitted by **Mahmud Al Saeed, ID: 151-35-1121 and MD. Asif Adham, ID: 151-35-1044** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc in Software Engineering and approved as to its style and contents.

BOARD OF EXAMINERS

[illegible] 26/12/18

Dr. Touhid Bhuiyan
Professor and Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Chairman

[illegible]

Mohammad Khaled Sohel
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

[illegible]

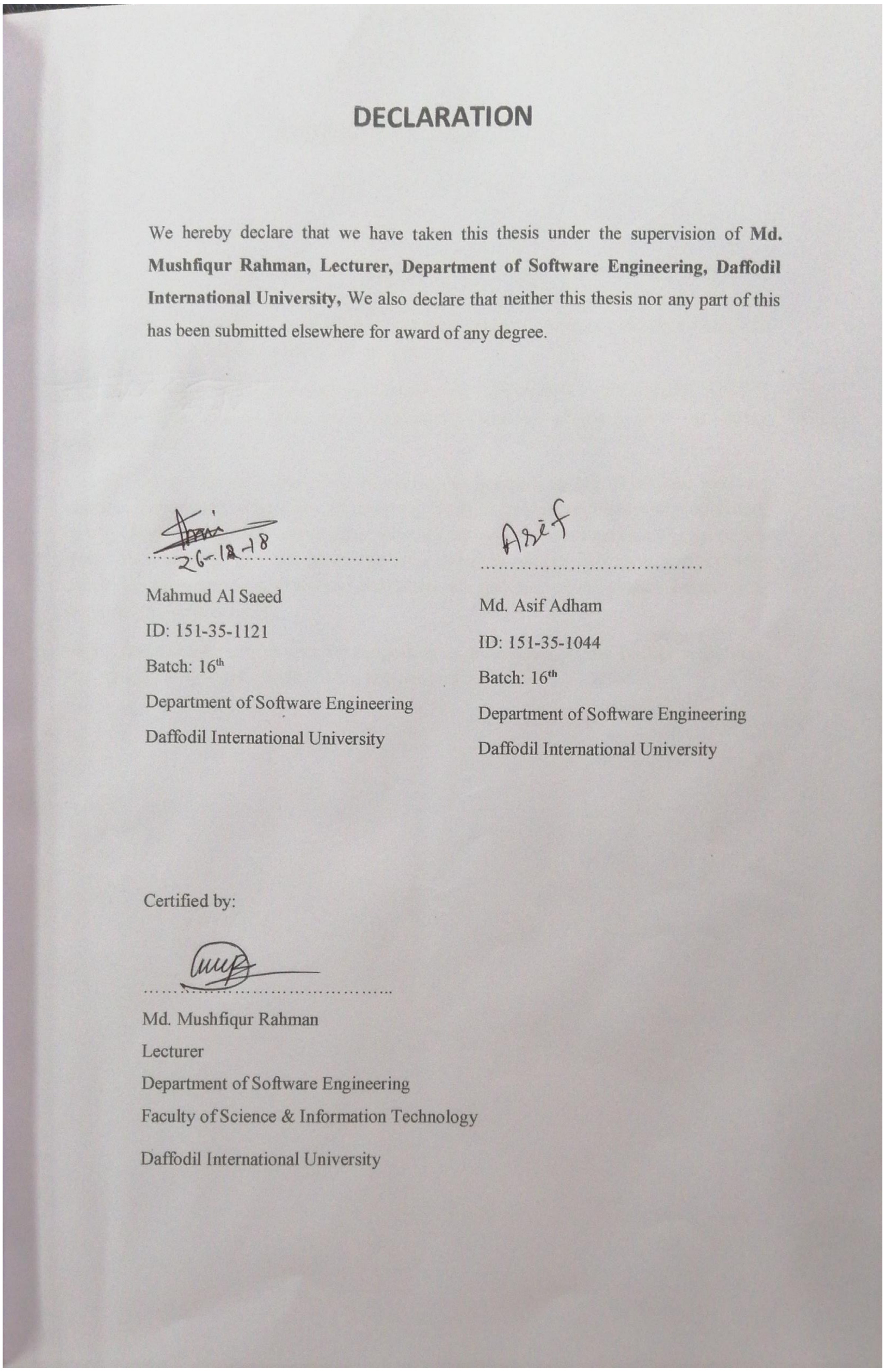
Md. Shohel Arman
Lecturer
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Shafiul Hasan

Mr. Shafiul Hasan
Managing Director
Vivacom Solution, Dhaka

External Examiner



DECLARATION

We hereby declare that we have taken this thesis under the supervision of **Md. Mushfiqur Rahman, Lecturer, Department of Software Engineering, Daffodil International University,** We also declare that neither this thesis nor any part of this has been submitted elsewhere for award of any degree.

[illegible] 26-12-18

………………………………………

Mahmud Al Saeed

ID: 151-35-1121

Batch: 16th

Department of Software Engineering

Daffodil International University

Asif

………………………………………

Md. Asif Adham

ID: 151-35-1044

Batch: 16th

Department of Software Engineering

Daffodil International University

Certified by:

[illegible]

………………………………………..

Md. Mushfiqur Rahman

Lecturer

Department of Software Engineering

Faculty of Science & Information Technology

Daffodil International University

ACKNOWLEDGEMENT

It stands to reason that the completion of main project needs the support of many people. We take this opportunity to express our boundless thanks and commitment to each and every one, who helped us in successful completion of our main project. I am happy to acknowledge the help of all the individuals to fulfill our attempt.

First and foremost we wish to express wholehearted indebtedness to the almighty ALLAH for his gracious constant care and magnanimity showered blissfully over us during this Endeavour.

I would like to take the opportunity to express our gratitude to Md. Mushfiqur Rahman, Lecturer. Department of Software Engineering, Daffodil International University. Although he was always loaded with several other activities, he gave us more than enough time in this work. He not only gave us time, but also gives us proper guidance and valuable advice whenever we faced with some difficulties. His comments and guidance helped us in preparing our project report.

Finally, we thank to our parents for supporting us throughout all my studies at Daffodil International University. Thanks to our teacher and our friend

ABSTRACT

This project entitled “Book Sharing” is a web based application to exchange book with other. The Book of Life is the 'brain' of The School of Life, a gathering of the best ideas around wisdom and emotional intelligence. Marcus Tullius Cicero said “A room without books is like a body without a soul”. **Book Sharing** is the practice of a swap of books between one person and another. Practiced among book groups, friends and colleagues at work, it provides an inexpensive way for people to exchange books, find out about new books and obtain a new book to read without having to pay. Because swaps occur between individuals, without central distribution or warehousing, and without the copyright owner making a profit. In this era of internet everyone is connected virtually with each other. So we can exchange book with a web site. With this thought we made this project of Book sharing. Here a user can fide book, share book with other. He can exchange book with other user books. He can add his exchangeable book with picture. They can communicate with other by user profile address. It save money. It can build our relation and other. It will make each books lovers house a library accessible to all of our users.

We used Laravel 5.6 (PHP framework), HTML 5, CSS 3, Advanced CSS3, and Bootstrap for this website. We also used Client side scripting language: JavaScript to make it more user friendly.

LIST OF CONTENTS

Page

Title pages …………………………………………………………………………………. i

Declaration …………………………………………………………………………………ii

Acknowledgement …………………………………………………………………...iii

Abstract ……………………………………………………………………………………iv

List of content ……………………………………….……………………….……v

List of the figure ……………………………………………………………..........vii

List of the table ……………………………………………………………………..viii

Chapter

1. INTRODUCTION ……………………………………………………………1-2

   1.1 Purpose ……………………………………………………………………………1

   1.2 Background ……………………………………………………………………….1

   1.3 Motivation ………………………………………………………………………...1

   1.4 Objective ………………………………………………………………………….2

   1.5 Stakeholders ……………………………………………………………………2

2. REQUIREMENT SPECIFICATION AND ANALYSIS …………………………3-11

   2.1 System Analysis ………………………………………………………………..3

      2.1.1 Requirement Elicitation ………………………………………………...3

      2.1.2 Software Requirement Specification ……………………………………...4

   2.2 Used Technology ……………………………………………………………….6

   2.3 Project Schedule ………………………………………………………………7

3. SYSTEM DESIGN ……………………………………………………………12-18

3.1 Use Case Diagram ……………………………………………………………12
4. CODES ……………………………………………………………………..19-24
4.1 Front …………………………………………………………………………..19
4.1.1 Authentication ……………………………………………………...19
4.1.2 Home ………………………………………………………………….21
4.2 User …………………………………………………………………………...25
Chapter Page
4.2.1 Manage books …………………………………………………...25
4.2.2 Contacts …………………………………………………………………29
4.2.3 Manage exchange …………………………………………………..30
4.2.4 Manage profile and exchanges from the profile …………………………31
4.2.5 Manage suggestions …………………………………………………….40
5. SYSTEM TESTING ……………………………………………………..42-46
5.1 PHP Unit Test ……………………………………………………………...42
5.1.1 Module 1: user …………………………………………………………..42
5.1.2 Module 2: Favorite book ……………………………………………...45
6. USER MANUAL ……………………………………………………………...47-60
6.1 Guest User …………………………………………………………………47
6.2 Authenticate User ……………………………………………………………..48
7. CONCLUSION……………………………………………………………... 61
7.1 Future Scope ……………………………………………………………….61
7.2 Limitations ……………………………………………………………………61
Appendix ……………………………………………………………………....63-93

LIST OF FIGURE

Figure Page

Figure 2.1: Schedule chart ………………………………………………………………..13

Figure 3.1: Use case diagram ………………………………………………………….14

Figure 3.2: Activity diagram …………………………………………………………..15

Figure 3.3: Class diagram ……………………………………………………………..16

Figure 3.5: Schema diagram …………………………………………………………..19

Figure 3.4: Sequence diagram ………………………………………………………….20

Figure 6.1: Front hero ………………………………………………………………….44

Figure 6.2: Book cards………………………………………………………………....44

Figure 6.3: Book Details ………………………………………………………………..45

Figure 6.4: Sign In ……………………………………………………………………..46

Figure 6.5: Sign Up …………………………………………………………………….46

Figure 6.6: Request Book ……………………………………………………………...47

Figure 6.7: Select my books for offers ………………………………………………...47

Figure 6.8: exchange notification ……………………………………………………..48

Figure 6.9: Exchange Offers …………………………………………………………..49

Figure 6.10: Edit Book ………………………………………………………………..50

Figure 6.11: My offers ………………………………………………………………...51

Figure 6.12: Shipping address ………………………………………………………….52

Figure 6.13: Add to contacts …………………………………………………………..53

Figure 6.14: Rate book ………………………………………………………………...54

Figure 6.16: Add books ……………………………………………………………..56
Figure 6.17: Suggest books ………………………………………………………….57
Figure 6.17: Suggest books ………………………………………………………….58

LIST OF THE TABLE

Table Page

Table 2.1 Requirement Specification table ……………………………………………..4
Table 2.2 Initial Project Schedule Table ……………………………………………….7
Table 2.3 Idea Proposal ……………………………………………………………...7
Table 2.4 Requirement Gathering ……………………………………………………...8
Table 2.5 Physical System Design ……………………………………………………..8
Table 2.6 Logical System design ……………………………………………………….9
Table 2.7 Development Phase ………………………………………………………….9
Table 2.8 System Testing ……………………………………………………………..10
Table 5.1 Unit Test Case Module 1 …………………………………………………...42
Table 5.2 Unit Test Case Module 1 …………………………………………………...43
Table 5.3 Unit Test Case Module 1 …………………………………………………...43
Table 5.4 Unit Test Case Module 1 …………………………………………………...44
Table 5.5 Unit Test Case Module 1 …………………………………………………...44
Table 5.6 Unit Test Case Module 2 …………………………………………………...45
Table 5.7 Unit Test Case Module 2 …………………………………………………...45
Table 5.8 Unit Test Case Module 2 …………………………………………………...46
Table 5.9 Unit Test Case Module 2 …………………………………………………...46

CHAPTER 1

INTRODUCTION

1.1 Purpose

Our purpose of building the system is to make a big platform for all types of books absolutely for free for a book lover.

1.2.1 Background

As a book lover we often don’t get our expected books easily. From that we were thinking about such a platform where we can get easily our expected books.

1.2.2 Motivation

We are thinking on a book sharing initiative website. Where people can come and choose a book and exchange with his own book if he like.

We were thinking on a book sharing initiative website. Where people come and choose a book and exchange with his own book if he likes.

A. Unavailability of all types of books.
B. Expense of books.

1.2.3 Objective

1. Make easy and free access of all types of books
2. Lend own books and Borrow from others
3. Exchange books in the community.
4. Suggest books to your friends.
5. It can build our relation and other.
6. It will make each books lovers house a library accessible to all of our users.

1.2 Stakeholders

As we have one type user and user exchange their own books with each other and do other stuffs, so we have only one stakeholder.

CHARTER 2

REQUIREMENT SPECIFICATION AND ANALYSIS

2.1 System Analysis

Requirement analysis are several part and describe in details below

2.1.1 Requirement Elicitation

Requirement elicitation in our project is to find out what we need to full fill our project. This is very first step to build our project. Book Sharing is web based application to maintain day care system. Three stakeholder is in this application.

a) Admin

b) User

For those stakeholder there have many functionalities and non-functionalities. We will discuss in SRS subsection. Feature and quality requirement of admin, user in this section.
CRUD level of admin is high priority. Admin can do all what user want. . With this thought we made this project of Book sharing. Here a user can fide book, share book with other. He can exchange book with other user books. He can add his exchangeable book with picture. They can communicate with other by user profile address. It save money. It can build our relation and other. It will make each books lovers house a library accessible to all of our users.

Studying similar project: Most of the requirements are collected by studying similar projects. We observed their projects carefully and make our project advance. We observed their system not way of making. Some common feature is in our system such user profile, upload book, book details suggest book, favorite book and more on;

Name of website we studied:

- http://www.readingrockets.org/article/tips-sharing-bookswww.chapelhilldaycarecenter.com
- https://www.bookcrossing.com
- www.booksharing.com

Expert's opinion: we talk about our project with many web developer and try to understand what they suggest. Some of they have real life experienced in this sector. They suggest something key thing. And that was very important for this project. A security expert mark down some issues that we might face

Brainstorming: except similar projects and experts opinion everything we done in this projects is brainstorming. We write down everything in a scratch book and review all the term very carefully. Then we sort out best outcome and try to compare with our real life. We discuss mostly based on advanced feature ideas, programming capability.

2.1.2. Software Requirement Specification

Table 2.1: Requirement Specification table

Requirement Id	Requirement Title	Priority
FR_01	End user must be register in system. But only admin is pre-installed.	High
FR_02	In login process, system will fetch data from database and confirm identification for multiple user.	High
FR_03	Activities will create for particular user. Activities will be sortable and time notable.	Low
FR_04	In uploading time image will process in multiple size and format.	Medium
FR_05	System will generate an automated token. This will need to add user. One user one token	High
FR_06	System will inform about upcoming event. And create dynamic to desire destination.	Medium
FR_07	Developer would the ability access all data in testing purpose.	High

FR_08	Only register user have access to enter system. Admin will pre-registered. And able to change their avatar, name, password etc.	High
FR_9	Multiple authentication system is not injectable. Cross verification have to pass. System will not give access to view another page which is Guard by other type user.	High
FR_10	User friendly interface and easy system should be build. Non-technical people can use this System easily.	Medium
FR_11	System will remove same book.	Low
NFR_12	Good GUI, User Interface, User Experience	High
NFR_13	MySQL default capacity and extend version also Supported.	High

NFR_14	During the process, to update any type of data response time will not more than two second.	High
NFR_15	Password sorting hash, SQL query with pdo format, protect SQL injection, CSRF, Encryption.	High
NFR_16	Run smoothly in cross platform. Though apache web server is recommended but in IIS it should Be run smoothly.	High

2.2 Used Technology

This subsection is for what technology we use for build and run this project. Which Programming language, framework, libraries are used in project.

- Programming language: PHP 7.2
- Programming language Framework: LARAVEL 5.7
- Web Design: HTML5, CSS3, Bootstrap 4, Bootstrap material.
- Client side scripting language: JavaScript, Ajax
- Database : MySQL
- Web server: Apache.

2.3 Project Schedule

A. Initial Step

Table 2.2: Initial Project Schedule Table

Serial	Work Description	Start (date)	End (date)
1	Idea Finding	01/07/2018	08/08/2018
2	Feasibility study	08/08/2018	12/08/2018
3	Available Source Check	13/08/2018	18/09/2018
4	Mind Mapping	19/09/2018	26/09/2018
5	Similar site analysis	27/09/2018	30/09/2018

B. Idea Proposal

Table 2.3: Idea Proposal

Serial	Work Description	Start (date)	End (date)	Total day
1	Idea Finding With Supervisor	01/09/2018	04/09/2018	3
2	Feasibility study With Supervisor	05/09/2018	07/09/2018	2
3	Feature Discussion With Supervisor	08/09/2018	10/09/2018	2
4	Work Flow maintenance	11/09/2018	16/09/2018	5
5	SDLC Selection	17/09/2018	19/09/2018	2
Total Days				14

C. Requirement Gathering

Table 2.4: Requirement Gathering

Serial	Work Description	Start (date)	End (date)	Total day
1	System Flow sketch	20/08/2018	28/08/2018	8
2	Requirement gathering for proposed system	29/08/2018	05/08/2018	7
3	Requirement Collection	06/09/2018	08/09/2018	2
4	SRS	09/09/2018	11/09/2018	2
5	all requirement and Information	12/09/2018	16/09/2018	4
Total Days				23

D. Physical System Design

Table 2.5: Physical System Design

Serial	**Work Description**	**Start (date)**	**End (date)**	**Total day**
1	Designing Prototype	17/08/2018	22/08/2018	5
2	GUI (Graphical user Interface)	23/08/2018	05/08/2018	12
3	Process Design	06/08/2018	08/08/2018	2
Total Days				19

E. Logical System design

Table 2.6: Logical System design

Serial	**Work Description**	**Start (date)**	**End (date)**	**Total day**
1	Use case Diagram Design	09/08/2018	10/08/2018	1
2	Activity Diagram	11/08/2018	12/08/2018	1
3	Schema Diagram	13/08/2018	14/08/2018	1
4	Class Diagram	15/08/2018	16/08/2018	1
Total Days				4

F. Development Phase

Table 2.7: Development Phase

Serial	**Work Description**	**Start (date)**	**End (date)**	**Total day**
1	Build User Module	20/08/2018	27/08/2018	7
2	Build books Module	28/08/2018	03/08/2018	7
3	Build Admin Module	04/09/2018	12/09/2018	8
4	Multiple Auth	13/09/2018	16/09/2018	3
5	Database Integration	17/09/2018	19/09/2018	2
5	Live Streaming Implementation	20/09/2018	23/09/2018	3
7	Refactoring Code	30/08/2018	06/09/2018	7
8	Intend All Code	07/09/2018	08/09/2018	1

G. System Testing

Table 2.8: System Testing

Serial	**Work Description**	**Start (date)**	**End (date)**	**Total day**
1	Separate Module Test	10/09/2018	12/09/2018	2
2	Boundary Value Testing	13/09/2018	16/09/2018	3
3	Functionality Test	17/09/2018	19/09/2018	2
Total Days				07
Total Working Days 137				

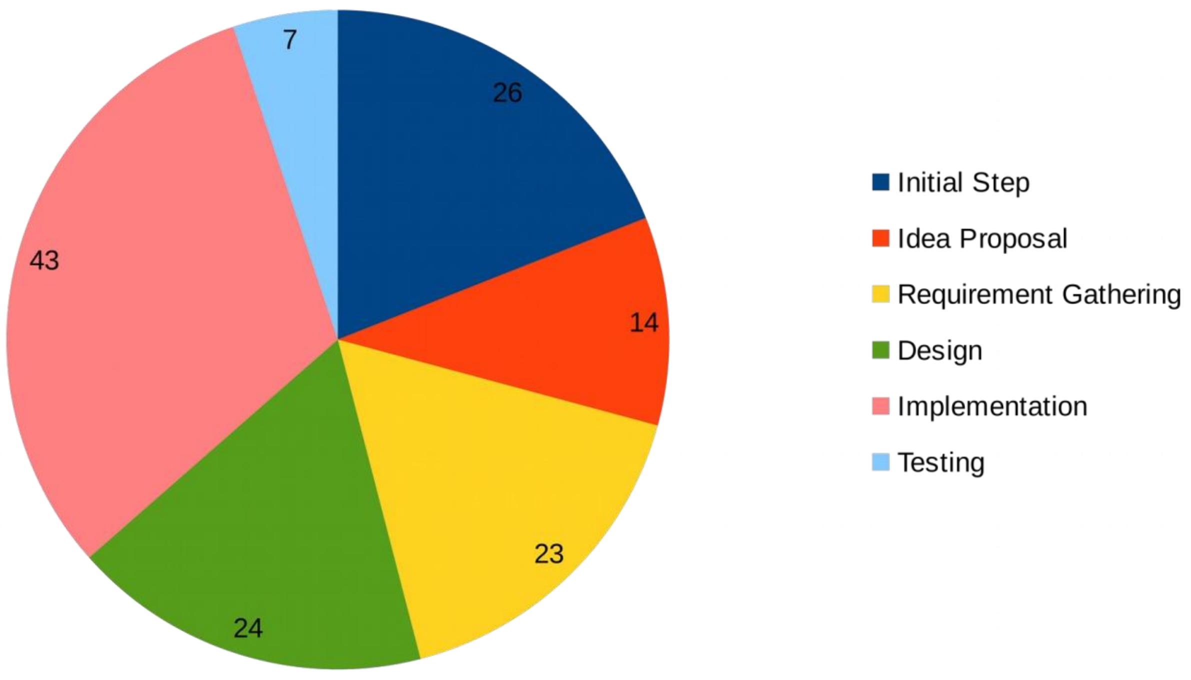


Figure 2.1: Schedule chart

CHAPTER 3

SYSTEM DESIGN

3.1 Use Case Diagram

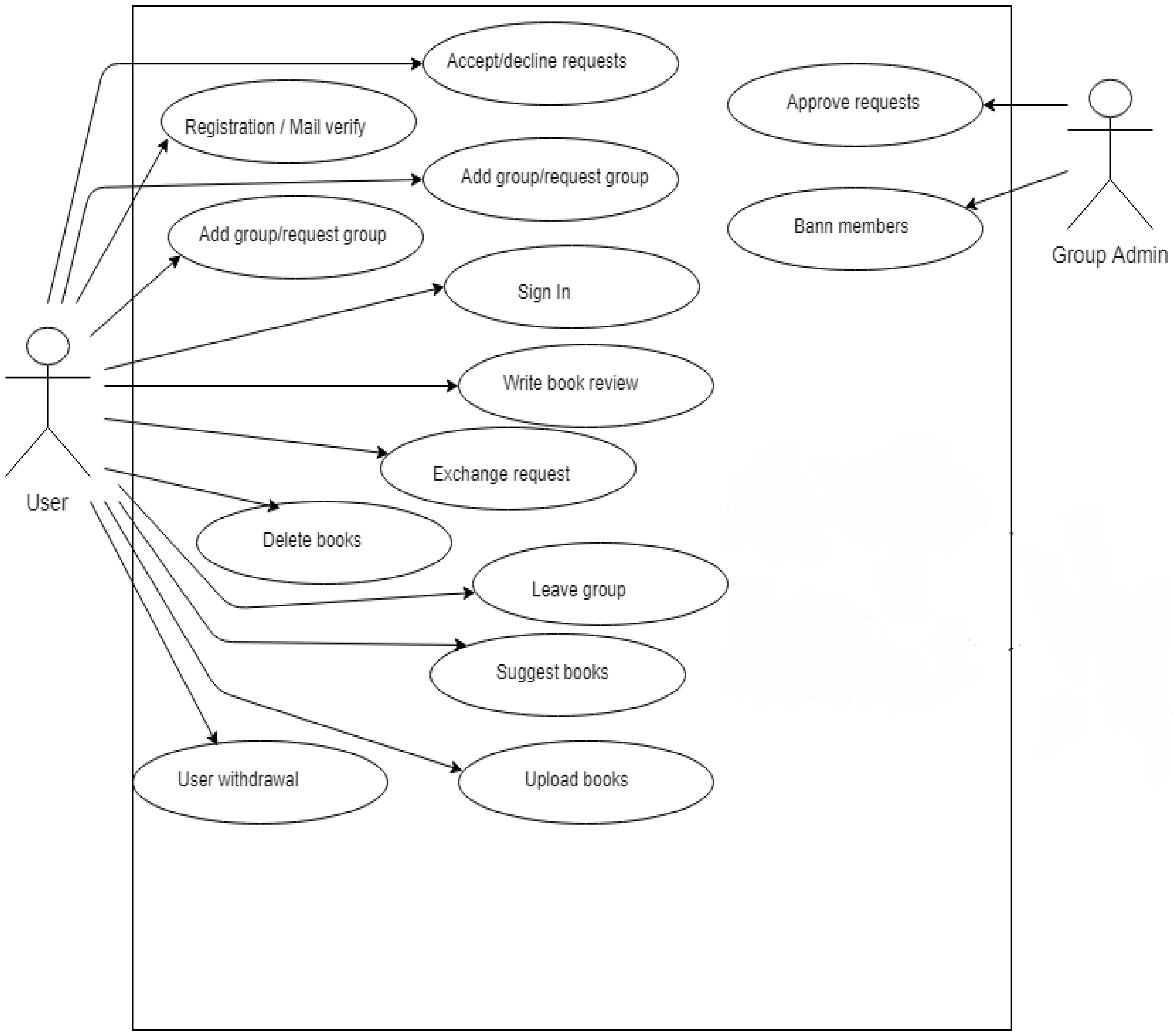


Figure 3.1: Use case diagram

Here user, admin are the actors of this system. The line, indicates the relationship of use cases like registration, login, comment, post, view book with the actors.

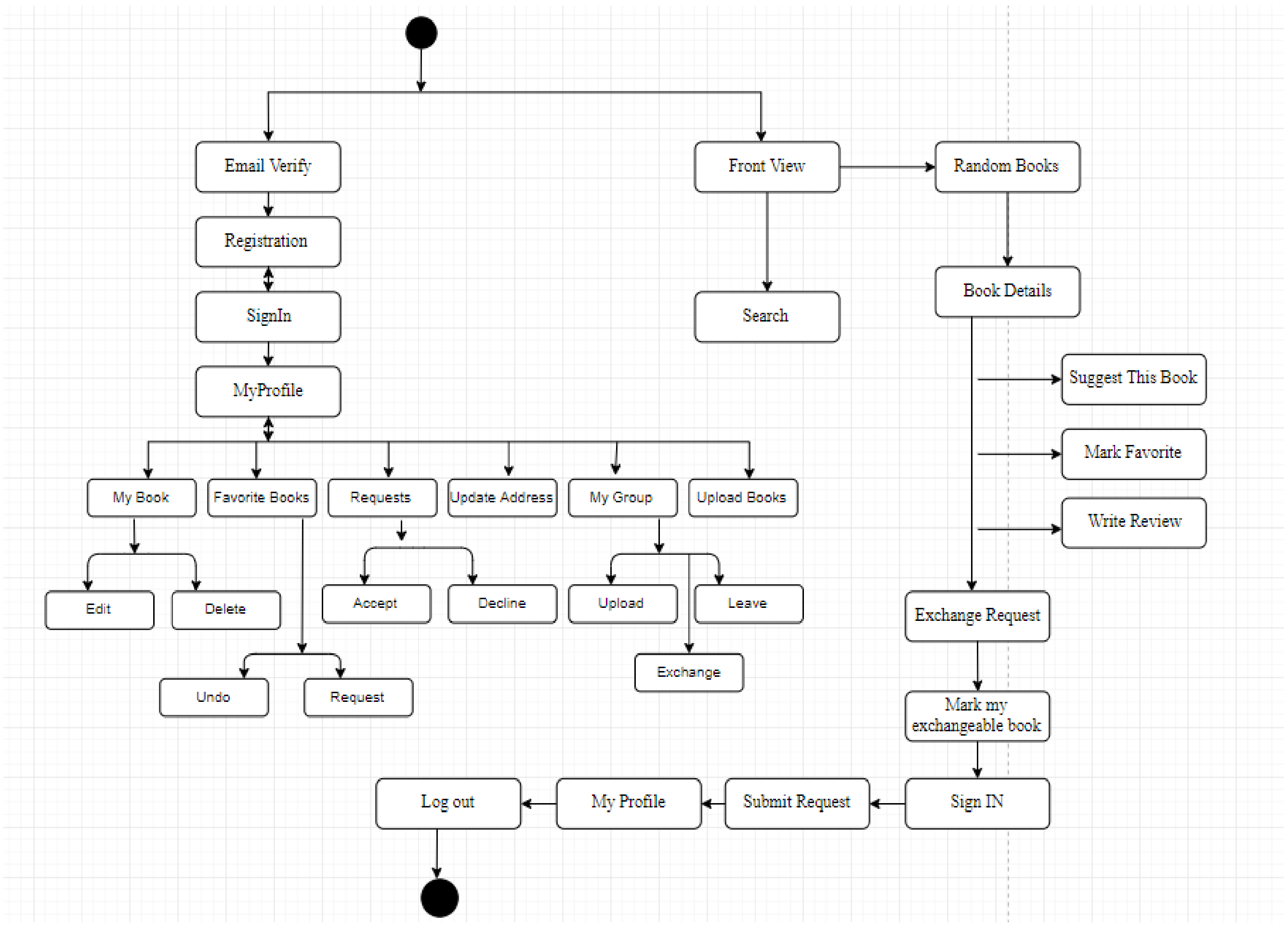


Figure 3.2: Activity diagram

Fig: 3.2 activity diagram are the sequence of the system. From the activity diagram we can understand the sequential activities of the system.

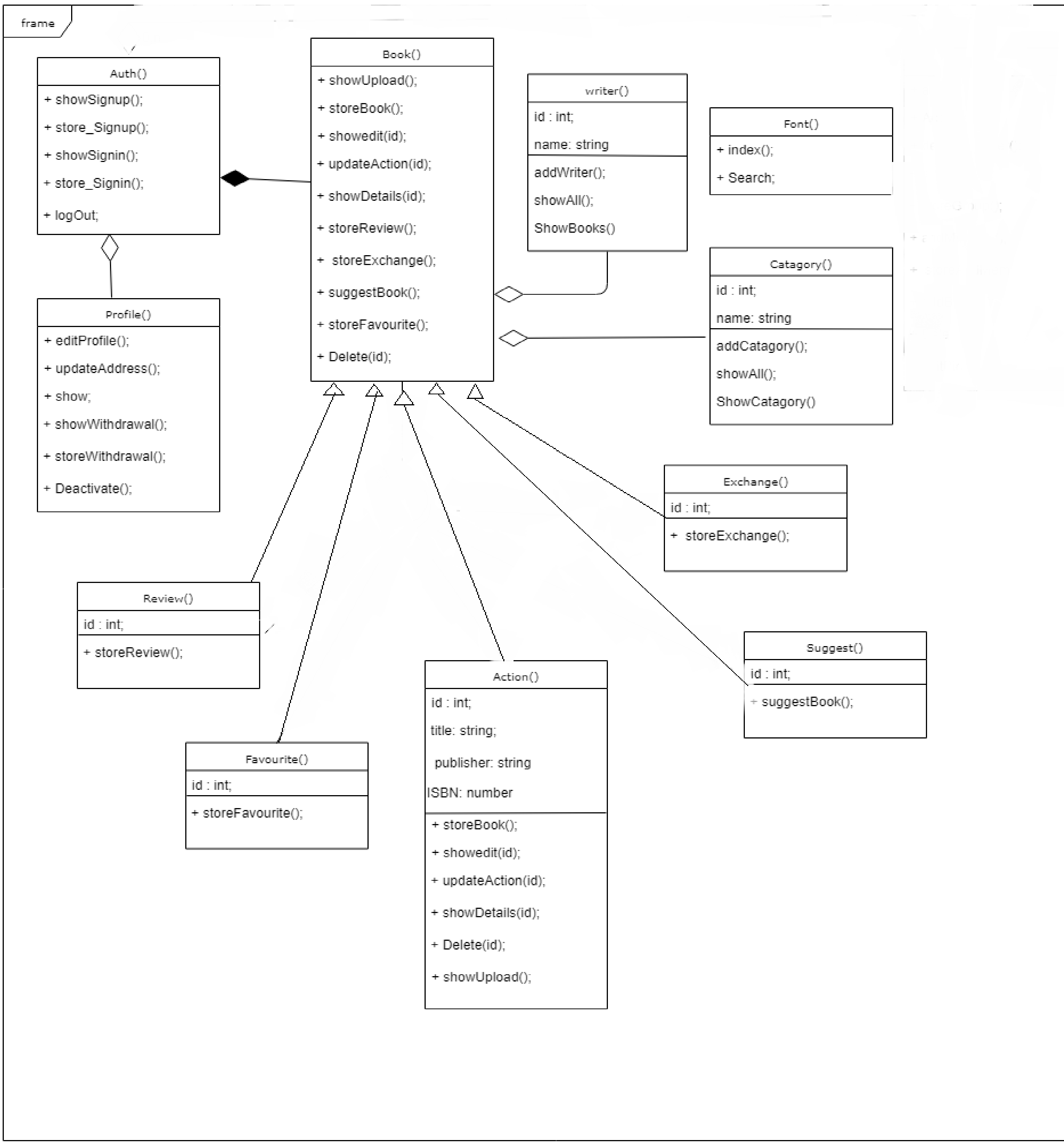
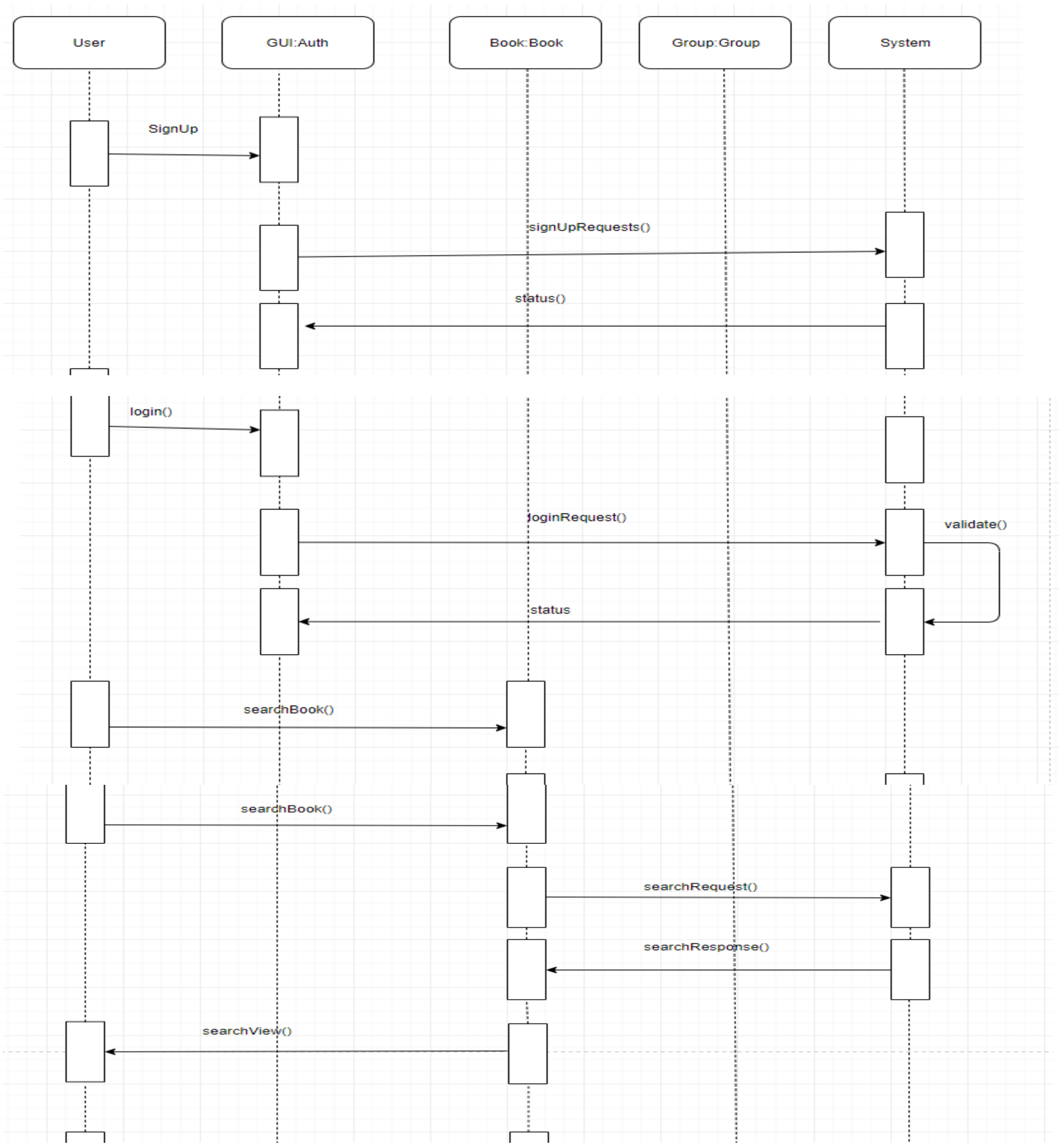
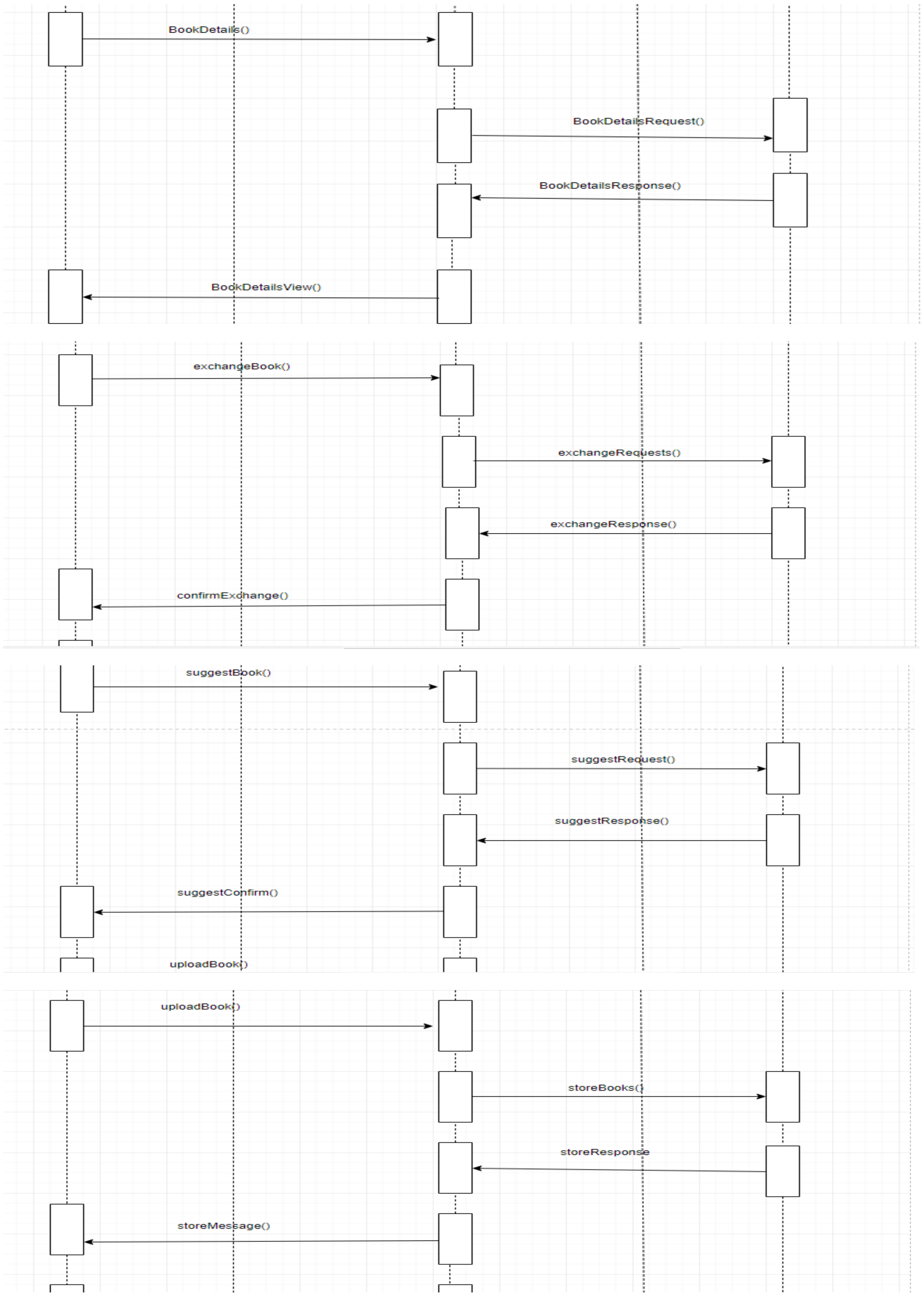


Figure 3.3: Class diagram

Here class diagram contain all the classes of this system. Profile class inherit Auth class. Request, favorite, Action, exchange, suggest class implement book class. Writer, category class inherit book class. All classes extends Controller class.


User
GUI:Auth
Book:Book
Group:Group
System
SignUp
signUpRequests()
status()
login()
loginRequest()
validate()
status
searchBook()
searchBook()
searchRequest()
searchResponse()
searchView()


BookDetails()
BookDetailsRequest()
BookDetailsResponse()
BookDetailsView()
exchangeBook()
exchangeRequests()
exchangeResponse()
confirmExchange()
suggestBook()
suggestRequest()
suggestResponse()
suggestConfirm()
uploadBook()
uploadBook()
storeBooks()
storeResponse
storeMessage()

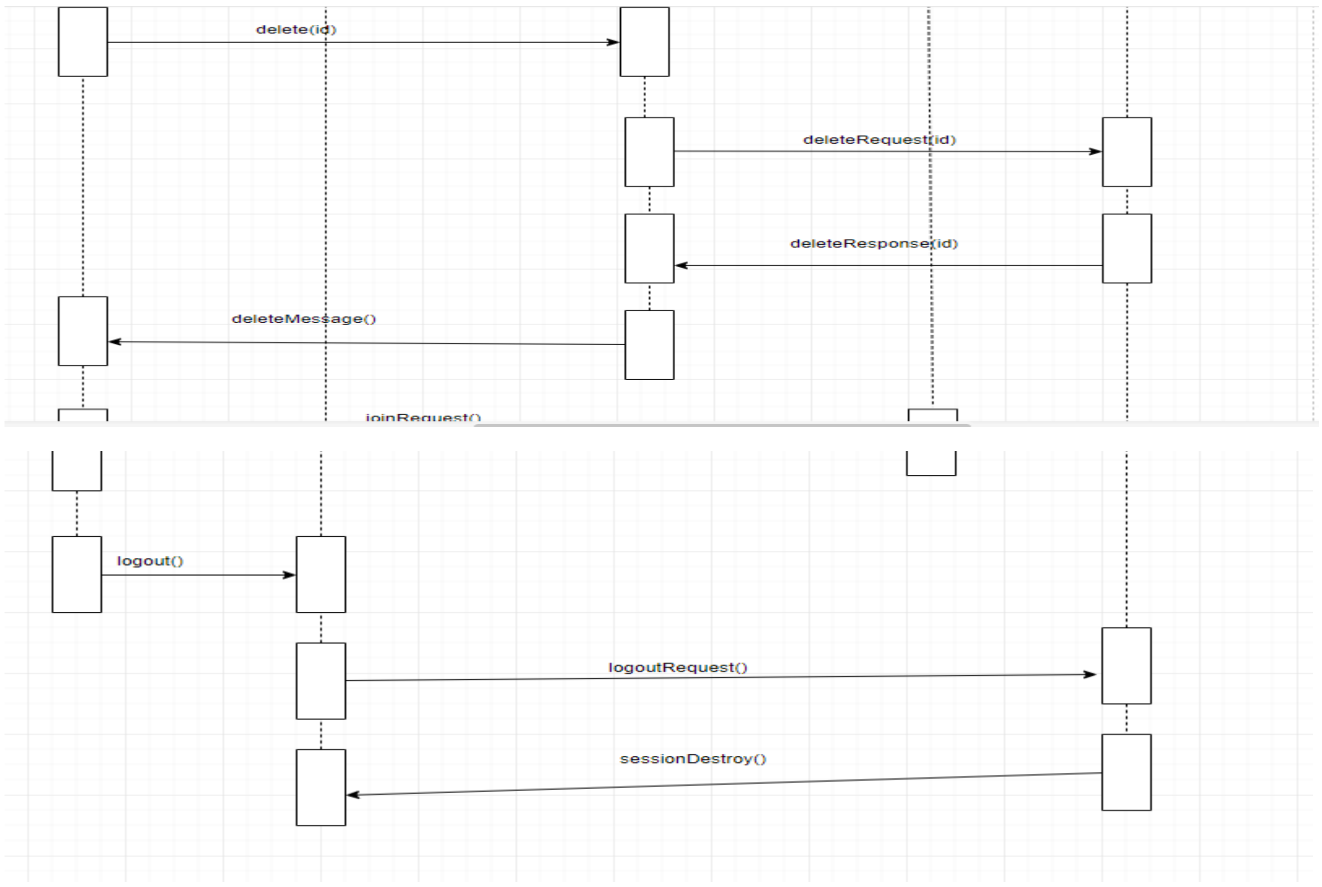


Figure 3.5: Sequence diagram

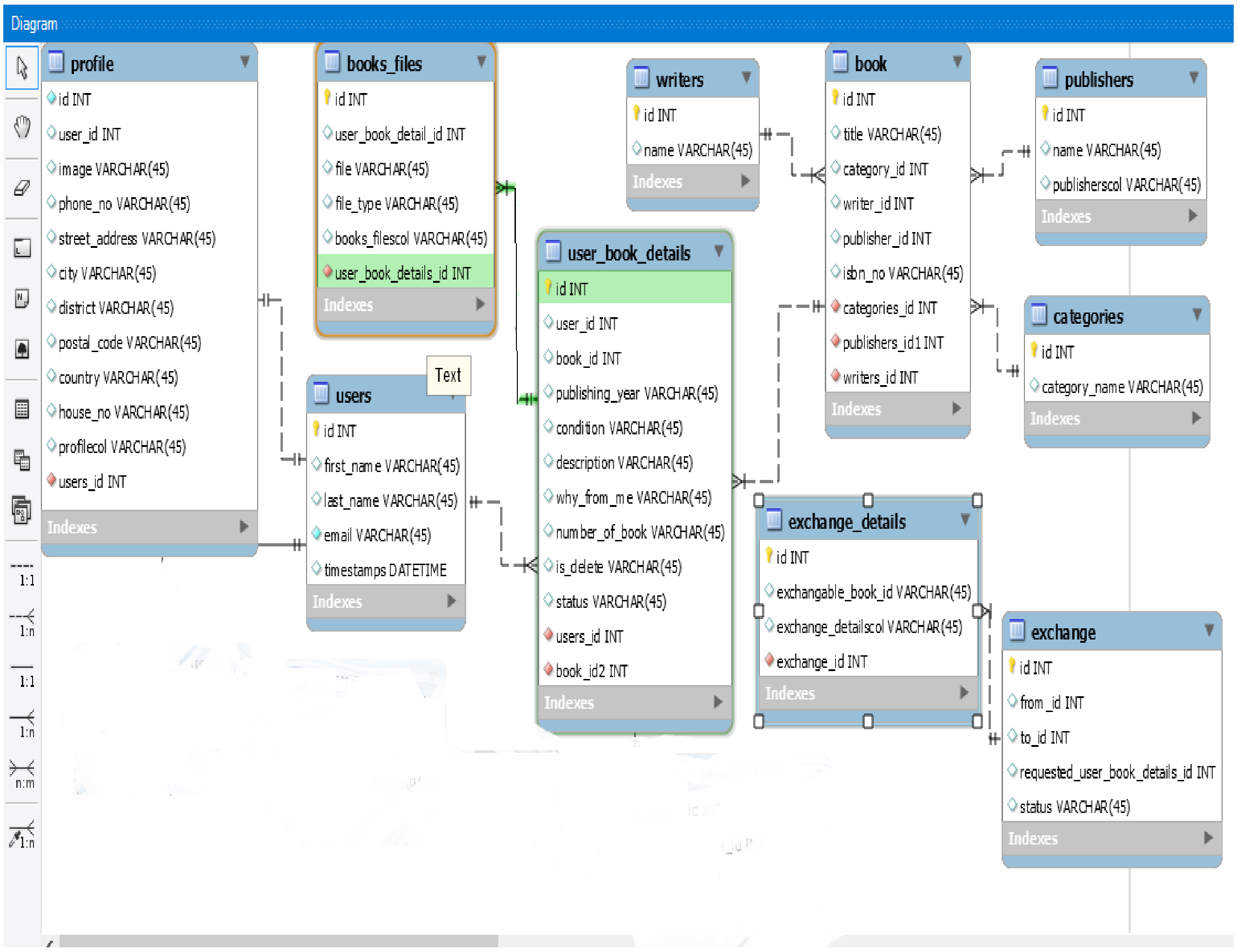


Figure 3.4: Schema diagram

CHAPTAR 4

CODES

4.1 Front

We've managed our all codes for front functionalities in two controllers belongs to Front folder. For authentication there is AuthController and for other front functionalities there is HomeController.

4.1.1 Authentication

In the AuthController we've managed all our functionalities for authentication. Here the code of the controller with the namespaces of traits it has used.

```
?php

namespace App\Http\Controllers\Front;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Auth;
use App\User;
use App\Model\Profile;


class AuthController extends Controller
{
  //SIGN-UP

  public function signUp(){
    return view('front.signUp');
  }

  public function store_signUp(Request $request){

    // return $request;
    $user = New User();

    $user->fname = $request->fname;
    $user->lname = $request->lname;
    $user->email = $request->email;
    $user->password = bcrypt($request->password);

    $user->save();

    $profile = new Profile();
    $profile->user_id = $user->id;
    $profile->save();


    return redirect()->route('front-signIn');
}
//SIGN-UP-ENDS

//SIGN-IN

  public function signIn(Request $request){
    if (isset($request->id)) {
      $data['bookId'] = $request->id;
      return view('front.signIn', $data);
    }
    return view('front.signIn');

  }

  public function store_signIn(Request $request){
    if (!Auth::attempt($request->only(['email', 'password']),
$request->has('remember_token'))) {
      return redirect()->back()->with('error', 'Credentials do
not match');
    }

    if (isset($request->bookId)) {
      return redirect()->route('front-book-details',
['id'=>$request->bookId]);
    }
    return redirect()->route('user-my-page');
  }

//SIGN-IN-ENDS

//SIGN-OUT

  public function logout(){
    Auth::logout();
    return redirect('/');
  }

//SIGN-OUT-ENDS


}
?>
```

4.1.2 Home

In the HomeController we've managed the front functionalities of our system for guest users. Here codes for viewing and using the features before sign in are written.

```
<?php
namespace App\Http\Controllers\Front;


use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Model\UserBooksDetail;
use App\Model\Book;
use App\Model\Category;
use App\Model\Writer;
use App\Model\BookImage;
use App\Model\Contact;
use Auth;
use Image;


class HomeController extends Controller
{
    public function index(Request $request){
    //   $data['allBooks'] = UserBooksDetail::paginate(6);
    $allBooks = UserBooksDetail::Join('books',
'userbooksdetails.book_id', '=', 'books.id')
    ->join('categories', 'categories.id', '=',
'books.category_id')
    ->join('writers', 'writers.id', '=', 'books.writer_id')
    ->select('userbooksdetails.*', 'books.title',
'categories.category_name', 'writers.writers_name' )
    ->where('status', '=', 1);


    if(isset($request->cat) || isset($request->writer)){
        if($request->cat){
            $data['allBooks'] = $allBooks-
>where('categories.category_name', '=', $request->cat)
            ->paginate(6);
        }
        if($request->writer){
            $data['allBooks'] = $allBooks-
>where('writers.writers_name', '=', $request->writer)
            ->paginate(6);
        }
    }else{
        $data['allBooks'] =   $allBooks->paginate(6);


    }


      return view('front.home', $data);
    }


    public function bookDetails(Request $request){
      $data['details'] = UserBooksDetail::Join('books',
'userbooksdetails.book_id', '=', 'books.id')
      ->join('categories', 'categories.id', '=',
'books.category_id')
      ->join('writers', 'writers.id', '=', 'books.writer_id')
      ->select('userbooksdetails.*', 'books.title',
'categories.category_name', 'writers.writers_name' )
      ->where('userbooksdetails.id', '=', $request->id)
      ->first();
      $images = BookImage::where('user_books_detail_id',
$request->id)->get();


      foreach($images as $key=>$image){
          if($key == 0){
              $data['image1'] = '';
              $data['image1']  = $image->image;
          }elseif($key == 1){
              $data['image2'] = '';
              $data['image2'] = $image->image;
          }elseif($key == 2){
              $data['image3'] = '';
              $data['image3'] = $image->image;
          }else{

          }

        }
        // return ($data['details']);
        if (Auth::check()) {
            $data['my_books'] = UserBooksDetail::Join('books', 
'userbooksdetails.book_id', '=', 'books.id')
                                ->join('categories', 
'categories.id', '=', 'books.category_id')
                                ->join('writers', 
'writers.id', '=', 'books.writer_id')
                                ->select('userbooksdetails.*', 
'books.title', 'categories.category_name', 
'writers.writers_name' )
                                -
>where('userbooksdetails.user_id', '=', Auth::user()->id)
                                ->get();



            $data['contacts'] = Contact::Join('users', 
'users.id', '=', 'contacts.contact_id')
                                ->Join('profile', 
'profile.user_id', '=', 'users.id')
```

```
                                ->select('profile.user_id',
'profile.image', 'users.fname', 'users.lname', 'users.email')
                                ->where('contacts.user_id',
'=', Auth::user()->id)
                                ->get();
        }
        if (isset($request->exchange_id)) {
            $data['exchangeId'] = $request->exchange_id;
        }


        $getRating = UserBooksDetail::Join('ratings',
'ratings.user_books_detail_id', '=', 'userbooksdetails.id')
                    ->where('userbooksdetails.id', '=',
$request->id)->get();


        if (!$getRating->isEmpty()) {
            $rateCount  = $getRating->count();
            $totalRate = 0;
             foreach ($getRating as $rate) {
                 $totalRate += $rate->ratingValue;
             }

             $data['avg'] = ceil($totalRate / $rateCount);
        }
        //  dd($avg);
        return view('front.bookDetails', $data);


    }



}
```

4.2 User

We’ve managed here our all codes for the feature functionalities an authenticated user can access and use such as manage books for exchange, suggest, rate or make favorite, manage profile, manage contacts through each separate controllers for correspondent stuffs.

4.2.1 Manage Books

In the BookController we’ve managed all our functionalities for manage user’s books. Here the code of the controller with the namespaces of traits it has used.

```
<?php

namespace App\Http\Controllers\User;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Model\Category;
use App\Model\Writer;
use App\Model\UserBooksDetail;
use App\Model\Book;
use App\Model\BookImage;
use App\Model\Rating;

use Image;
use Auth;



class BookController extends Controller
{
  //add-new-book
    public function addNew(){
      $data['category'] = Category::all();
      $bookInfo = [
        'title' => '',
        'id'    => '',
        'category_name' => '',
        'writers_name'  => ''
      ];
      $data['bookInfo'] = (object)$bookInfo;
      return view('users.newBook', $data);
    }
    public function getBooks(Request $request){
      $query = $request->book;
      $getBook = Book::where('title', 'LIKE', '%' .$query.
'%')->get();
      return $getBook;
    }
    public function getBooksInfo(Request $request){
      $data['bookInfo'] = Book::Join('categories',
'categories.id', '=', 'books.category_id' )
                   ->Join('writers', 'writers.id', '=',
'books.writer_id')
                   ->select('categories.category_name',
'writers.writers_name', 'books.title', 'books.id')
                   ->first();
      return view('users.newBook', $data);
    }
    public function storeBook(Request $request){
      $userBook = new UserBooksDetail();
      $userBook->user_id = Auth::user()->id;
      $userBook->book_id = $request->title;
      $userBook->publisher = $request->publisher;
      $userBook->publishing_year = $request->publishing_year;
      $userBook->isbn_no = $request->isbn;
      $userBook->description = $request->description;
      $userBook->condition = $request->condition_to_exchange;
      $userBook->status = 1;
      $userBook->is_delete = 0;


      $userBook->summery = $request->summery;
      $userBook->why_from_me = $request->why_from_me;
      $userBook->save();


      foreach ($request->image as $image) {
        if ($request->hasFile('image')) {

          $filename = time() . "." . $image-
>getClientOriginalExtension();
          $location = public_path('images/' . $filename);
          $img = Image::make($image);
          $img->save($location );

          $file = new BookImage;
          // $file->user_email = Auth::user()->email;
          $file->user_books_detail_id = $userBook->id;
          $file->image = $filename;
          $file->save();

        }
      }


      return redirect()->back()->with('success', 'A new book
added');
    }

    //add-book-ends
    //update-books

    public function showEditPage(Request $request, $title,
$id){
      $data['details'] = UserBooksDetail::Join('books',
'userbooksdetails.book_id', '=', 'books.id')
      ->join('categories', 'categories.id', '=',
'books.category_id')
      ->join('writers', 'writers.id', '=', 'books.writer_id')
      ->select('userbooksdetails.*', 'books.title',
'categories.category_name', 'writers.writers_name' )
      ->where('userbooksdetails.id', '=', $request->id)
      ->first();
      return view('users.editBook', $data);
    }

    public function storeEditBook(Request $request){
      $userBook = UserBooksDetail::where('id', $request->id)-
>first();
      $userBook->user_id = Auth::user()->id;
      $userBook->book_id = $request->title;
      $userBook->publisher = $request->publisher;
      $userBook->publishing_year = $request->publishing_year;
      $userBook->isbn_no = $request->isbn;
      $userBook->description = $request->description;
      $userBook->condition = $request->condition_to_exchange;
      $userBook->status = 1;
      $userBook->is_delete = 0;


      $userBook->summery = $request->summery;
      $userBook->why_from_me = $request->why_from_me;
      $userBook->save();


      if($request->image){
        foreach ($request->image as $image) {
          if ($request->hasFile('image')) {

            $filename = time() . "." . $image-
>getClientOriginalExtension();
            $location = public_path('images/' . $filename);
            $img = Image::make($image);
            $img->save($location );

            $file = new BookImage;
            // $file->user_email = Auth::user()->email;
            $file->user_books_detail_id = $userBook->id;
            $file->image = $filename;
            $file->save();
          }
        }
      }


      return redirect()->back()->with('success', 'Book
updated');
    }

    //update-books-ends

    //delte-books

    public function delete(Request $request){
      $getBook = UserBooksDetail::where('id', $request->book)-
>first();
      $getBook->status = 0;
      $getBook->save();
      // $getImage = BookImage::where('user_books_detail_id',
$request->book)->first();

      return redirect()->back();

    }

    //rate-book

    public function storeRating(Request $request){
      $checkRating = Rating::where('user_id', Auth::user()-
>id)->where('user_books_detail_id', $request->bookId)-
>first();

      if (empty($checkRating)) {

        $rating = new Rating();
        $rating->user_id = Auth::user()->id;
        $rating->user_books_detail_id = $request->bookId;
        $rating->ratingValue = $request->rateValue;

        $rating->save();
      }else{

        $checkRating->ratingValue = $request->rateValue;
        $checkRating->save();
      }
      return redirect()->route('exchange-log');

    }
}
```

4.2.2 Contacts

In the ContactController we've managed all our functionalities for manage user's books. Here codes for viewing and using the features before sign in are written.

```
<?php

namespace App\Http\Controllers\User;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Auth;
use App\Model\Contact;
use App\User;


class ContactController extends Controller
{
    //accept-contact-request
    public function accept(Request $request){
        $accept =   Contact::where('contact_id', Auth::user()-
>id)
                    ->where('user_id', $request->id)->first();
        $accept->status = 1;
        $accept->save();
        $setRecurring = new Contact();
        $setRecurring->user_id = Auth::user()->id;
        $setRecurring->contact_id = $request->id;
        $setRecurring->status = 1;

        $setRecurring->save();

        return redirect()->route('exchange-log');
    }

    //send-contact-request
    public function sendReq(Request $request){

        $sendReq = new Contact();
        $sendReq->user_id = Auth::user()->id;
        $sendReq->contact_id = $request->id;
        $sendReq->status = 0;

        $sendReq->save();
        return redirect()->route('book-on-exchange');
    }
}
```

4.2.3 MANAGE EXCHANGE

In the ExchangeController we've managed all our functionalities for exchanges user's books. Here codes for viewing and using the features before sign in are written.

```
<?php

namespace App\Http\Controllers\User;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Model\Category;
use App\Model\Writer;
use App\Model\UserBooksDetail;
use App\Model\Book;
use App\Model\BookImage;
use App\Model\Exchange;
use App\Model\ExchangeDetails;

use Image;
use Auth;

class ExchangeController extends Controller
{
    //STORE-EXCHANGE-REQUESTS
    public function storeExchange(Request $request)
    {
        //dd($request->all());
        $exchange = new Exchange();
        $exchange->from_id = Auth::user()->id;
        $exchange->to_id = $request->userId;
        $exchange->user_books_detail_id = $request->bookId;
        $exchange->status = 0;

        $exchange->save();

        foreach($request->exchangableBookId as $key =>
$exchangableBookId){
            $exchangeDetails = new ExchangeDetails();
            $exchangeDetails->exchange_id = $exchange->id;
            $exchangeDetails->user_books_detail_id =
$exchangableBookId;
            $exchangeDetails->qty = $request-
>get('qty_'.$exchangableBookId);
            $exchangeDetails->save();
        }
        return redirect()->back()-
>with('requested','requested');
    }

    public function storeDecission(Request $request){
        $eDtails = Exchange::where('id', $request-
>exchange_id)->first();
        $eDtails->status = 1;
        $eDtails->save();

        foreach($eDtails->exchangeDetail as $eDetail){
            $userBookOffer = UserBooksDetail::where('id',
$eDetail->user_books_detail_id)->first();
            $userBookOffer->status = 2;
            $userBookOffer->save();
        }
        $delete_other_exchange_offers_on_this_book =
Exchange::where('user_books_detail_id', $eDtails-
>user_books_detail_id)
                                                     -
>where('status', '!=', 1)
                                                     -
>delete();

        $userBook = UserBooksDetail::where('id', $eDtails-
>user_books_detail_id)->first();
        $userBook->status = 2;
        $userBook->save();
        return redirect()->back();
    }

    //GOT-MY-BOOK-BACK-AFTER-EXCHANGE
    public function gotBook(Request $request){
        $eDtails = Exchange::where('id', $request->book)-
>first();
        $eDtails->status = 2;
        $eDtails->save();

        foreach($eDtails->exchangeDetail as $eDetail){
            $userBookOffer = UserBooksDetail::where('id',
$eDetail->user_books_detail_id)->first();
            $userBookOffer->status = 1;
            $userBookOffer->save();
        }


        $userBook = UserBooksDetail::where('id', $eDtails-
>user_books_detail_id)->first();
        $userBook->status = 1;
        $userBook->save();
        return redirect()->back();
    }
}
```

4.2.4 Manage profile and exchanges from the profile

In the ProfileController we've managed all our functionalities for exchanges user's books from user profile and profile updates. Here codes for viewing and using the features before sign in are written.

```
<?php

namespace App\Http\Controllers\User;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Model\UserBooksDetail;
use App\Model\Book;
use App\Model\Category;
use App\Model\Writer;
use App\Model\BookImage;
use App\Model\Exchange;
use App\Model\ExchangeDetails;
use App\User;
use App\Model\Profile;
use App\Model\Rating;
use App\Model\Contact;




use Auth;
use Image;

class ProfileController extends Controller
{
  //SHOW-ALL-MY-BOOKS-IN-MY-PROFILE
    public function index(){
      $data['details'] = UserBooksDetail::Join('books',
'userbooksdetails.book_id', '=', 'books.id')
      ->join('categories', 'categories.id', '=',
'books.category_id')
      ->join('writers', 'writers.id', '=', 'books.writer_id')
      ->select('userbooksdetails.*', 'books.title',
'categories.category_name', 'writers.writers_name' )
      ->where('userbooksdetails.user_id', '=', Auth::user()-
>id)
      ->get();
      $data['profile'] = User::Join('profile', 'users.id',
'=', 'profile.user_id')
      ->where('users.id', '=',  Auth::user()->id)->first();
      $book_data = array();
      $i = 0;
      $getImage = '';
      foreach($data['details'] as $userBook){
        $book_data[$i][0] = $userBook;
        $book_data[$i][1] =
BookImage::where('user_books_detail_id', $userBook->id)-
>get();
        $book_data[$i][2] =
Exchange::where('user_books_detail_id', $userBook->id)-
>where('status', 0)->count();
        $getImage = $book_data[$i][1];
        foreach($getImage as $key => $image){
          if($key == 1){
            $coverImage = '';
            $coverImage = $image->image;
          }
        }
        $book_data[$i][1] = $coverImage;
        $i++;
      }

      $data['contacts'] = Contact::Join('users', 'users.id',
'=', 'contacts.contact_id')
                          ->Join('profile', 'profile.user_id',
'=', 'users.id')
                          ->select('profile.user_id',
'profile.image', 'users.fname', 'users.lname', 'users.email')
                          ->where('contacts.user_id', '=',
Auth::user()->id)
                          ->get();

      $data['book_data'] = $book_data;
      $data['tabActive'] = 'index';


      return view('users.myProfile', $data);
    }

    //SHOW-ALL-MY-SUGGESTIONS-IN-MY-PROFILE
    public function suggestions(){
      $data['details'] = UserBooksDetail::Join('books',
'userbooksdetails.book_id', '=', 'books.id')
      ->join('categories', 'categories.id', '=',
'books.category_id')
      ->join('writers', 'writers.id', '=', 'books.writer_id')
      ->join('suggestion', 'suggestion.user_books_detail_id',
'userbooksdetails.id')
      ->select('userbooksdetails.*', 'books.title',
'categories.category_name', 'writers.writers_name' )
```

```
      ->where('userbooksdetails.user_id', '=', Auth::user()-
>id)
      ->get();
      $data['profile'] = User::Join('profile', 'users.id',
'=', 'profile.user_id')
      ->where('users.id', '=',  Auth::user()->id)->first();

      $book_data = array();
      $i = 0;
      $getImage = '';
      foreach($data['details'] as $userBook){
        $book_data[$i][0] = $userBook;
        $book_data[$i][1] =
BookImage::where('user_books_detail_id', $userBook->id)-
>get();
        $getImage = $book_data[$i][1];
        foreach($getImage as $key => $image){
          if($key == 1){
            $coverImage = '';
            $coverImage = $image->image;
          }
        }
        $book_data[$i][1] = $coverImage;
        $i++;
      }


      $data['book_data'] = $book_data;
      $data['tabActive'] = 'suggestion';


      return view('users.mySuggestion', $data);
    }

    //SHOW-ALL-REQUESTS-FOR-MY-BOOKS
    public function get_book_requests(Request $request){
      $getUser = Exchange::where('to_id', Auth::user()->id)-
>where('status', '0')->get();
      $book_data = array();
      $i = 0;
      foreach($getUser as $book_id){
        $book_data[$i][0] = UserBooksDetail::Join('books',
'userbooksdetails.book_id', '=', 'books.id')
        ->join('categories', 'categories.id', '=',
'books.category_id')
        ->join('writers', 'writers.id', '=',
'books.writer_id')
        ->select('userbooksdetails.*', 'books.title',
'categories.category_name', 'writers.writers_name' )
        ->where('userbooksdetails.id', '=', $book_id-
>user_books_detail_id)
```

```
        ->first();
        $book_data[$i][1] =
BookImage::where('user_books_detail_id', $book_id-
>user_books_detail_id)->get();
        $getImage = $book_data[$i][1];
        foreach($getImage as $key => $image){
          if($key == 1){
            $coverImage = '';
            $coverImage = $image->image;
          }
        }
        $book_data[$i][1] = $coverImage;
        // dd($book_id->id);
        $book_data[$i][2] =
ExchangeDetails::Join('userbooksdetails',
'userbooksdetails.id', '=',
'exchangeDetails.user_books_detail_id')
                            ->Join('books', 'books.id', '=',
'userbooksdetails.book_id')
                            ->Join('exchange', 'exchange.id',
'=', 'exchangeDetails.exchange_id')
                            ->select('userbooksdetails.id',
'books.title', 'exchangeDetails.exchange_id',
'exchangeDetails.user_books_detail_id', 'exchangeDetails.qty',
'exchange.from_id', 'exchange.to_id')
                            ->where('exchange_id', $book_id-
>id)->get();

        $i++;
      }
      // dd($book_data);
      $data['book'] = $book_data;
      $data['profile'] = User::Join('profile', 'users.id',
'=', 'profile.user_id')
      ->where('users.id', '=',  Auth::user()->id)->first();

      $data['tabActive'] = 'myBookRequest';


      return view('users.my-books-requests', $data);
    }

    //MY-EXCHANGE-REQUESTS-ARE-ACCEPTED
    public function exchangeLog(Request $request){

      $getUser = Exchange::where('from_id', Auth::user()->id)-
>where('status', '1')->get();
      $book_data = array();
      $i = 0;
      foreach($getUser as $book_id){
        $book_data[$i][0] = UserBooksDetail::Join('books',
'userbooksdetails.book_id', '=', 'books.id')
        ->join('categories', 'categories.id', '=',
'books.category_id')
        ->join('writers', 'writers.id', '=',
'books.writer_id')
        ->select('userbooksdetails.*', 'books.title',
'categories.category_name', 'writers.writers_name' )
        ->where('userbooksdetails.id', '=', $book_id-
>user_books_detail_id)
        ->first();
        $book_data[$i][1] =
BookImage::where('user_books_detail_id', $book_id-
>user_books_detail_id)->get();
        $getImage = $book_data[$i][1];
        foreach($getImage as $key => $image){
          if($key == 1){
            $coverImage = '';
            $coverImage = $image->image;
          }
        }
        $book_data[$i][1] = $coverImage;
        $book_data[$i][2] =
ExchangeDetails::Join('userbooksdetails',
'userbooksdetails.id', '=',
'exchangeDetails.user_books_detail_id')
                            ->Join('books', 'books.id', '=',
'userbooksdetails.book_id')
                            ->select('userbooksdetails.id',
'books.title', 'exchangeDetails.exchange_id',
'exchangeDetails.user_books_detail_id', 'exchangeDetails.qty')
                            ->where('exchange_id', $book_id-
>id)->get();
        $book_data[$i][3] = User::Join('profile', 'users.id',
'=', 'profile.user_id')
        ->where('users.id', '=',  $book_id->to_id)->first();

        $getRating = Rating::where('user_id', Auth::user()-
>id)
                            ->where('user_books_detail_id',
$book_id->user_books_detail_id)
                            ->first();
        if (empty($getRating)) {
          $book_data[$i][4] = 0;
        }else{
          $book_data[$i][4] = $getRating;

        }

        $checkContact = Contact::where('contact_id',
Auth::user()->id)
```

```
        ->where('user_id', $book_id->to_id)
        ->where('status', 0 )->first();
          if(!empty($checkContact)){
          $book_data[$i][5] = 'accept';
          }else{
            $checkContactReq = Contact::where('user_id',
Auth::user()->id)
                    ->where('contact_id', $book_id->to_id)
                    ->where('status', 0 )->first();
            if (!empty($checkContactReq)) {
            $book_data[$i][5] = 'ReqSent';
            }else{
              $checkContactReq1 = Contact::where('user_id',
Auth::user()->id)
                    ->where('contact_id', $book_id->to_id)
                    ->where('status', 1 )->first();
              $checkContactReq2 = Contact::where('contact_id',
Auth::user()->id)
                    ->where('user_id', $book_id->to_id)
                    ->where('status', 1 )->first();
              if(!empty($checkContactReq1) ||
!empty($checkContactReq2)){
                $book_data[$i][5] = 'contact';
              }else{
                $book_data[$i][5] = 'sendReq';

              }
            }
        }
        $i++;
      }
      // dd($getRating);

      $data['book'] = $book_data;
      $data['profile'] = User::Join('profile', 'users.id',
'=', 'profile.user_id')
      ->where('users.id', '=',  Auth::user()->id)->first();

      $data['tabActive'] = 'myExchangeLog';

      return view('users.my-exchange-log', $data);


    }

    //MY-BOOKS-ON-EXCHANGE
    public function bookOnExchange(Request $request){
      $getUser = Exchange::where('to_id', Auth::user()->id)-
>where('status', '1')->get();
      $book_data = array();
      $i = 0;
      foreach($getUser as $book_id){
        $book_data[$i][0] = UserBooksDetail::Join('books',
'userbooksdetails.book_id', '=', 'books.id')
        ->join('categories', 'categories.id', '=',
'books.category_id')
        ->join('writers', 'writers.id', '=',
'books.writer_id')
        ->select('userbooksdetails.*', 'books.title',
'categories.category_name', 'writers.writers_name' )
        ->where('userbooksdetails.id', '=', $book_id-
>user_books_detail_id)
        ->first();

        $book_data[$i][1] =
BookImage::where('user_books_detail_id', $book_id-
>user_books_detail_id)->get();
        $getImage = $book_data[$i][1];
        foreach($getImage as $key => $image){
          if($key == 1){
            $coverImage = '';
            $coverImage = $image->image;
          }
        }
        if(isset($coverImage)){

          $book_data[$i][1] = $coverImage;
        }
        $book_data[$i][2] =
ExchangeDetails::Join('userbooksdetails',
'userbooksdetails.id', '=',
'exchangeDetails.user_books_detail_id')
                            ->Join('books', 'books.id', '=',
'userbooksdetails.book_id')
                            ->select('userbooksdetails.id',
'books.title', 'exchangeDetails.exchange_id',
'exchangeDetails.user_books_detail_id', 'exchangeDetails.qty')
                            ->where('exchange_id', $book_id-
>id)->get();
        $book_data[$i][3] = User::Join('profile', 'users.id',
'=', 'profile.user_id')
                            ->where('users.id', '=',
$book_id->from_id)->first();

        $book_data[$i][4] = $book_id->id;
        $checkContact = Contact::where('contact_id',
Auth::user()->id)
                        ->where('user_id', $book_id->from_id)
                        ->where('status', 0 )->first();
        if(!empty($checkContact)){
          $book_data[$i][5] = 'accept';
        }else{
          $checkContactReq = Contact::where('user_id',
Auth::user()->id)
                        ->where('contact_id', $book_id-
>from_id)
                        ->where('status', 0 )->first();
          if (!empty($checkContactReq)) {
            $book_data[$i][5] = 'ReqSent';
          }else{
            $checkContactReq1 = Contact::where('user_id',
Auth::user()->id)
                  ->where('contact_id', $book_id->from_id)
                  ->where('status', 1 )->first();
            $checkContactReq2 = Contact::where('contact_id',
Auth::user()->id)
                  ->where('user_id', $book_id->from_id)
                  ->where('status', 1 )->first();
            if(!empty($checkContactReq1) ||
!empty($checkContactReq2)){
              $book_data[$i][5] = 'contact';
            }else{
              $book_data[$i][5] = 'sendReq';

            }
          }
        }


        $i++;
      }

      $data['book'] = $book_data;
      $data['profile'] = User::Join('profile', 'users.id',
'=', 'profile.user_id')
      ->where('users.id', '=',  Auth::user()->id)->first();

      $data['tabActive'] = 'myBookOnExchange';


      return view('users.my-book-on-exchange', $data);
    }

    //UPDATE-MY-PROFILE-INFO
    public function storeProfile(Request $request){
      $profile = Profile::where('user_id', Auth::user()->id)-
>first();

      $profile->address = $request->address;
      $profile->city = $request->city;
      $profile->district = $request->district;
      $profile->postal_code = $request->postal_code;
      $profile->country = $request->country;
      $profile->phone_no = $request->phone_no;

      if ($request->hasFile('image')) {
        $image = $request->file('image');
        $filename = time() . "." . $image-
>getClientOriginalExtension();
        $location = public_path('images/' . $filename);
        $img = Image::make($image);
        $img->save($location );
        $profile->image = $filename;
      }

      $profile->save();
      return redirect()->back();
    }
}
```

4.2.5 Manage suggestions

In the SuggestController we've managed all our functionalities for suggestions of our user's books. Here codes for viewing and using the features before sign in are written.

```
<?php

namespace App\Http\Controllers\User;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use Auth;
use App\Model\Suggestion;

class SuggestController extends Controller
{
    //STORE-SUGGESTIONS
    public function storeSuggestion(Request $request){

        foreach ($request->suggest_id as $suggest_id) {
            $suggest = new Suggestion();
            $suggest->suggest_id = $suggest_id;
            $suggest->user_books_detail_id = $request-
>book_id;
            $suggest->suggested_by = Auth::user()->id;

            $suggest->save();
        }

        return redirect()->route('user-my-page');
    }
}
```

All the codes of the traits and models are given in the Appendix A.Migratins are given in Appendix B and Routes are given in Appendix c.

CHAPTER 5

SYSTEM TESTING

5.1 PHP Unit Test

Software testing is a way, to evaluate the functionality of a application with an intent to find whether the developed software assists with the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product. Manual Testing: Manual testing is the way of testing the software manually to find the error. Tester should have the perspective of an end users and to ensure all the features are working as mentioned in the requirement document. In this process, developer run the test cases and generate the reports manually without using any automation tools. Automation Testing: Automation testing is the way of testing the software using an automation tools to find the error. In this process, testers execute the test file and generate the test results automatically by using automation tools. Some of the famous automation testing software for functional testing are QTP/UFT and Selenium. In our case we use manually PHP UNIT TEST. We go through every function which need to be tested. Give assertion and return result. We also use a tool (test cafe) to final checkup proposed system. Result are included in this section.

5.1.1 Module 1: user

This module describe all test case of user

Features should be tested:

- Registration & Login
- Add book
- request
- rate
- favorite

Table 5.1: Unit Test Case Module 1

Test name: Registration & Login			
Test procedure	**PHP Unit Test**		
Test Id	**KCUT001**		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Form Validation Check	Escape slash, escape special character, File size not more than 5M	OK
2	User existing (registration)	Checking Existing user	OK
3	Secure Logged In	User Session creating	OK

Table 5.2: Unit Test Case Module 1

Test name: Upload book			
Test procedure	**PHP Unit Test**		
Test Id	**KCUT002**		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Give book title	Get the book available in the system with category, writer name and publication	OK
2	Give book title	The book available in the system with category,	OK

		writer name and publication not available. User upload the book on system.	
3	Upload image	Upload more or less than 3 image	OK

Table 5.3: Unit Test Case Module 1

Test name: Request for exchange			
Test procedure	**PHP Unit Test**		
Test Id	**KCUT003**		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Request for book	The book is already requested	NOK
2	Request for book	The book is not requested yet.	OK
3	Mark own books with quantity to offer	True	OK

Table 5.4: Unit Test Case Module 1

Test name: Accept exchange request	
Test procedure	**PHP Unit Test**
Test Id	**KCUT004**

Step number	Operator actions	Expected result and evaluation criteria	Result
1	See offers	True	OK
2	Accept offer	True	OK
3	Decline offer	True	OK

Table 5.5: Unit Test Case Module 1

Test name: Shipment			
Test procedure	**PHP Unit Test**		
Test Id	**KCUT005**		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	See shipping address	True	OK
2	Send book	True	OK

5.1.2 Module 2: Favorite book

This module describe all test case of favorite book.

Features should be tested:

- Make favorite
- Undo favorite

Table 5.6: Unit Test Case Module 2

Test name: Favorite book			
Test procedure	**PHP Unit Test**		
Test Id	**KCUT006**		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Mark favorite	True	OK
2	Undo favorite	True	OK

Table 5.7: Unit Test Case Module 2

Test name: Rate book			
Test procedure	**PHP Unit Test**		
Test Id	**KCUT007**		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Rate book	Rate for the first time	OK
2	Rate book	Rated before, just edit score.	OK

Table 5.8: Unit Test Case Module 2

Test name: Suggest book			
Test procedure	**PHP Unit Test**		
Test Id	**KCUT008**		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Suggest own book to contacts	Already suggested	NOK
2	Suggest own book to contacts	True	OK

Table 5.9: Unit Test Case Module 2

Test name: Update profile			
Test procedure	**PHP Unit Test**		
Test Id	**KCUT008**		
Step number	**Operator actions**	**Expected result and evaluation criteria**	**Result**
1	Update profile info	Already updated just edit	OK
2	Update profile info	True	OK

CHAPTAR 6

USER MANUAL

6.1 Guest user

User manual for user's before authenticated.

<u>Step 1</u>:

Visit the front page.

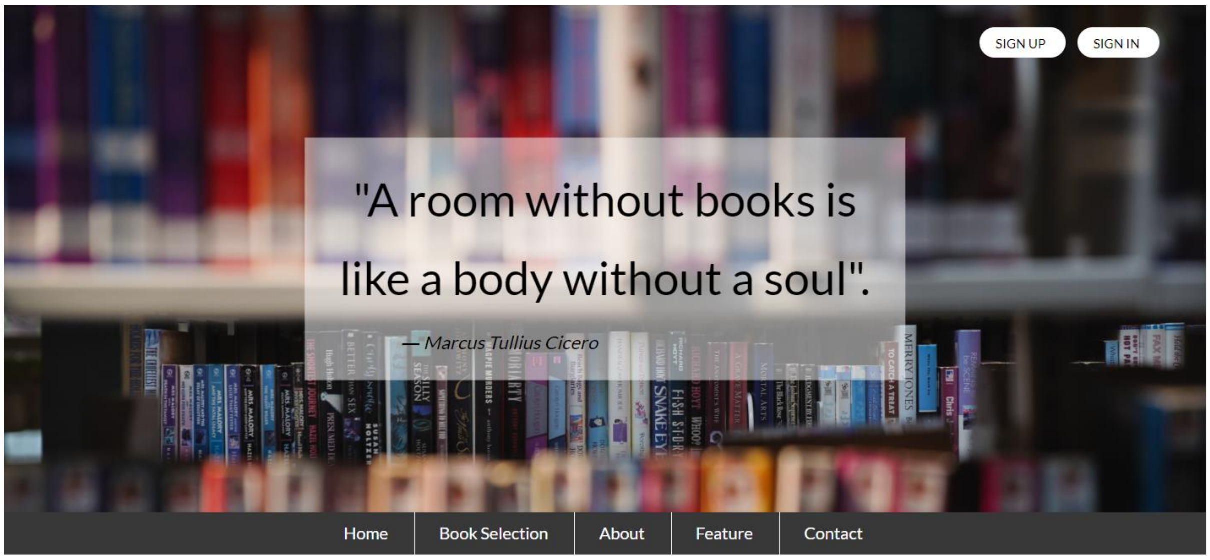


Figure 6.1: Front hero

See books cards. And click the indicated button to see details.

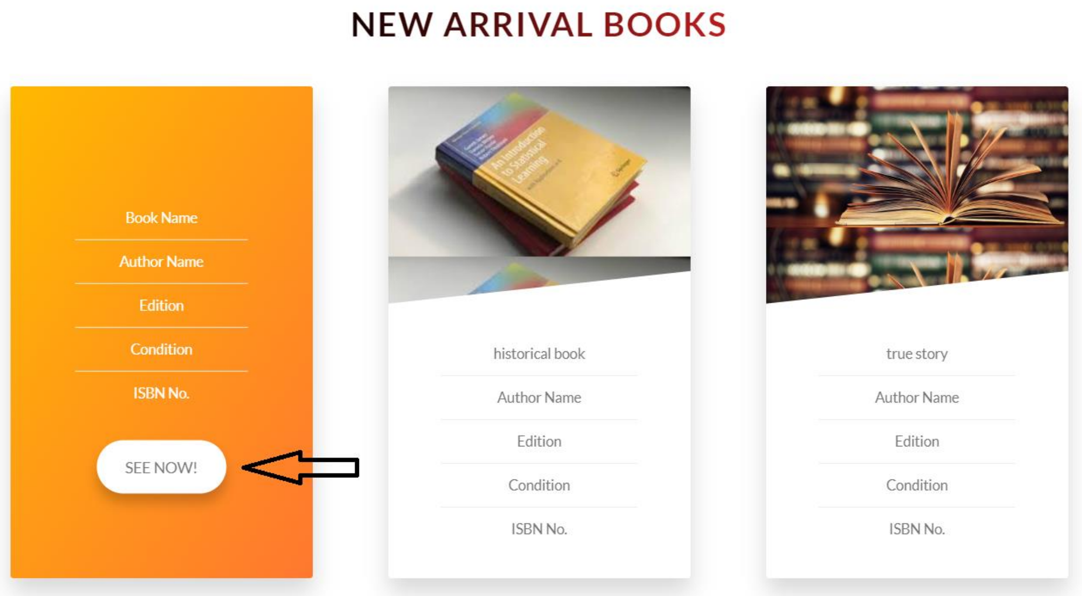
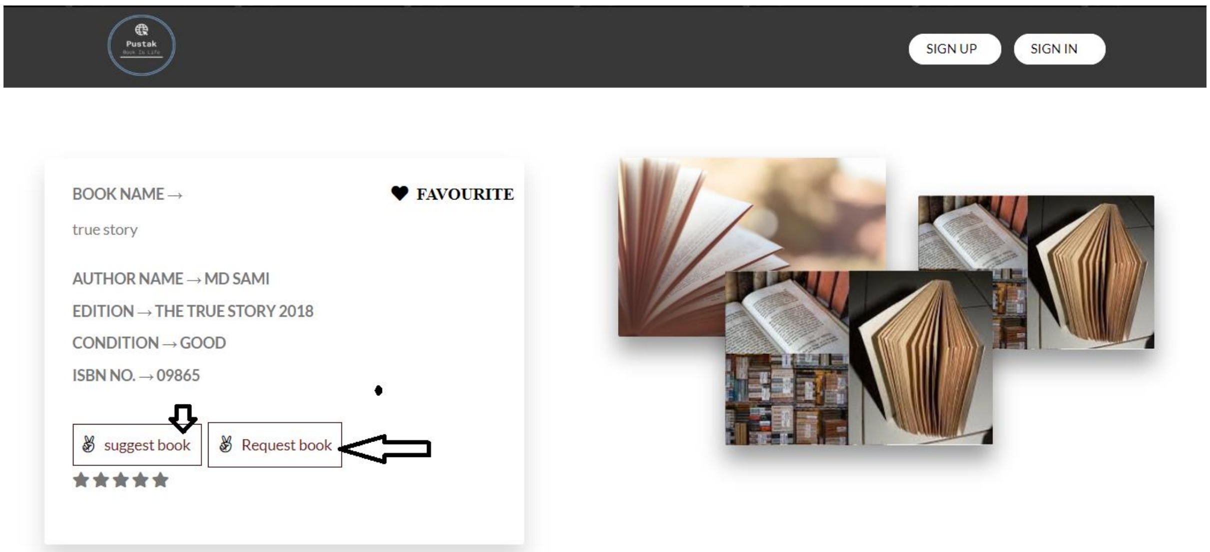


Figure 6.2: Book cards

Step2:

User can request for the book or suggest book to contacts.



BOOK NAME →

true story

AUTHOR NAME → MD SAMI

EDITION → THE TRUE STORY 2018

CONDITION → GOOD

ISBN NO. → 09865

Figure 6.3: Book Details

6.2 Authenticate User

Step3:

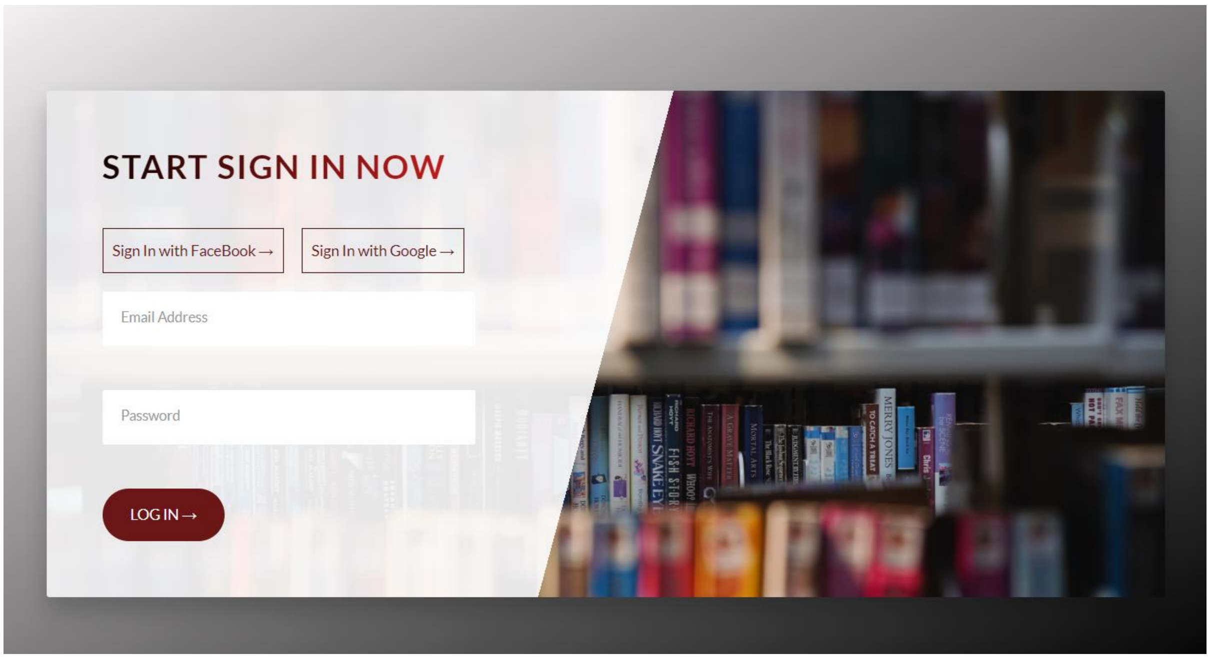


Figure 6.4: Sign In

User redirected to sign in page to requesting or suggesting.

If user is not registered first sign up and then sign in.

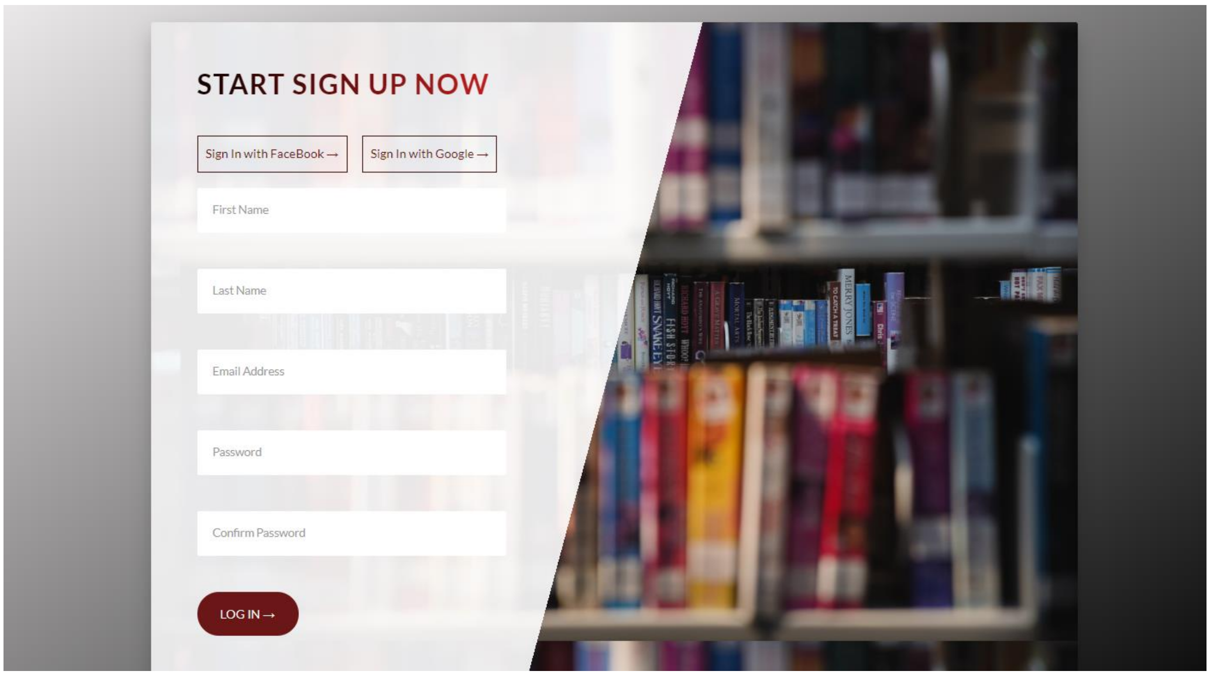
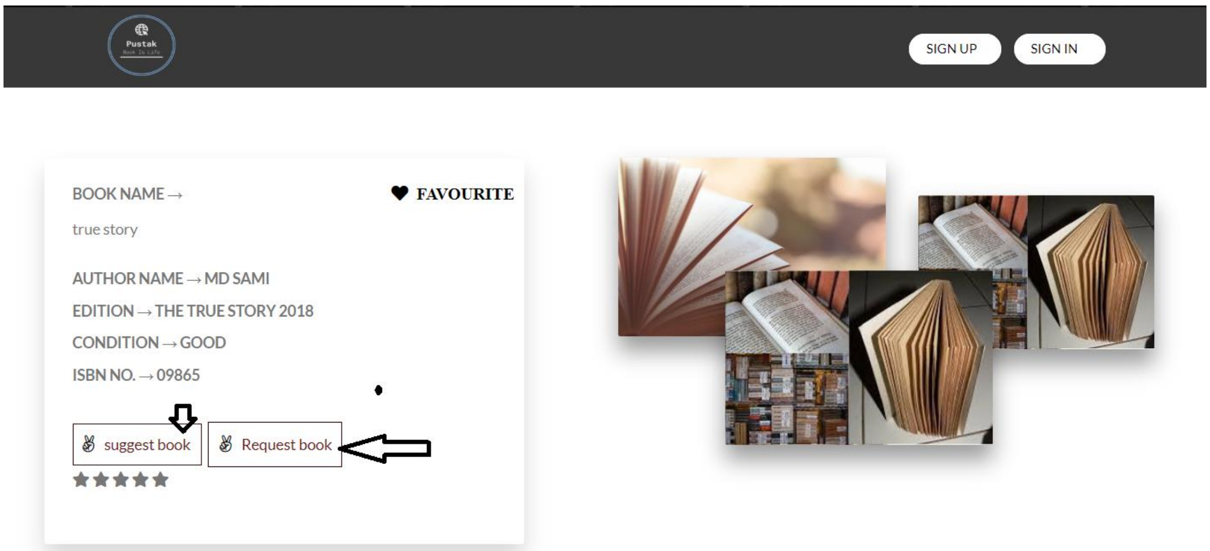


Figure 6.5: Sign Up

<u>Step4:</u>

Suggest or request the book.



BOOK NAME →

true story

AUTHOR NAME → MD SAMI

EDITION → THE TRUE STORY 2018

CONDITION → GOOD

ISBN NO. → 09865

Figure 6.6: Request Book

After clicking request book button there appears a pop up with user's own books to let him select his/her books to submit an exchange offer.

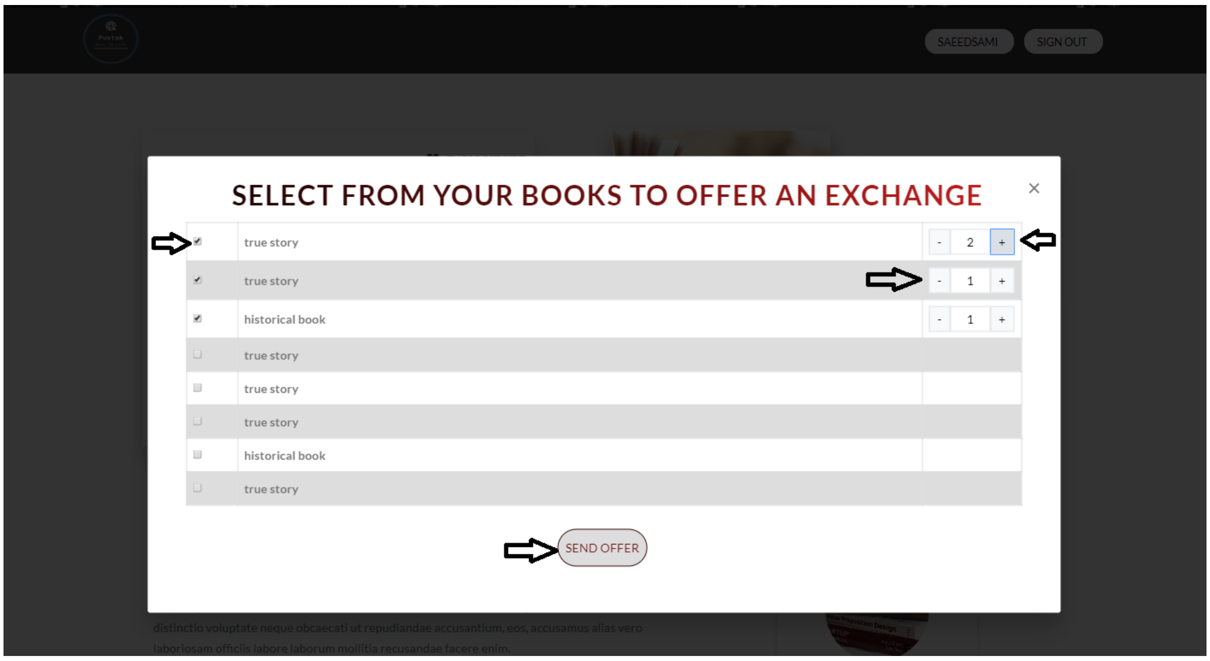
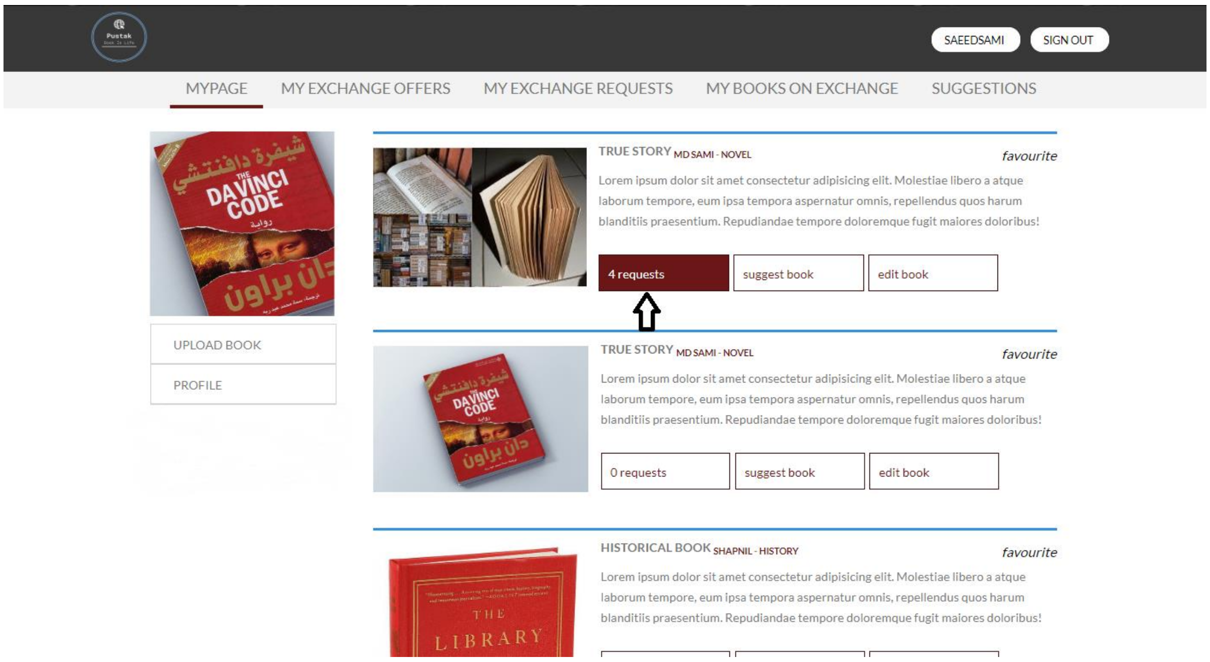


Figure 6.7: Select my books for offers.

<u>Step5:</u> My profile

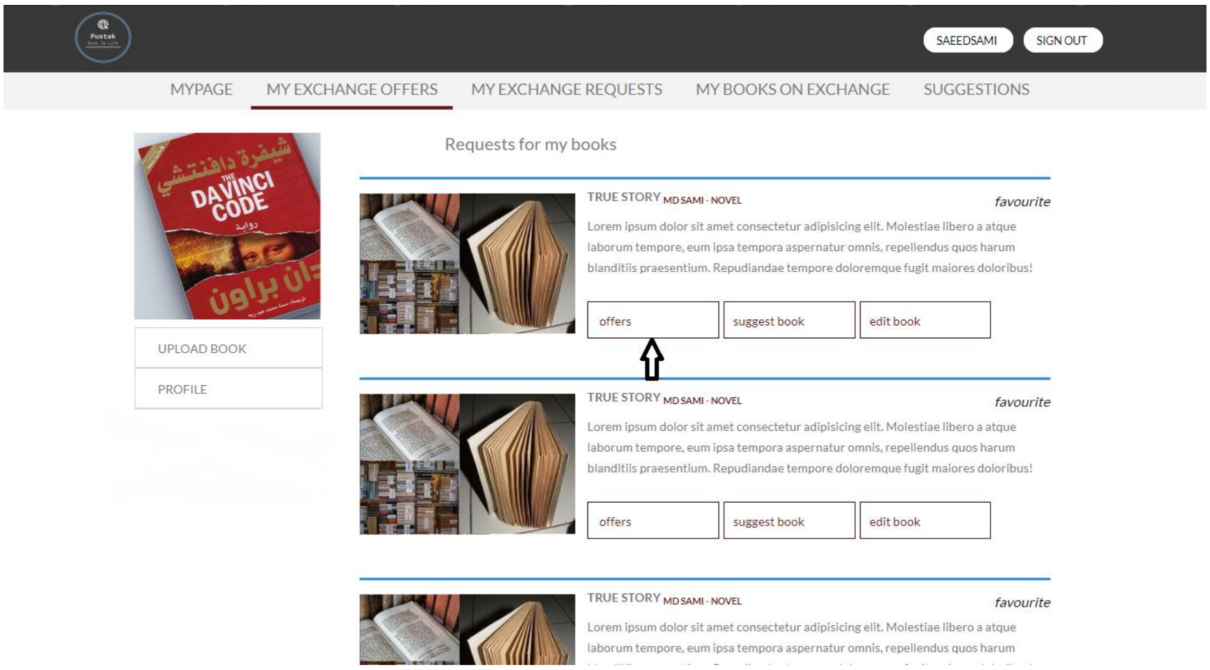
In the MY PAGE tab user see his own books list with exchange requests notification.



Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

Figure 6.8: exchange notification.

Clicking on the request notification red button it redirects to the my exchange offers tab with that clicked books requests.



Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

Here in the offers button shows a popup where offers for the books are shown.

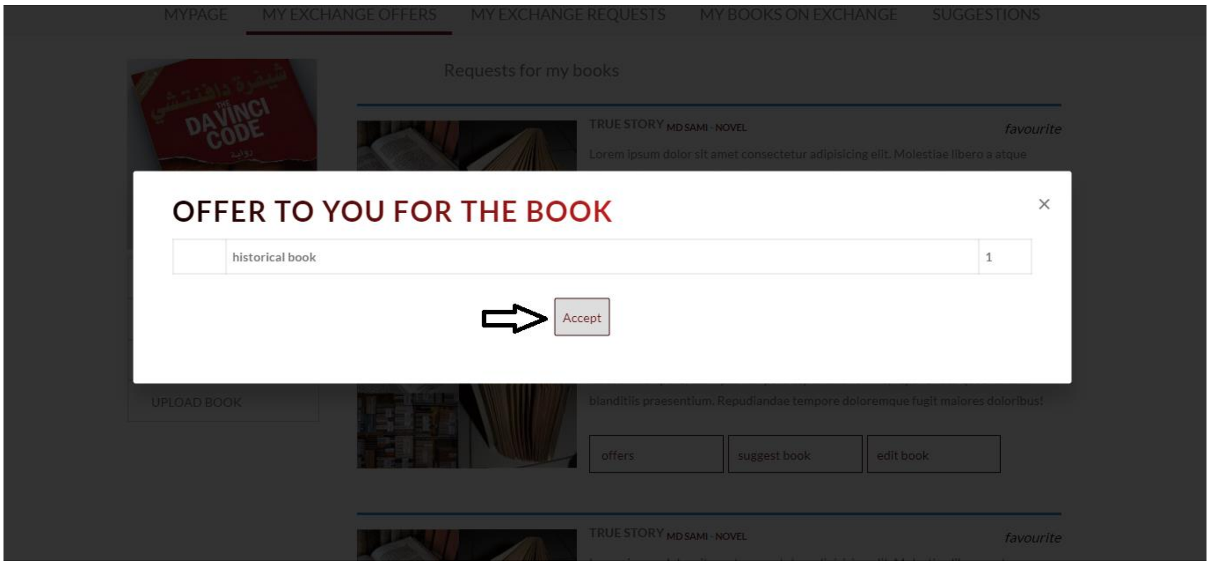
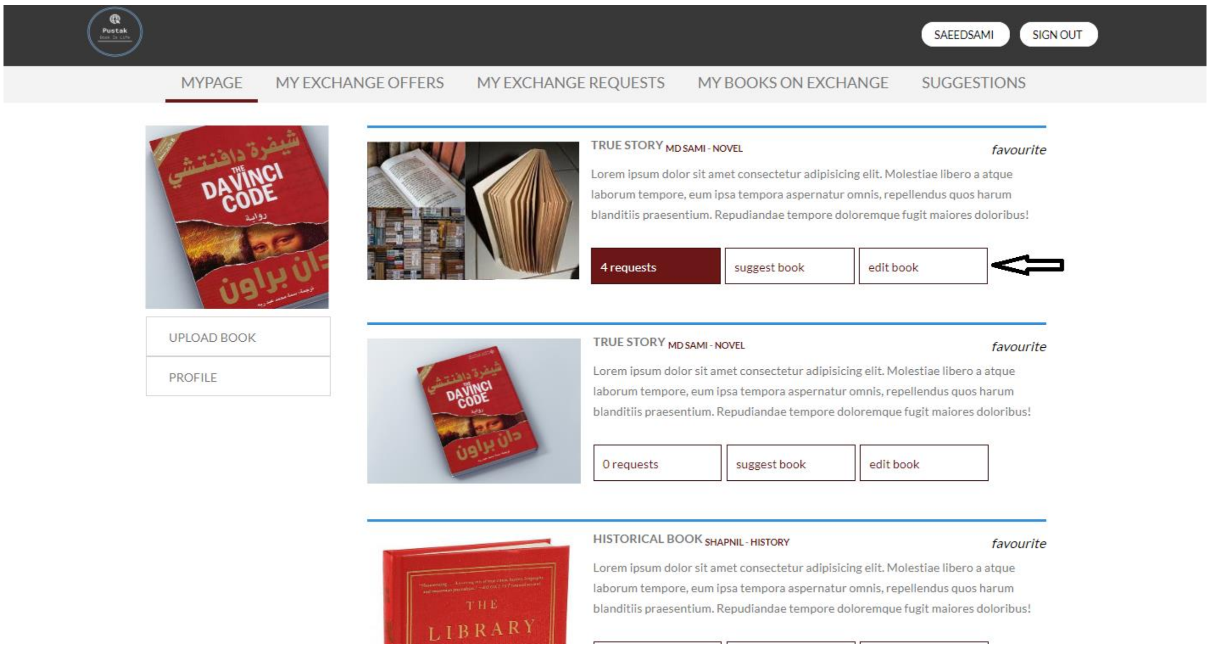


Figure 6.9: Exchange Offers

If offers are chosen, user will accept the exchange offers. Send the exchangeable books to the user.

My exchange offers

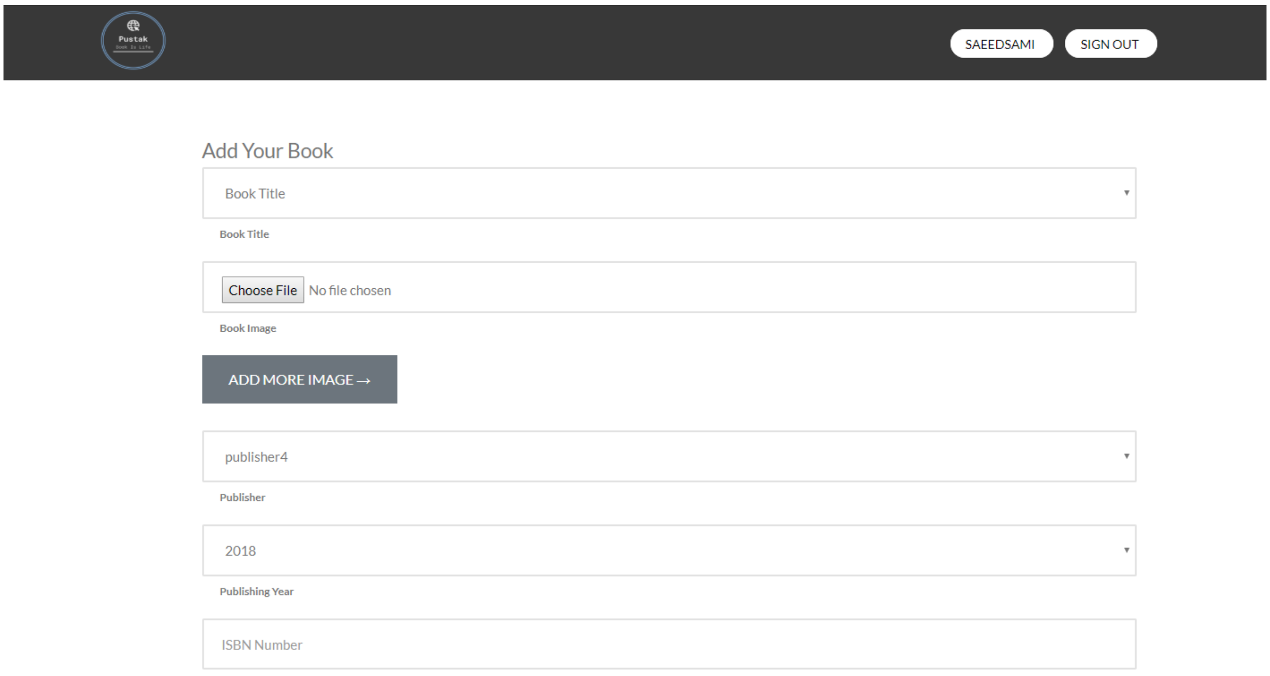
User edit his books.



Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

Figure 6.10: Edit Book

On edit page



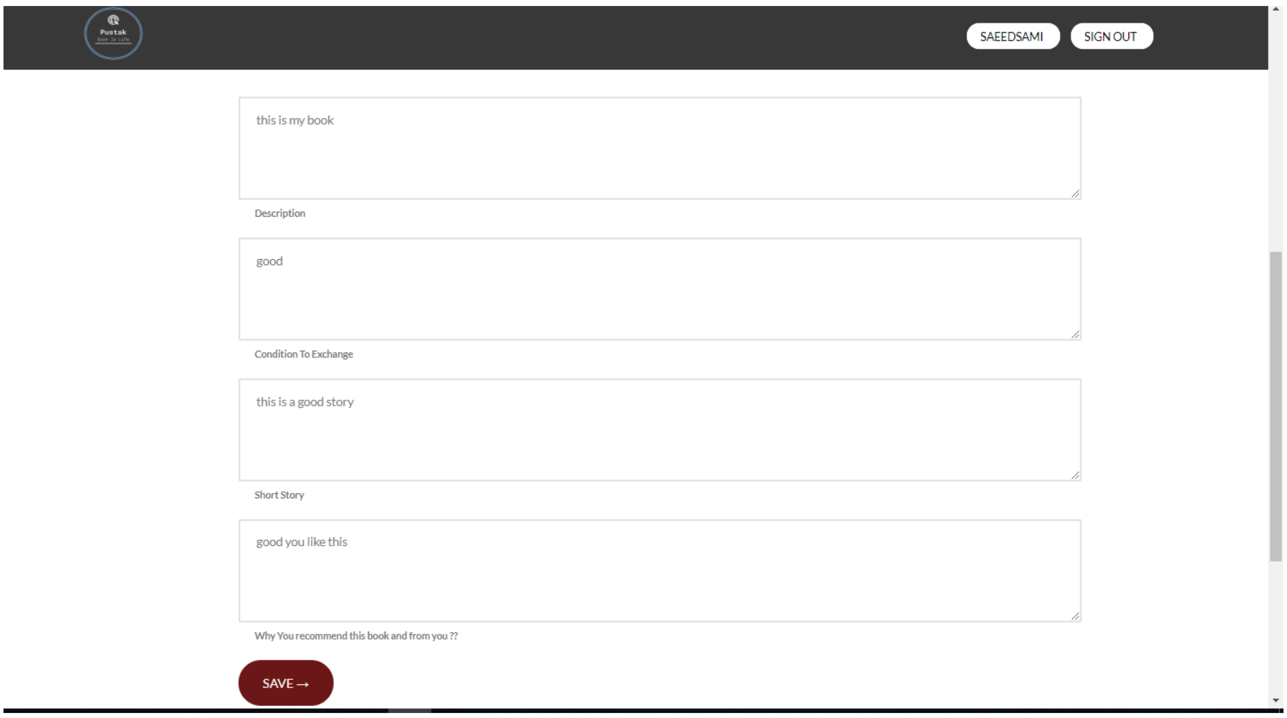
Book Title: Book Title

Book Image: No file chosen

Publisher: publisher4

Publishing Year: 2018

ISBN Number


SAEEDSAMI SIGN OUT

this is my book

Description

good

Condition To Exchange

this is a good story

Short Story

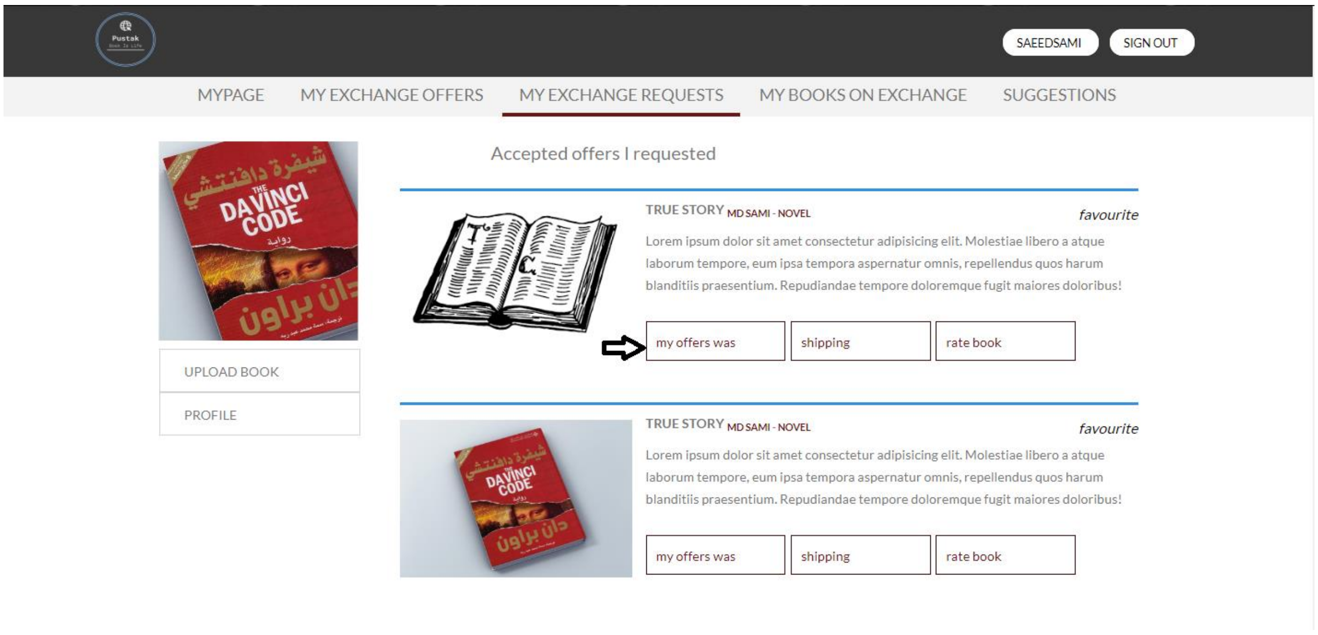
good you like this

Why You recommend this book and from you ??

SAVE →

Figure 6.11: Edit Book

Here in the my exchange request tab user can see the books he requested and which are accepted.


SAEEDSAMI SIGN OUT

MYPAGE MY EXCHANGE OFFERS MY EXCHANGE REQUESTS MY BOOKS ON EXCHANGE SUGGESTIONS

UPLOAD BOOK

PROFILE

Accepted offers I requested

TRUE STORY MD SAMI - NOVEL *favourite*

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

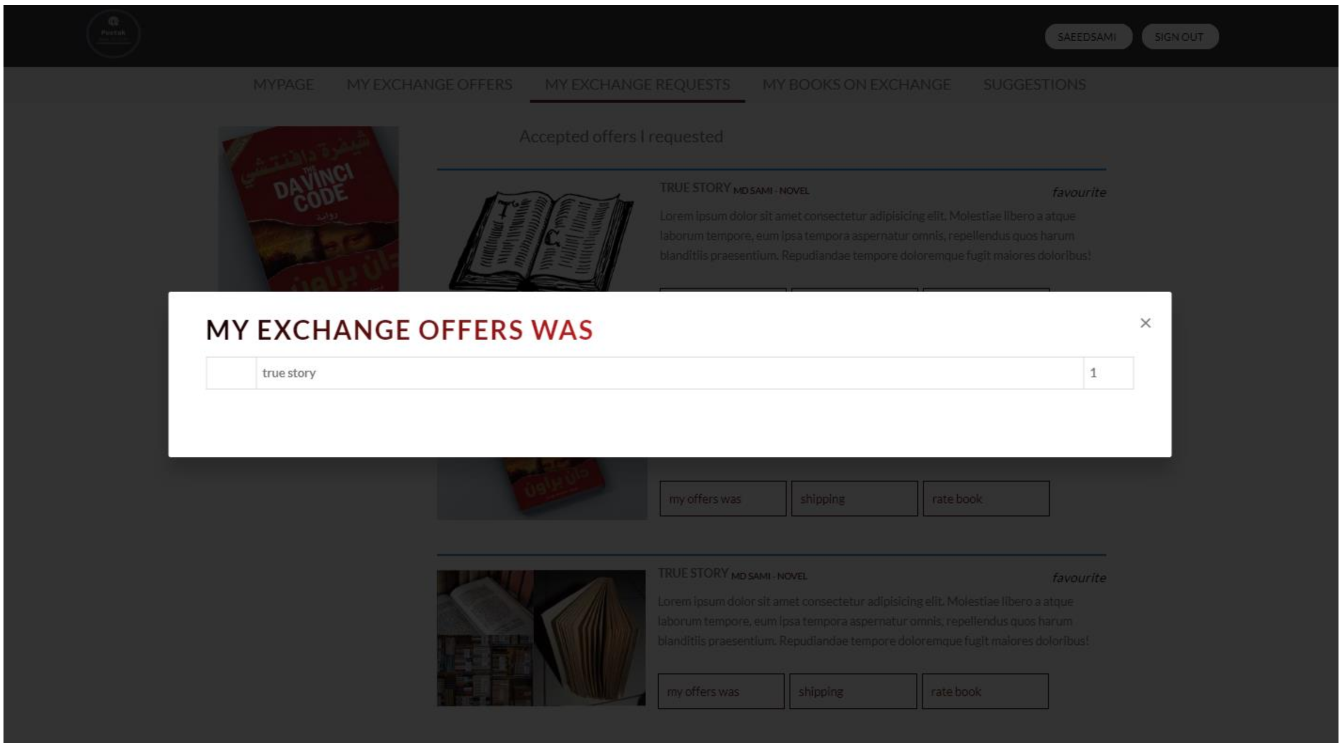
my offers was | shipping | rate book

TRUE STORY MD SAMI - NOVEL *favourite*

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

my offers was | shipping | rate book

User click on the indicated button to see his offers for this book.



Accepted offers I requested

TRUE STORY MD SAMI - NOVEL *favourite*

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

MY EXCHANGE OFFERS WAS

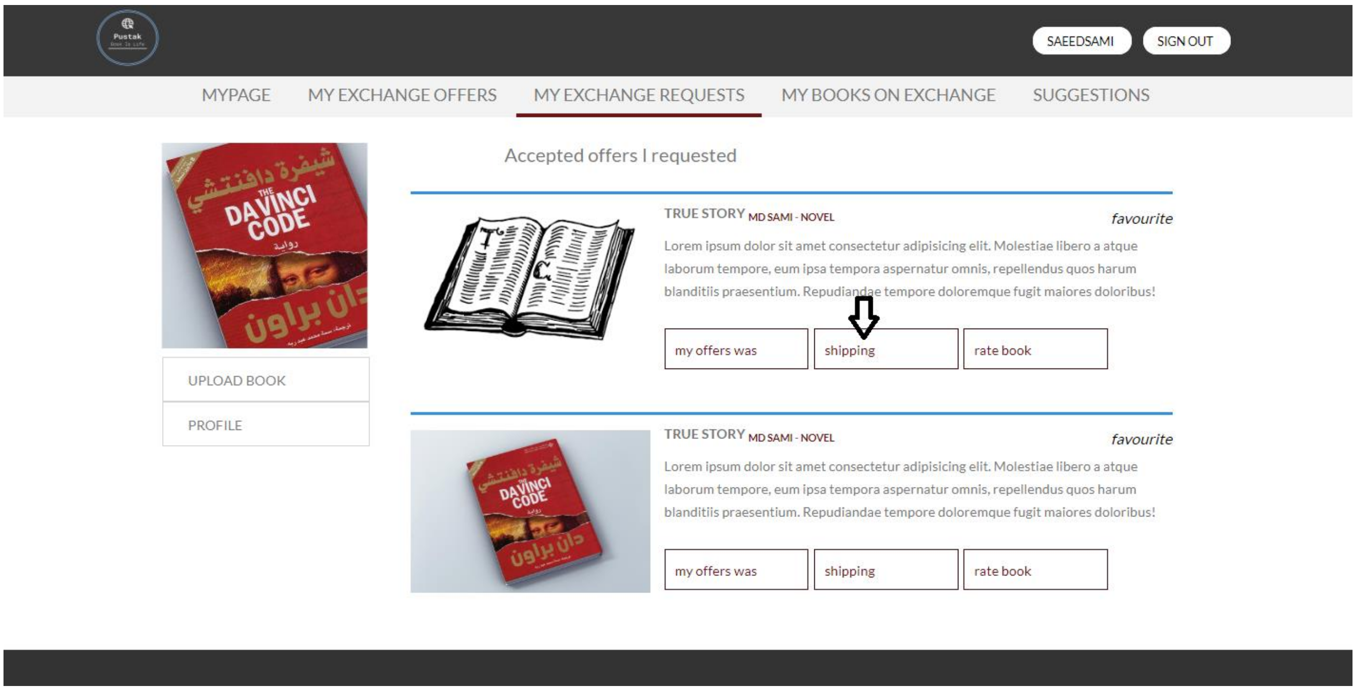
	true story	1

TRUE STORY MD SAMI - NOVEL *favourite*

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

Figure 6.12: My offers

User can see the shipping address for the book.



Accepted offers I requested

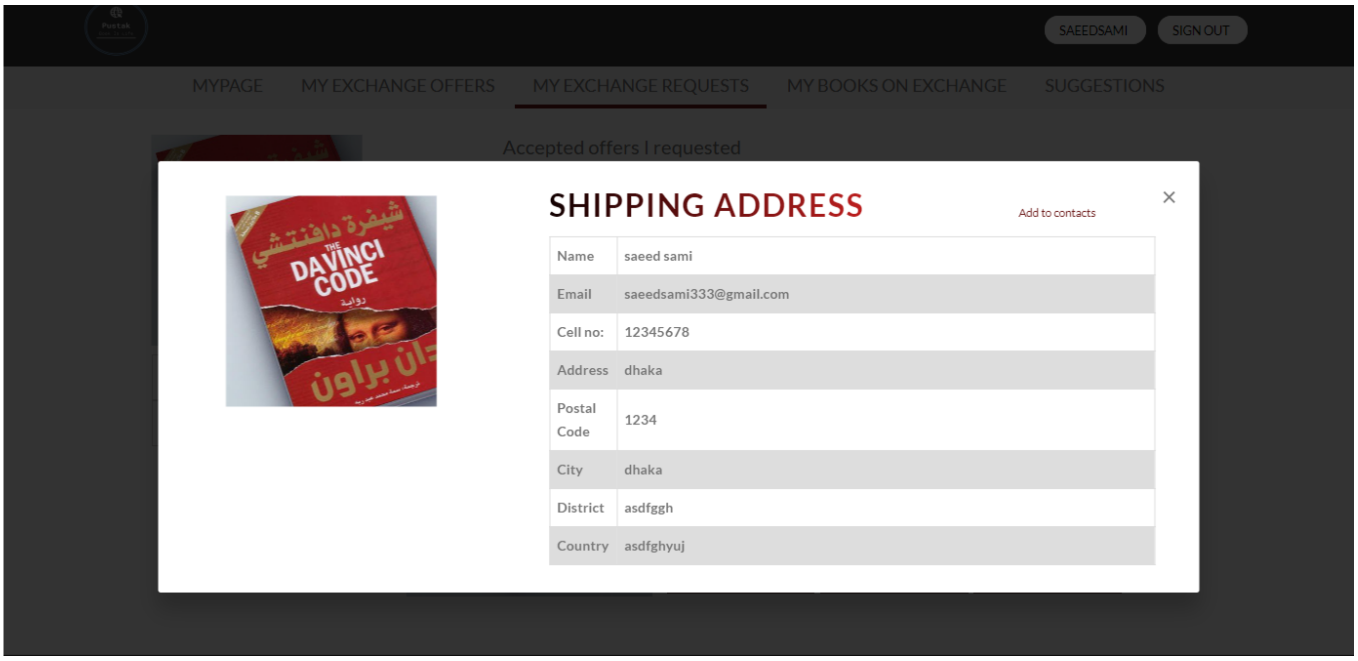
TRUE STORY MD SAMI - NOVEL *favourite*

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

TRUE STORY MD SAMI - NOVEL *favourite*

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

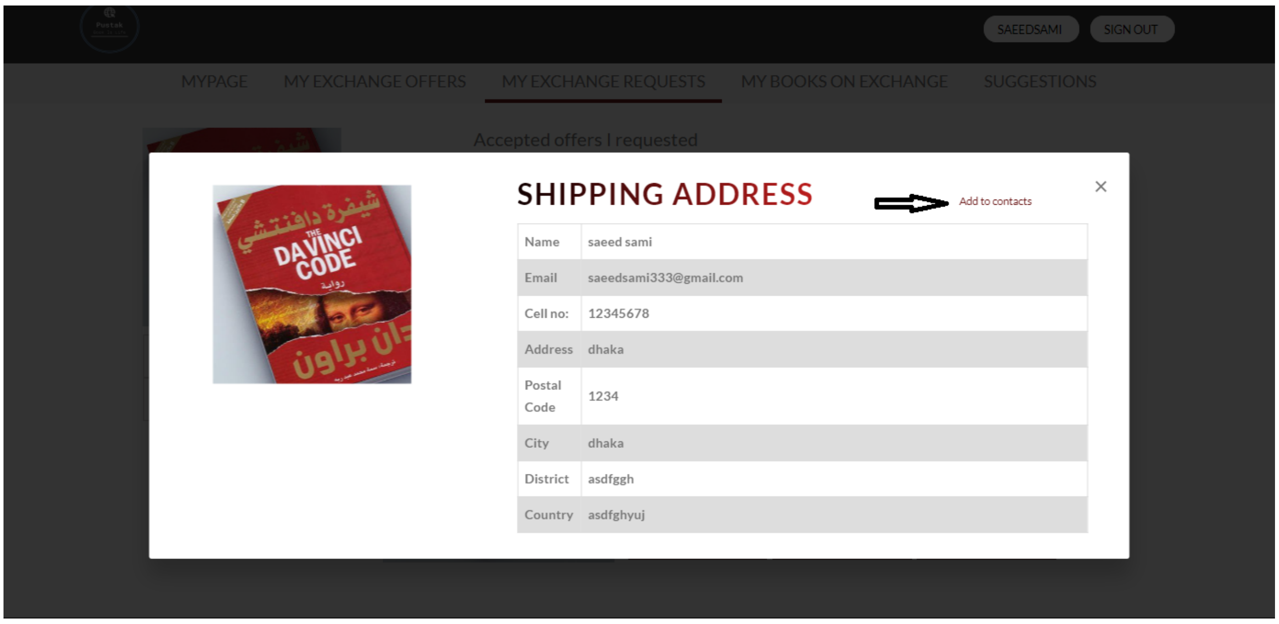
Shipping address



Name	saeed sami
Email	saeedsami333@gmail.com
Cell no:	12345678
Address	dhaka
Postal Code	1234
City	dhaka
District	asdfggh
Country	asdfghyuj

Figure 6.13: Shipping address

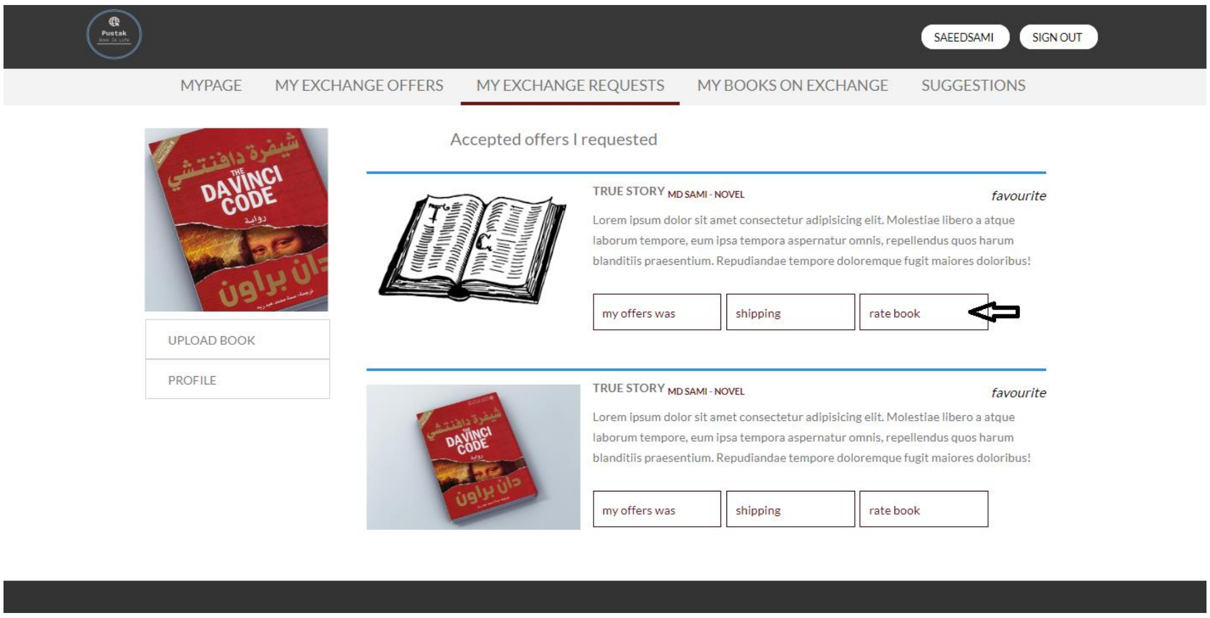
In the shipping address sytem allows user to create a contact. Add to contact or accept contact request.



Name	saeed sami
Email	saeedsami333@gmail.com
Cell no:	12345678
Address	dhaka
Postal Code	1234
City	dhaka
District	asdfggh
Country	asdfghyuj

Figure 6.14: Add to contacts

User rate exchanged book



Accepted offers I requested

TRUE STORY MD SAMI - NOVEL

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

TRUE STORY MD SAMI - NOVEL

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

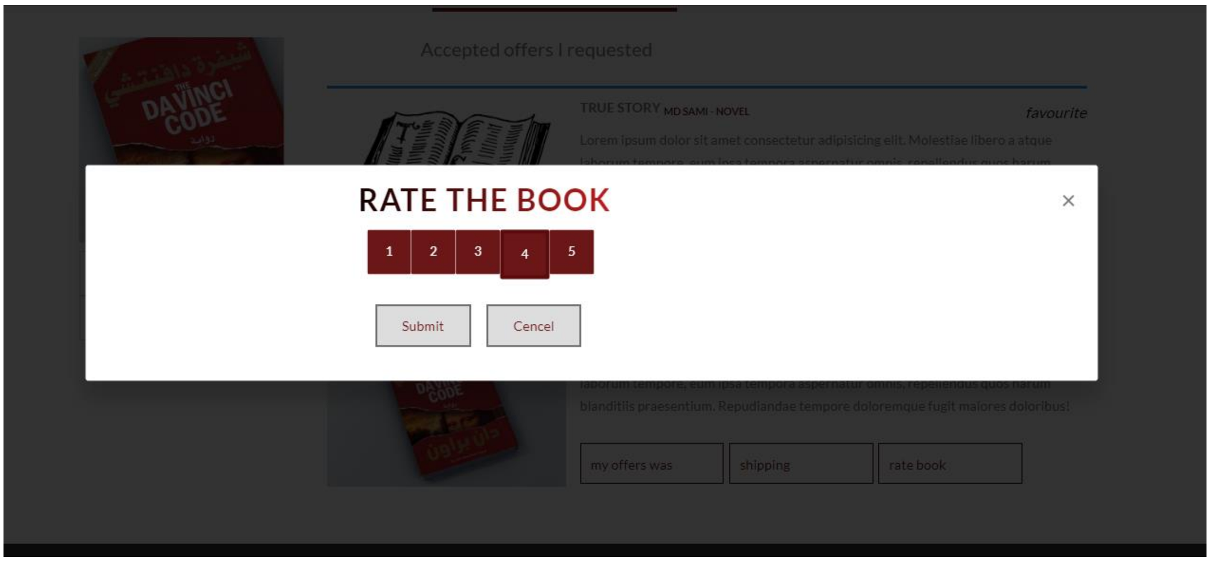
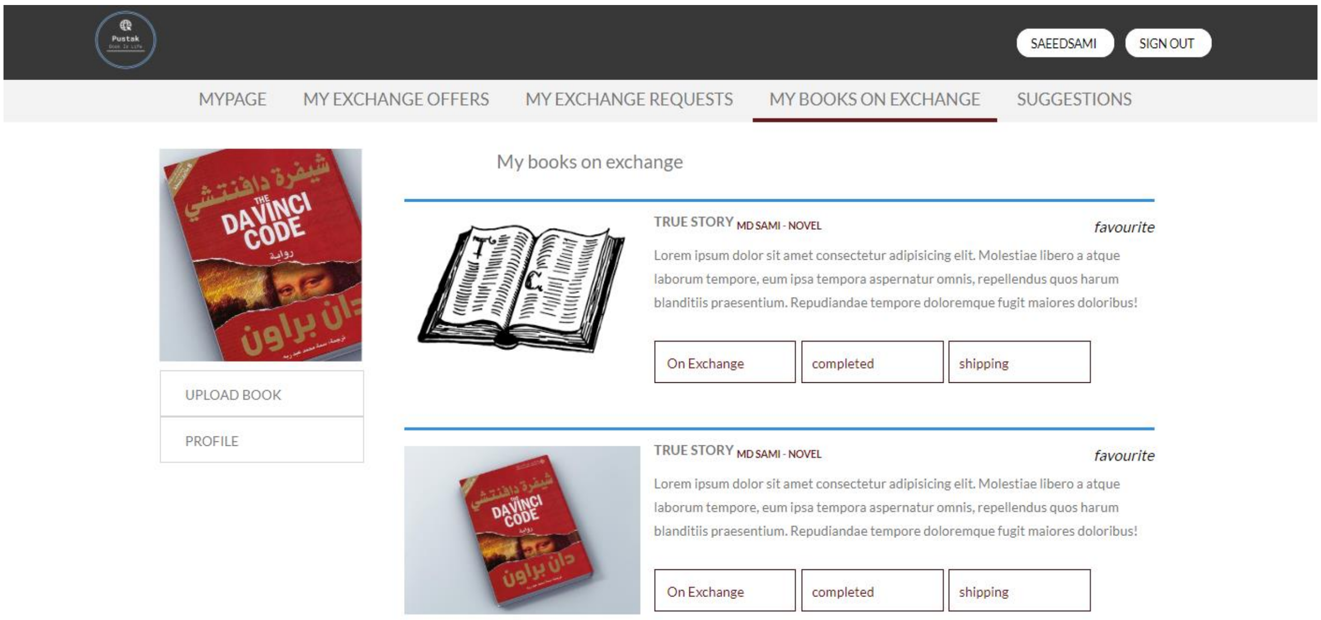


Figure 6.15: Rate book

My books on exchange shows user's books on exchange.



My books on exchange

TRUE STORY MD SAMI - NOVEL

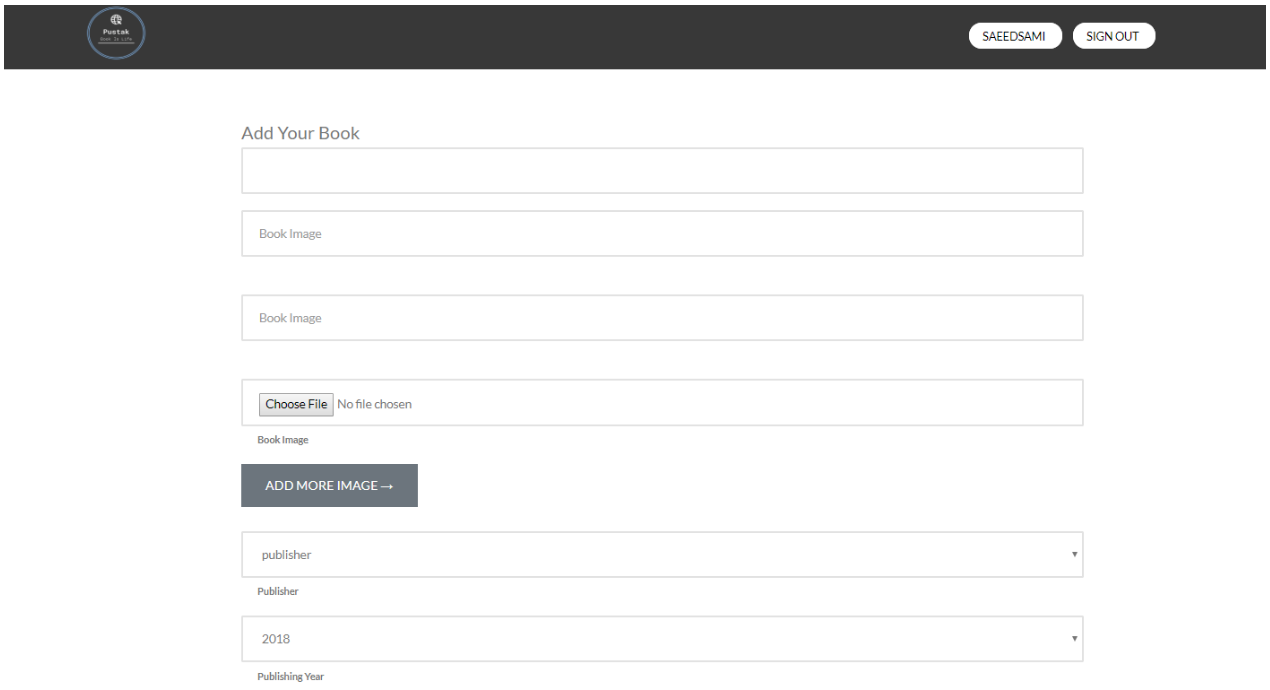
Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

TRUE STORY MD SAMI - NOVEL

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

Figure 6.16: My books on exchange

User add books.



Add Your Book

Book Image

Book Image

No file chosen

Book Image

publisher

Publisher

2018

Publishing Year

Figure 6.17: Add books

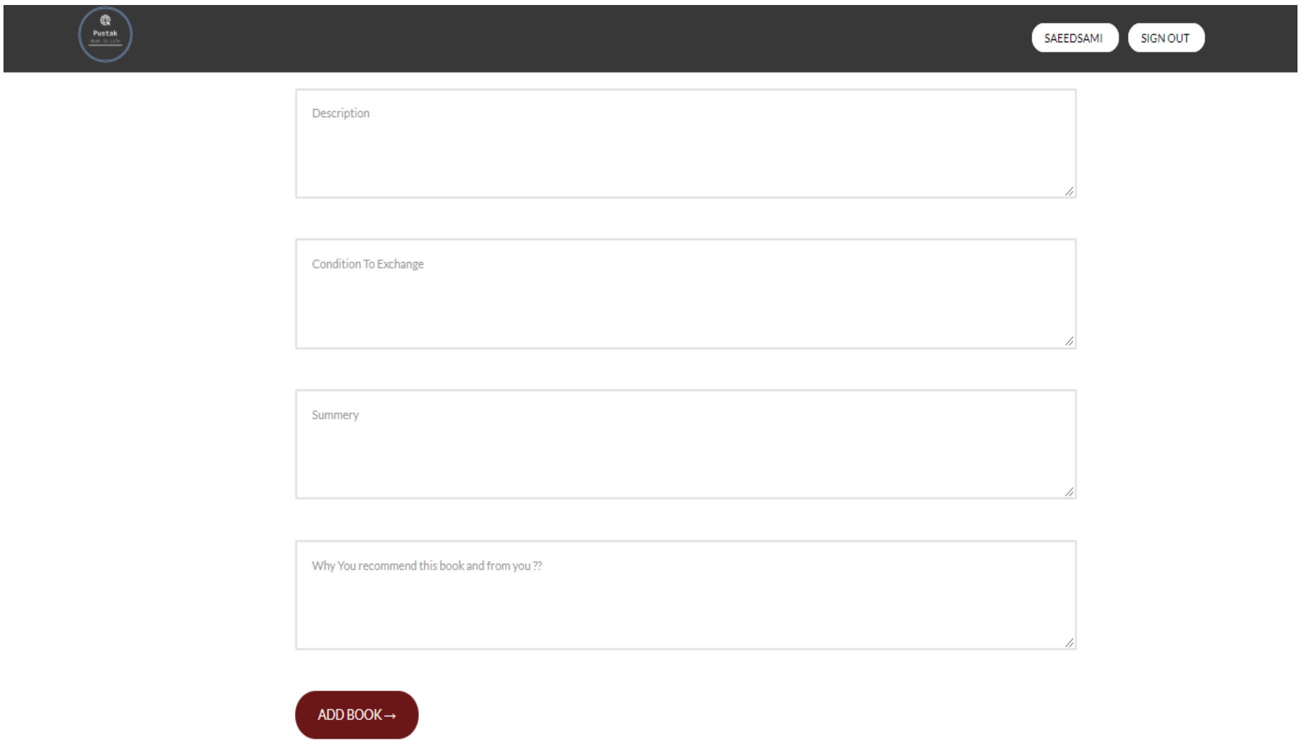
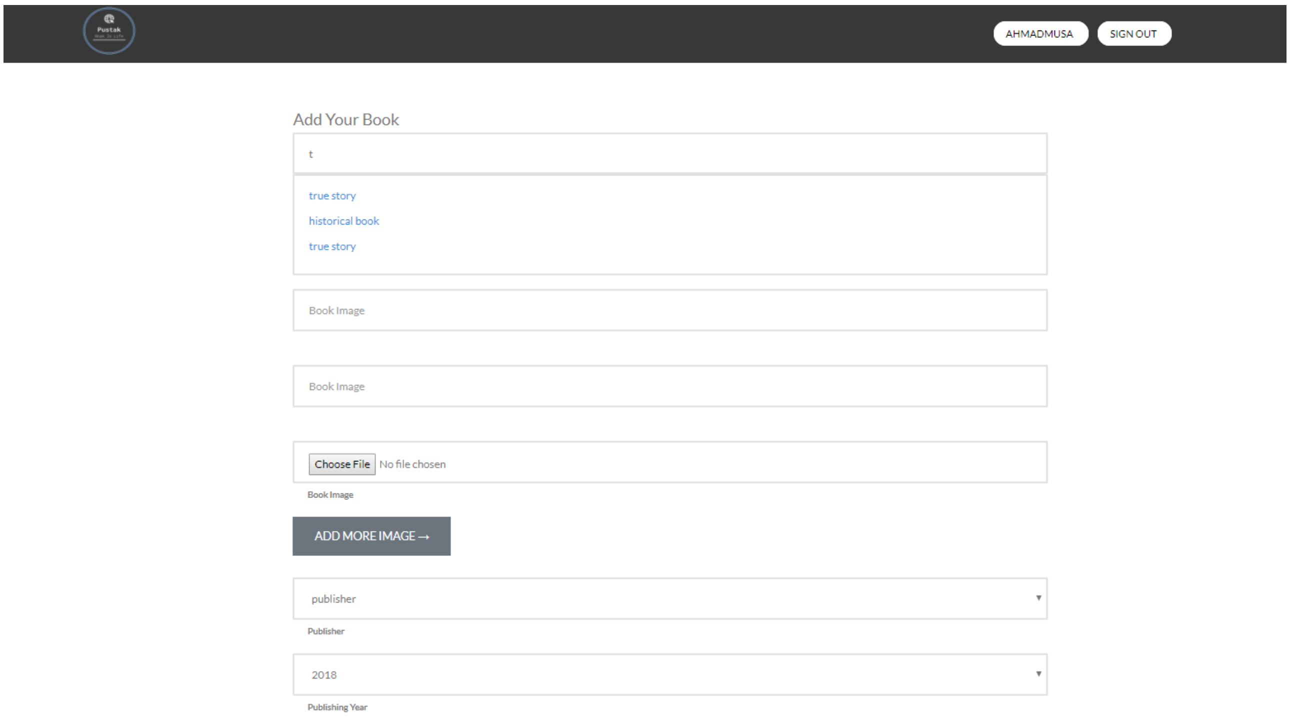
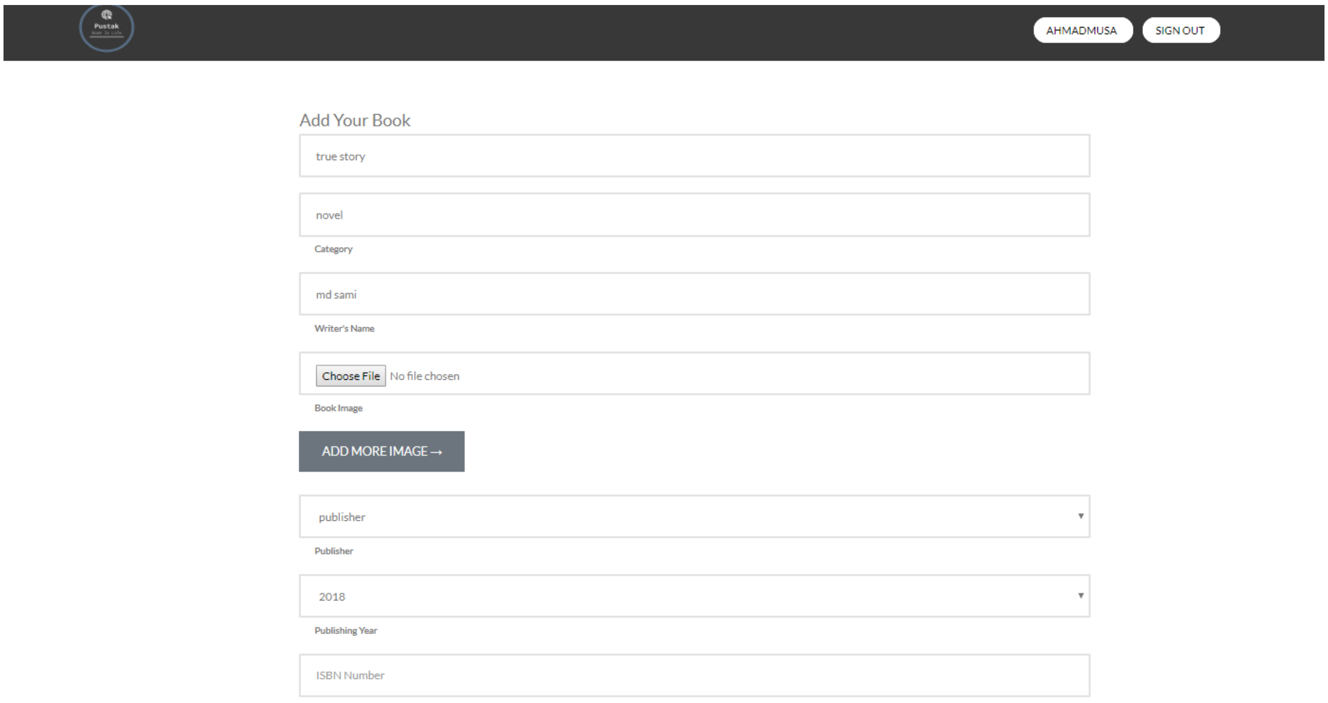


Figure 6.18: Add books

User getting suggestion by autocomplete when adding a book.



Selecting from suggestion it shows category and writer for that book.



Add Your Book

true story

novel

Category

md sami

Writer's Name

Choose File No file chosen

Book Image

ADD MORE IMAGE →

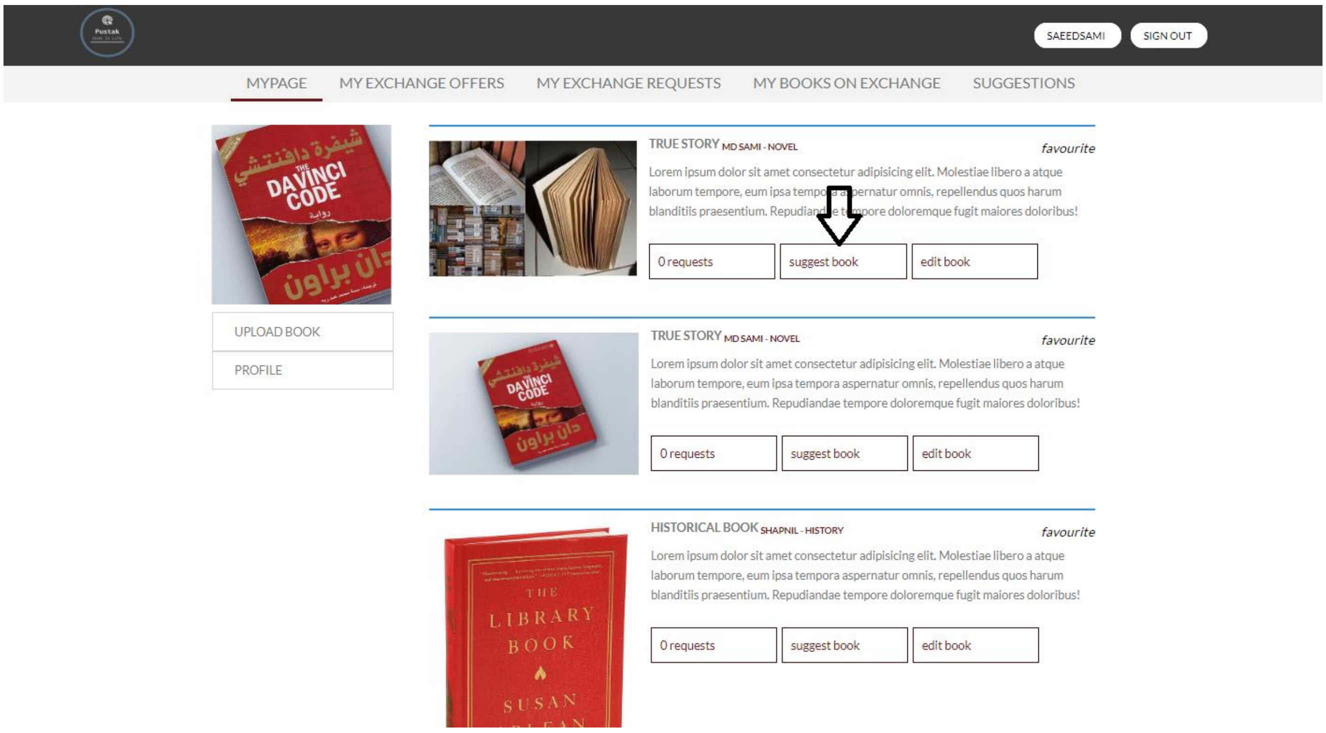
publisher

Publisher

2018

Publishing Year

ISBN Number



TRUE STORY MD SAMI - NOVEL favourite

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

0 requests | suggest book | edit book

TRUE STORY MD SAMI - NOVEL favourite

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

0 requests | suggest book | edit book

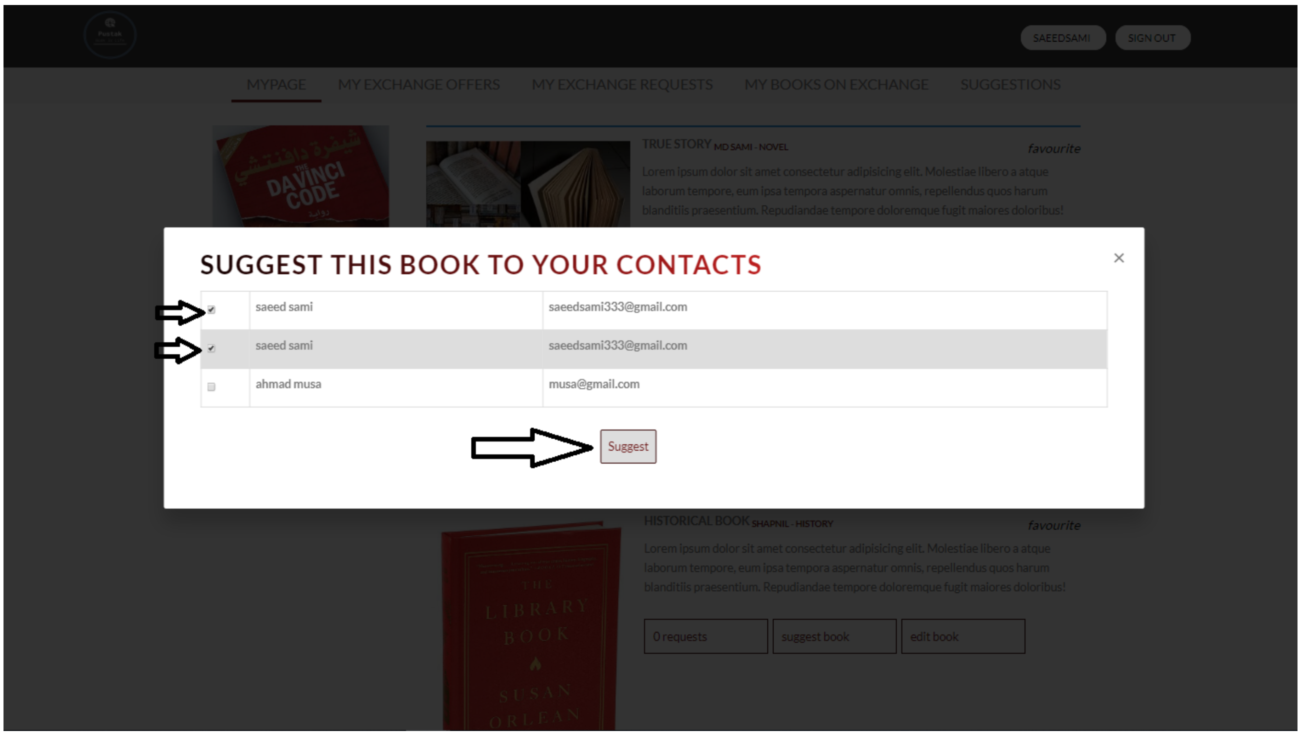
HISTORICAL BOOK SHAPNIL - HISTORY favourite

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

0 requests | suggest book | edit book

Figure 6.19: Suggest books

User can suggest any of his books to his contacts. Clocking this button there appears a popup with the options of his contacts to mark contacts and suggest multiple contacts.



TRUE STORY MD SAMI · NOVEL

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

SUGGEST THIS BOOK TO YOUR CONTACTS

✓	saeed sami	saeedsami333@gmail.com
✓	saeed sami	saeedsami333@gmail.com
☐	ahmad musa	musa@gmail.com

HISTORICAL BOOK SHAPNIL · HISTORY

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestiae libero a atque laborum tempore, eum ipsa tempora aspernatur omnis, repellendus quos harum blanditiis praesentium. Repudiandae tempore doloremque fugit maiores doloribus!

Figure 6.20: Suggest books

CHAPTER 7

CONCLUSION

Unavailability of every types of books and high expenses of books made us to think about such a platform where we can get easy access to any type of books and absolutely for free. Here our user can meet with many other users of different preference. This will make a big library with each of our user’s personal collections. So we started planning and designing the system. We have developed till now what we have presented here. There are lots of future planning and scope we have for this system.

Till now we have established a system where user can connect with others, exchange books, suggest and rate books. User will send exchanged books seeing the shipping address of other user.

7.1 Future Scope

- We are planning to develop a chat box for our better experience.
- Near future we are about to come with a single page application.
- Building web API will make our system available for each platform.

7.2 Limitations

- We couldn’t build a chat box with in this short time.
- We couldn’t build it for cross platform.
- This is not a single page application.

REFERENCES

1. www.kitabghor.com
2. https://www.bookcrossing.com
3. Thomas T. Barker, Writing Software Documentation, Second Edition , 1998
4. https://laravel.com/docs/5.7
5. Joe Lewis, Advanced CSS,2009
6. Olga Filipova, Learning Vue.js 2, December 13, 2016
7. Craig Larman, Applying UML and patterns, 2004

APPENDIX A

App\Model\UserBooksDetail

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class UserBooksDetail extends Model
{
  protected $table="userBooksDetails";

  public function user(){
     return $this->belongsTo('App\User');
  }

  public function image(){
     return $this->hasMany('App\Model\BookImage');
  }
  public function rating(){
   return $this->hasMany('App\Model\Rating');
}
  public function exchane(){
    return $this->hasMany('App\Model\Exchange');
 }
  public function exchaneDetails(){
    return $this->hasMany('App\Model\ExchaneDetails');
  }

  public function book(){
     return $this->belongsTo('App\Model\Book');
  }

}
```

App\Model\Book;

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class Book extends Model
{
    protected $table="books";

    public function userBooksDetail(){
       return $this->hasMany('App\Model\UserBooksDetail');
    }

    public function category(){
       return $this->belongsTo('App\Model\Category');
    }

    public function writer(){
       return $this->belongsTo('App\Model\Writer');
    }
}
```

App\Model\Category;

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
  protected $table="categories";

  public function book(){
     return $this->hasMany('App\Model\Book');
  }

}
```

App\Model\Writer;

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class Writer extends Model
{
  protected $table="writers";
  public function book(){
     return $this->hasMany('App\Model\Book');
  }

}
```

App\Model\BookImage

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class BookImage extends Model
{
  protected $table="images";

  public function userBooksDetail(){
     return $this->belongsTo('App\Model\UserBooksDetail');
  }

}
```

App\Model\Exchange

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;
```

```
class Exchange extends Model
{
  protected $table="exchange";

  public function exchangeDetail(){
    return $this->hasMany('App\Model\ExchangeDetails');
 }

 public function userBookDetail(){
  return $this->belongsTo('App\Model\UserBooksDetail');
}
}
```

App\Model\ExchangeDetails;

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class ExchangeDetails extends Model
{
  protected $table="exchangeDetails";

  public function exchange(){
    return $this->hasMany('App\Model\Exchange');
 }
  public function userBooksDetail(){
    return $this->belongsTo('App\Model\UserBooksDetail');
   }
}
```

App\Model\User;

```
<?php

namespace App;

use Illuminate\Notifications\Notifiable;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    public function userBooksDetail(){
       return $this->hasMany('App\Model\UserBooksDetail');
    }
    public function profile(){
       return $this->hasOne('App\Model\Profile');
    }
    public function contacts(){
        return $this->hasMany('App\Model\Contact');
     }

}
```

App\Model\Profile;

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class Profile extends Model
{
  protected $table="profile";

  public function user(){
     return $this->hasOne('App\User');
   }
}
```

App\Model\Rating;

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class Rating extends Model
{
    protected $table="ratings";

    public function userBooksDetail(){
        return $this->belongsTo('App\Model\UserBooksDetail');
     }

}
```

App\Model\Contact;

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class Contact extends Model
{
    protected $table="contacts";

    public function user(){
        return $this->belongsTo('App\User');
     }
}
```

App\Model\Suggestion;

```
<?php

namespace App\Model;

use Illuminate\Database\Eloquent\Model;

class Suggestion extends Model
{
    protected $table="suggestion";
}
```

All the migrations are given in the Appendix B.

APPENDIX B

2014_10_12_000000_create_users_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('fname');
            $table->string('lname');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')-
>nullable();
            $table->string('password');
            $table->string('status')->default(1);
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}
```

2014_10_12_100000_create_password_resets_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePasswordResetsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint
$table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
}
```

2018_10_20_155314_create_profile_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
```

```
class CreateProfileTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('profile', function (Blueprint $table)
{
            $table->increments('id');
            $table->integer('user_id');
            $table->string('image')->nullable();
            $table->string('phone_no')->nullable();
            $table->string('address')->nullable();
            $table->string('postal_code')->nullable();
            $table->string('city')->nullable();
            $table->string('district')->nullable();
            $table->string('country')->nullable();




            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('profile');
    }
}
```

2018_10_20_155355_create_book_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateBookTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('books', function (Blueprint $table) {
            $table->increments('id');
            $table->string('title');
            $table->integer('category_id');
            $table->integer('writer_id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('books');
    }
}
```

2018_10_20_155428_create_user_books_details

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUserBooksDetails extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('userBooksDetails', function (Blueprint
$table) {
            $table->increments('id');
            $table->integer('user_id');
            $table->integer('book_id');
            $table->string('isbn_no');
            $table->string('publishing_year');
            $table->string('publisher');
            $table->string('summery');
            $table->string('condition');
            $table->string('description');
            $table->string('why_from_me');
            $table->string('status');
            $table->string('is_delete');




            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('userBooksDetails');
    }
}
```

2018_10_20_155628_create_category_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateCategoryTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint
$table) {
            $table->increments('id');
            $table->string('category_name');

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('categories');
    }
}
```

2018_10_20_155744_create_writers_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateWritersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('writers', function (Blueprint $table)
{
            $table->increments('id');
            $table->string('writers_name');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('writers');
    }
}
```

2018_10_20_161259_create_book_image_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateBookImageTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('images', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('user_books_detail_id');
            $table->string('image');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('images');
    }
}
```

2018_11_21_075946_create_exchange_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
```

```
class CreateExchangeTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('exchange', function (Blueprint $table)
{
            $table->increments('id');
            $table->integer('from_id');
            $table->integer('to_id');
            $table->integer('user_books_detail_id');
            $table->boolean('status');

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('exchange');
    }
}
```

2018_11_21_080104_create_exchange-details_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateExchangeDetailsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('exchangeDetails', function (Blueprint
$table) {
            $table->increments('id');
            $table->integer('exchange_id');
            $table->integer('user_books_detail_id');
            $table->integer('qty');

            $table->timestamps();
        });
    }

    /**
```

```
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('exchangeDetails');
    }
}
```

2018_11_23_174916_create_rating_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateRatingTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('ratings', function (Blueprint $table)
{
            $table->increments('id');
            $table->integer('user_id');
            $table->integer('user_books_detail_id');
            $table->integer('ratingValue');

            $table->timestamps();

        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('ratings');
    }
}
```

2018_12_10_173414_create_contacts_table

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateContactsTable extends Migration
```

```
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('contacts', function (Blueprint $table)
{
            $table->increments('id');
            $table->integer('user_id');
            $table->integer('contact_id');

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('contacts');
    }
}
```

2018_12_10_182042_add_status_to_contacts

```
<?php
```

```
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class AddStatusToContacts extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('contacts', function (Blueprint $table)
{
            $table->boolean('status');

        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('contacts', function (Blueprint $table)
{
            dropIfExist('status');
        });
    }
}
```

2018_12_11_093530_create_suggestion_table

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateSuggestionTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('suggestion', function (Blueprint
$table) {
            $table->increments('id');
            $table->integer('suggest_id');
            $table->integer('user_books_detail_id');
            $table->integer('suggested_by');

            $table->timestamps();
        });
    }

    /**
```

```
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('suggestion');
    }
}
```

APPENDIX C
ROUTES

```
    <?php

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

// Route::get('/', function () {
//     return view('front.home');
// });
Route::get("/new", function(){
  return view('front.newBook');
});

Route::get('/', 'Front\HomeController@index')->name('front-home');
```

```
Route::get('/book-details',
'Front\HomeController@bookDetails')->name('front-book-
details');


Route::get('/signUp', 'Front\AuthController@signUp')-
>name('front-signUp');

Route::post('/signUp', 'Front\AuthController@store_signUp')-
>name('store-front-signUp');


Route::get('/signIn', 'Front\AuthController@signIn')-
>name('front-signIn');

Route::post('/signIn', 'Front\AuthController@store_signIn')-
>name('store-front-signIn');

Route::get('/sign-out', 'Front\AuthController@logout')-
>name('user-logout');

Route::get('/authenticate-require',
'Front\AuthController@authenticateRequire')->name('need-
authenticated');



Route::middleware('auth')->prefix('user')->group(function(){


  Route::get('/my-page', 'User\ProfileController@index')-
>name('user-my-page');


  Route::get('/add-book', 'User\BookController@addNew')-
>name('add-new-book');

  Route::post('/add-book', 'User\BookController@storeBook')-
>name('store-new-book');

  Route::get('/edit-book/{title}/{id}',
'User\BookController@showEditPage')->name('user-edit-book');

  Route::post('/edit-book',
'User\BookController@storeEditBook')->name('store-user-edit-
book');
```

```
  Route::post('/getBooks', 'User\BookController@getBooks')-
>name('getBooks');

  Route::get('/get-book-info',
'User\BookController@getBooksInfo');



  Route::get('/category',
'User\ProfileController@get_book_by_category')->name('book-by-
category');

  Route::get('/delete', 'User\BookController@delete')-
>name('book-delete');


  Route::post('/book-details',
'User\ExchangeController@storeExchange')->name('store-
exchange');

  Route::get('/my-books-requests',
'User\ProfileController@get_book_requests')->name('book-
requests');

  Route::post('/my-books-requests',
'User\ExchangeController@storeDecission')->name('store-
decission');

  Route::get('/my-exchange-log',
'User\ProfileController@exchangeLog')->name('exchange-log');

  Route::get('/my-books-on-exchange',
'User\ProfileController@bookOnExchange')->name('book-on-
exchange');

  Route::get('/my-suggestions',
'User\ProfileController@suggestions')->name('my-suggestions');


  Route::post('/my-page',
'User\ProfileController@storeProfile')->name('store-profile');

  Route::post('/my-rating',
'User\BookController@storeRating')->name('store-rating');


  Route::get('/got-my-book',
'User\ExchangeController@gotBook')->name('ex-complete');
```

```
  Route::get('/accept-contacts',
'User\ContactController@accept')->name('accept-contacts');

  Route::get('/send-req-contacts',
'User\ContactController@sendReq')->name('send-req-contacts');

  Route::post('/send-req-contacts',
'User\SuggestController@storeSuggestion')->name('store-
suggestion');




});
```