BookShare

TEAM-14

TEAM MEMBERS:

SWETHA CHANDRA KARROTI-21

PALLAVI RAMINENI-49

RAJARAMYA JANAGAMA-17

LATHA MUDDU-29

INTRODUCTION:

This Project allows users to share books among various registered users. This process of sharing books saves time. One need not go to the library and search for the books, rather they can just search for the required book using this app. The book collection centers (Library) are very vast and it is quite difficult to search for the required book in thousands of books available there. Instead of consuming the time, the user could easily get the same book by using this Application. The user could ask for the required book in the group. Then the user gets response regarding the availability of the book and the nearest pickup point.

OBJECTIVE:

Our basic point in selecting this project is to utilize the time effectively and to build up communication between people belonging to the same group. Also the user gets popup messages which reminds to collect the book on a particular date .Information about the return date of the book is also available in the application for which the user will get a popup message regarding the same.

FEATURES:

1. First each user must register in the login page of the application by providing the following fields:
    - User Name
    - Password
    - Full Name
    - Email Address

2. After registering into this application, each user must update all the books he has with him.
3. This directs to another page where he includes the following fields:
    - Book ISBN
    - Title
    - Author
    - Book Category
    - Availability – Default value 1.
4. On the other hand, the Borrower who wants to request for a book will have a Request option, where they can search for the books.
5. This search can be done with the following fields:
    - Search by Title
    - Search by Author
    - Search by Category
6. After the Borrower searches for a book, he gets the list of users having the book he requested.
7. Each user who has the book requested, specifies their Availability options which includes:
    - Date
    - Time

- Price

8. Now the borrower selects his preferences and chooses one among the available users.
9. Then a notification is sent to the selected user and he can accept or deny the request.
10. If the request is accepted then the user provides the location to collect the book.

EXISTING SERVICES/API:

1. Application was developed using Ionic Hybrid Mobile apps Framework on Jet Brains WebStorm IDE.

2. MongoDB is used for storing the data, MongoDB is being hosted on mlab.

3. Backend Services are built using Nodejs, Nodejs is hosted on Heroku platform.

4. Gravatar API is being used to pull the images of the users by computing MD5 hash of their email addresses and the image is being used in Account settings page.

DETAIL DESIGN OF FEATURES (Using Tools):

WIREFRAMES:

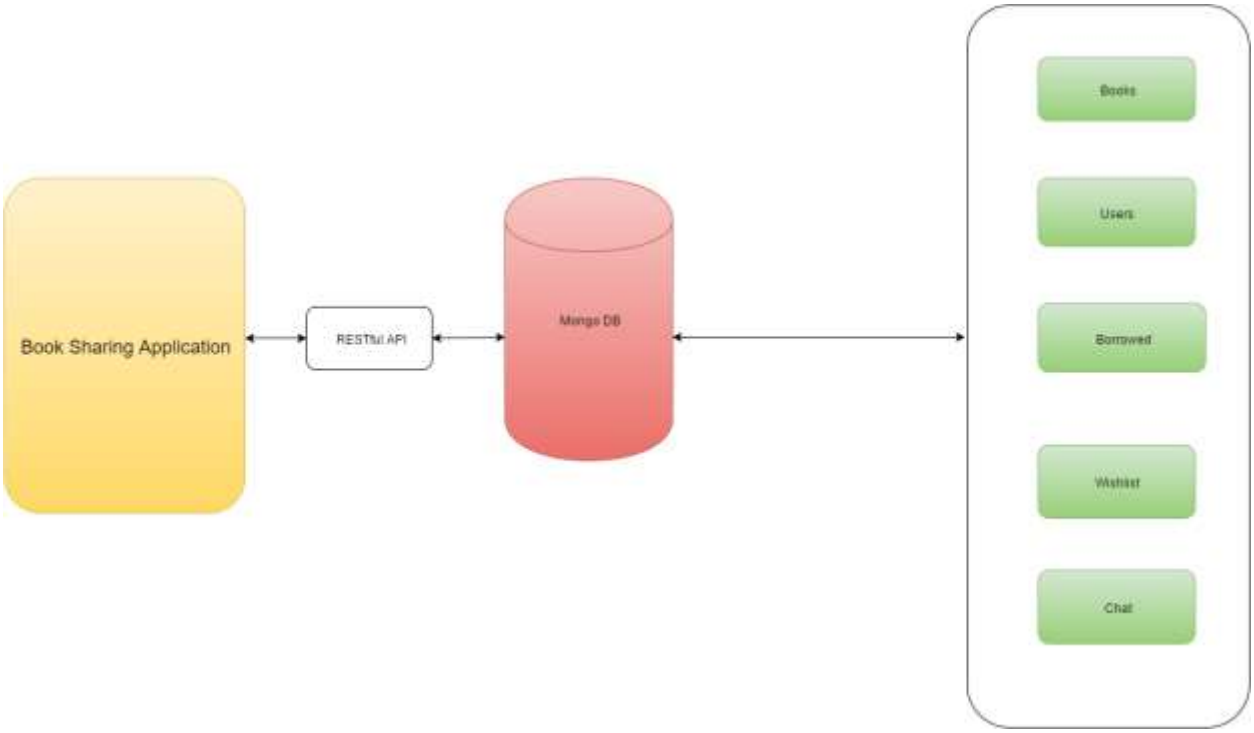

Welcome {{Username}}
Search
Available Books
Book Title
Author Name
Book Title
Author Name
Book Title
Author Name
Book Title
Author Name


< Dashboard
Book Title
Author Name
ISBN Number
Add to Wishlist
This book is available for borrowing from:
{{Username}}
Borrow

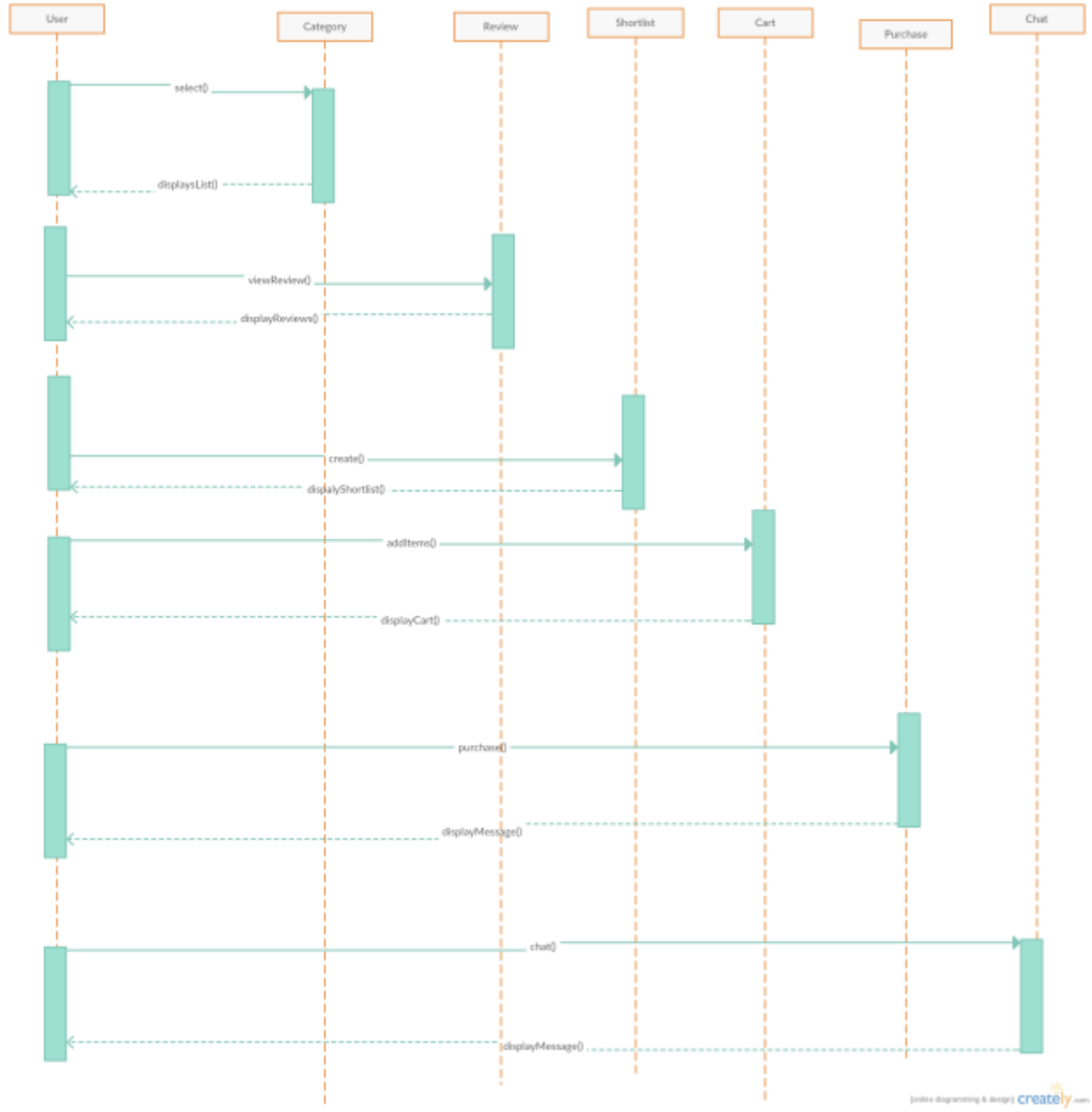

WishList
Book Title
Author Name
Book Title
Author Name
Book Title
Author Name
Book Title
Author Name


Account Settings
{{Username}}
New Password
user@gmail.com
Save Changes

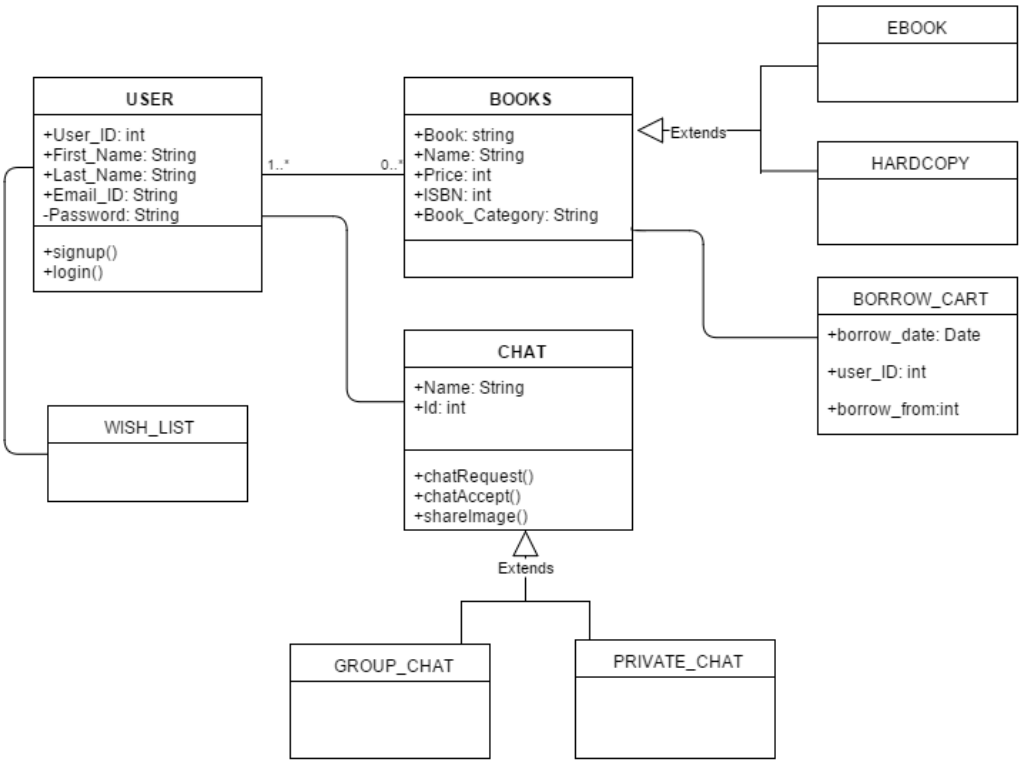
ARCHITECTURE DIAGRAM:



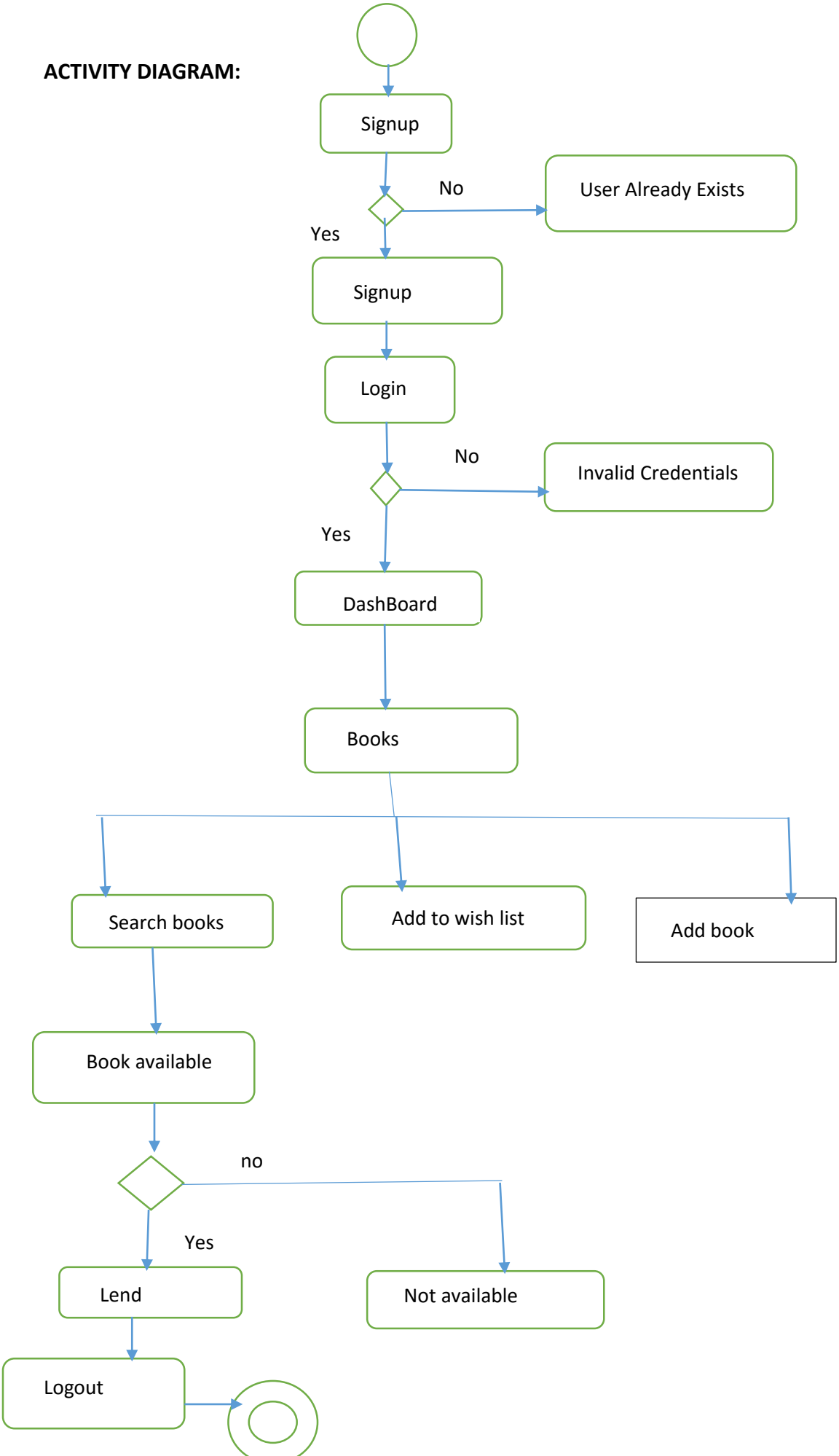
SEQUENCE DIAGRAM:



CLASS DIAGRAM:



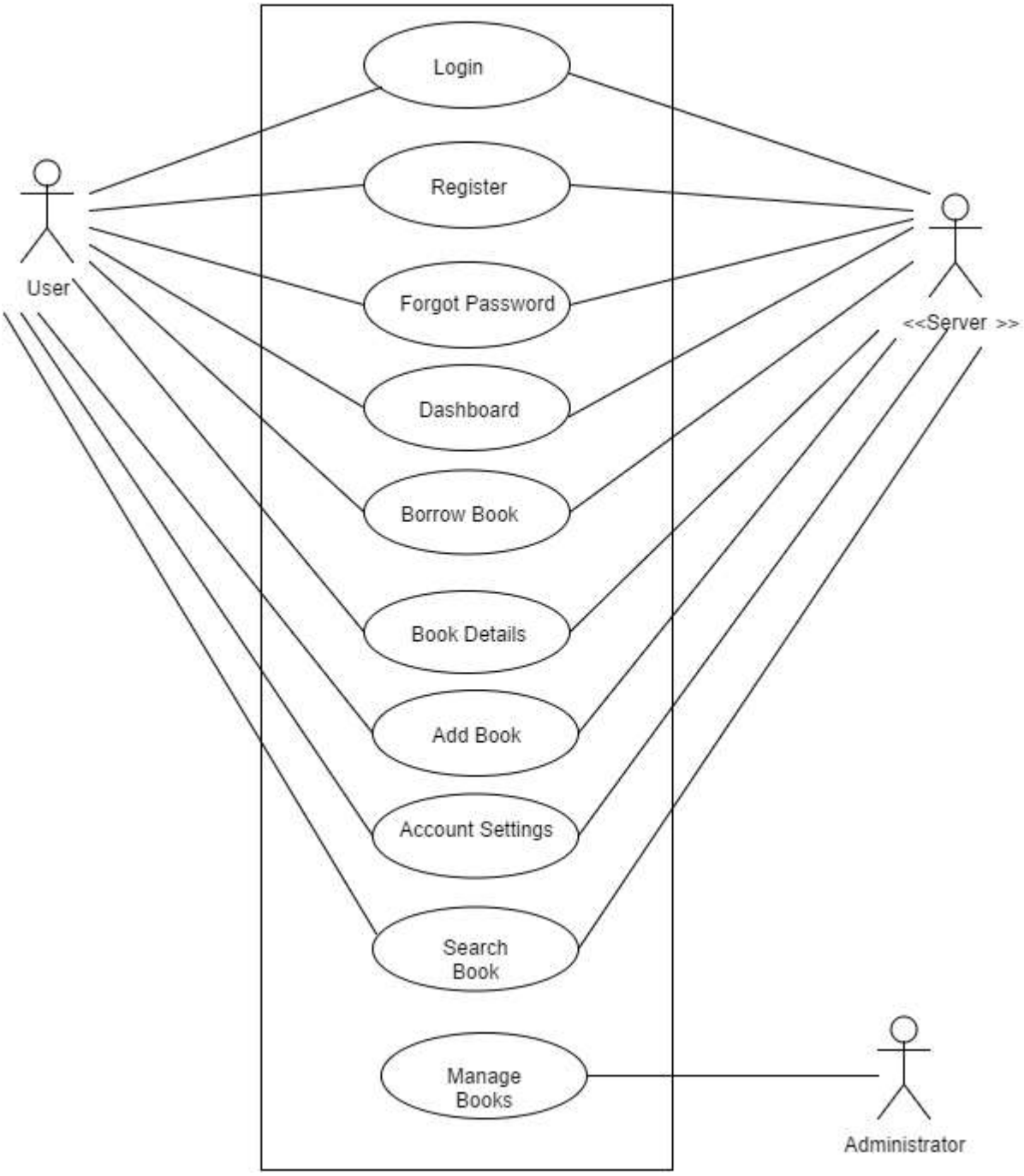
ACTIVITY DIAGRAM:



USER STORIES:

The user as of now will login with hard-coded username and password (admin & admin), upon login the user can search for the list of books available, upon tapping on a book title book details will be shown and the user can borrow the book or add the book to his/her wishlist.

USE CASE:



SERVICE DESCRIPTION:

This service helps students of a University to borrow books from fellow students at their convenience. Create alerts when books gets available or add books to wishlist and Text to speech API that reads out the description of the book.

TESTING:

UNIT TESTING:

Unit testing has been performed using Jasmine, a framework for testing JavaScript code. Karma, Karma – Command Line Interface and Angular Mocks have been installed. PhantomJS is being used for running automated tests. Test cases for Login and Registration are written.

```
describe('LoginCtrl', function() {

    var controller,
        deferredLogin,
        stateMock,
        ionicPopupMock;

    beforeEach(module('starter.controllers'));

    describe('#validateLogin', function() {


        it('should call login on userDashboard', function() {
            expect(userDashboard.login).toHaveBeenCalledWith('pr3md@mail.umkc.edu', 'password1');
        });

        describe('when the login is executed,', function() {
            it('if successful, should change state to dashboard', function() {

                expect(stateMock.go).toHaveBeenCalledWith('user-dash.home');
            });

            it('if login is unsuccessful, should show a popup', function() {

                expect(ionicPopupMock.alert).toHaveBeenCalled();
            });
        });
    })
});
```

```
describe('RegistrationCtrl', function() {

    var controller,
        deferredLogin,
        stateMock,
        ionicPopupMock;

    beforeEach(module('starter.controllers'));

    describe('#register', function() {


        describe('when the register is executed,', function() {
            it('if successful, should change state to login', function() {

                expect(stateMock.go).toHaveBeenCalledWith('tab-login');
            });

            it('if registration is unsuccessful, should show a popup', function() {

                expect(ionicPopupMock.alert).toHaveBeenCalled();
            });
        });
    })
});
```

IMPLEMENTATION:

MOBILE CLIENT IMPLEMENTATION:

The app is being developed using Ionic Framework. Functionality has been added to the controllers in this increment. Login, Registration and Adding a new book to the Collection features have been introduced. All these controller's functions make a **$http** call to the backend services whenever they are triggered by an event associated with them. In turn the backend services respond to the type of call and communicate with MongoDB to fetch the information requested and send the response back to the user.

Gravatar is being used to display the Profile picture of the user. The email address supplied by the user during registration is retrieved and a MD5 hash is computed for the same and a GET call is sent to Gravatar API to pull the image corresponding to the hash.

SERVER IMPLEMENTATION:

1. **Database:** MongoDB is being used for storing information about users, books and wish list. The database is currently hosted on mlab servers. As of now there are 2 collections defined. They are users and books. In Users collection, the document will have the fields: email, password, full name with the index as email. In books collection, the document will have the fields: Book Title, Author, ISBN and availability (which is automatically set to 1 – stating available).
2. **Service composition:** Backend services are developed with Nodejs. As of now three services are developed, they are: Processing Login, Registration and Adding new Book into the collection. The Nodejs application has been deployed to Heroku platform.

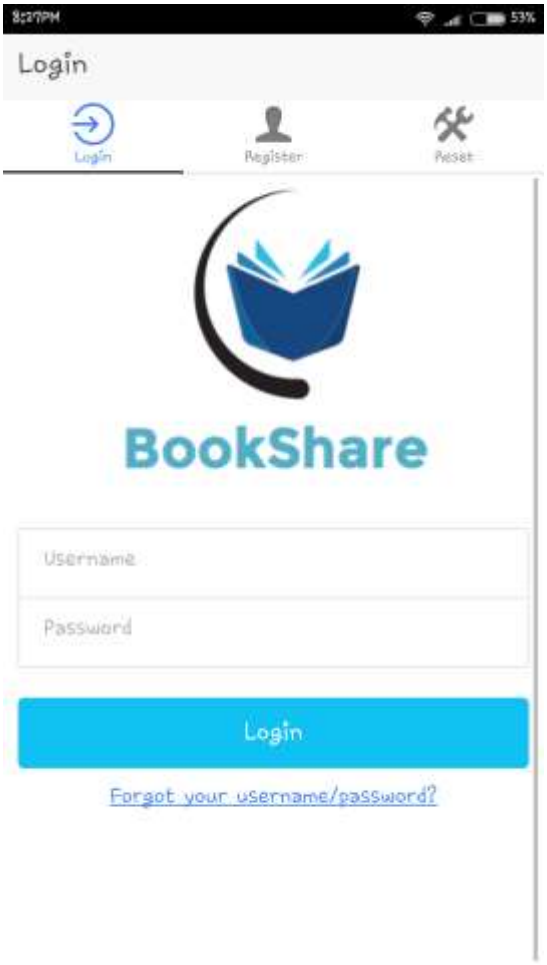
Service	Method	Service End Point	Request Payload	Response
Login	GET	https://floating-plateau-55000.herokuapp.com/bookshare/api/auth/login	Username, Password	Success – Fullname, Email
Register	POST	https://floating-plateau-55000.herokuapp.com/bookshare/api/auth/register	Email, Password, Full Name	Success – Document ID, Username, Fullname
Add Book	POST	https://floating-plateau-55000.herokuapp.com/bookshare/api/book/new	Book Title, Author, ISBN	Success – Document ID
Users	GET	https://floating-plateau-55000.herokuapp.com/bookshare/api/getUsers	-	Email, Fullname
Books	GET	https://floating-plateau-55000.herokuapp.com/bookshare/api/allbooks	-	Book Title, Author, Owner

DEPLOYMENT:

1. The Mobile application has been deployed to Ionic Framework servers and the working application can be viewed using Ionic View mobile app.
2. Web Portal for user administration has been deployed to DigitalOcean droplet.
3. Backend services are deployed to Heroku Nodejs platform.
4. MongoDB has been hosted on mlab.
5. All the source code has been pushed to Github.

SCREENSHOTS:

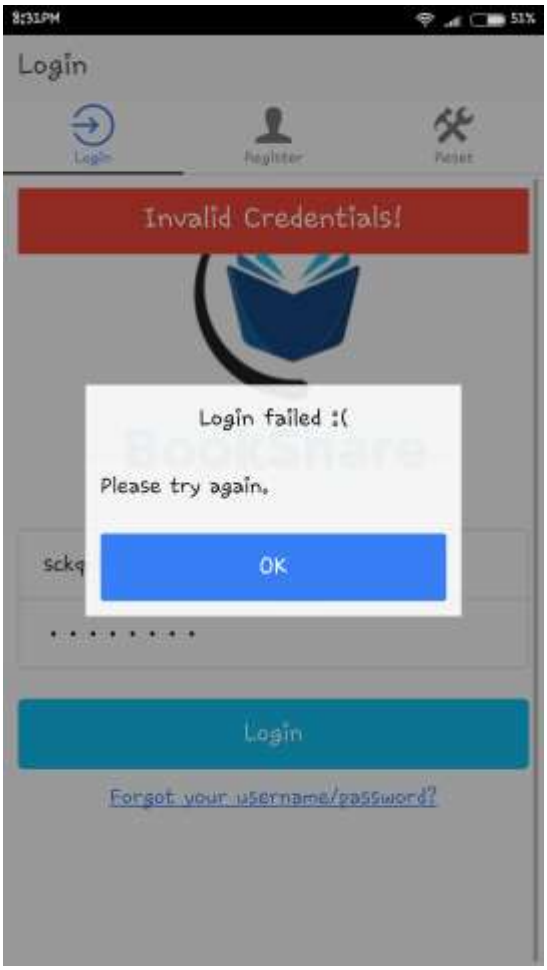
Login and Registration pages:



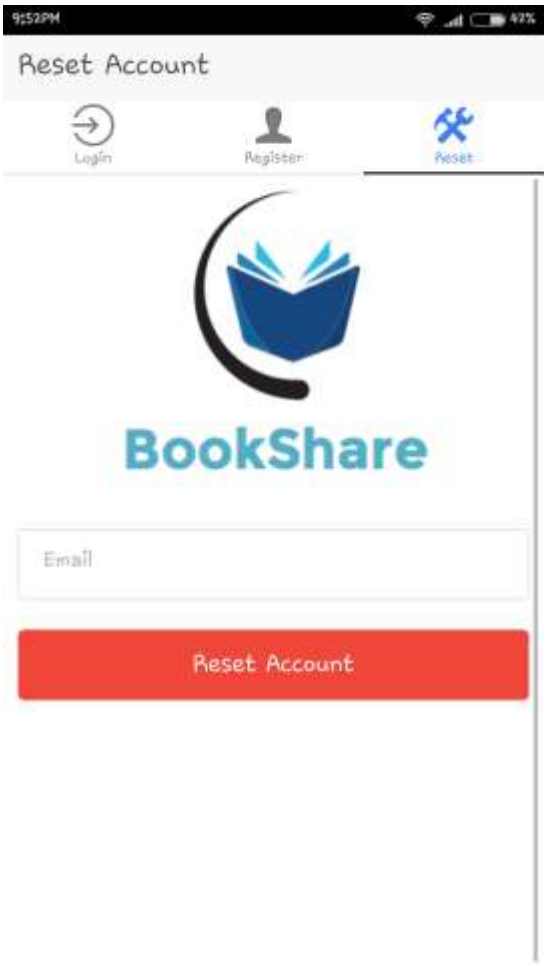
Registration of Existing Users:

Registration of New Users:

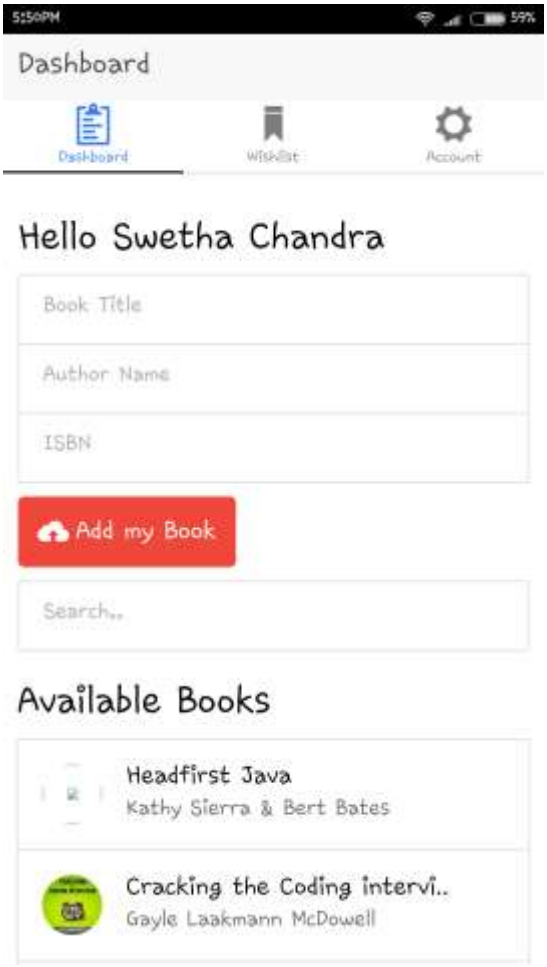
Invalid Credential and Successful Login:

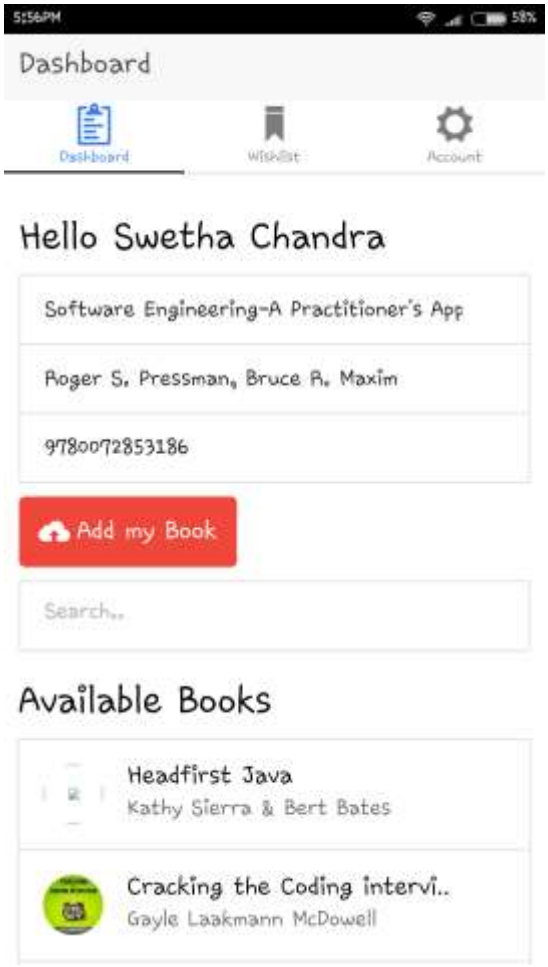


Forgot Password:

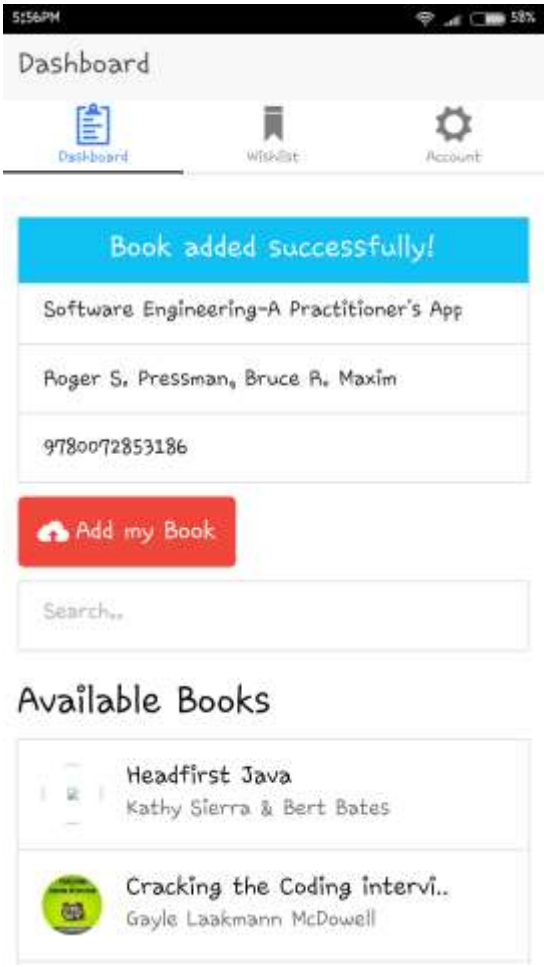


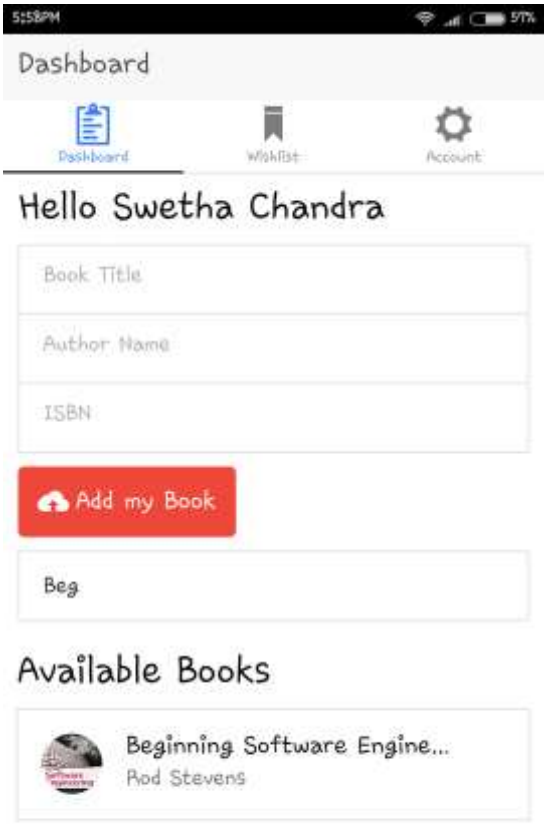
User's Dashboard after Successful Login and Adding Books:





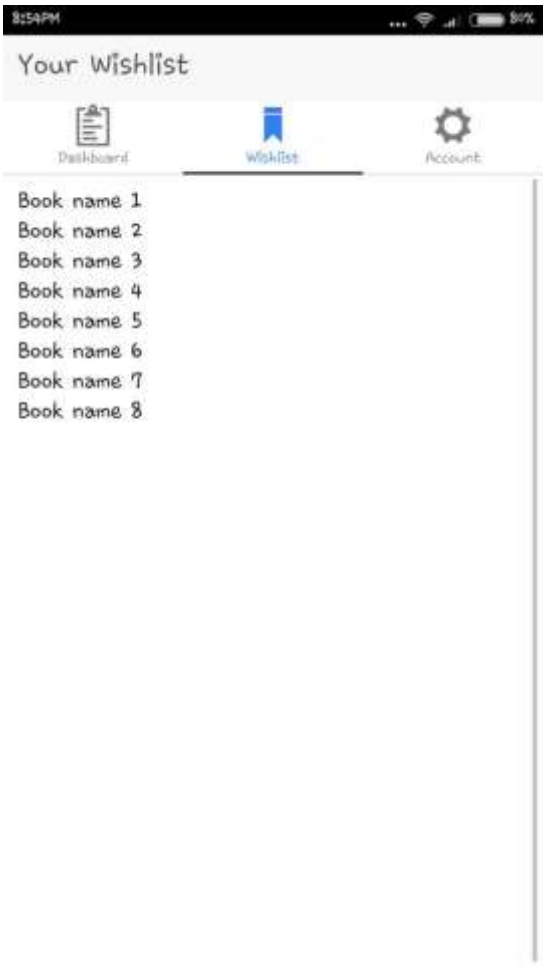
Book Added Successfully and Search for a Book:

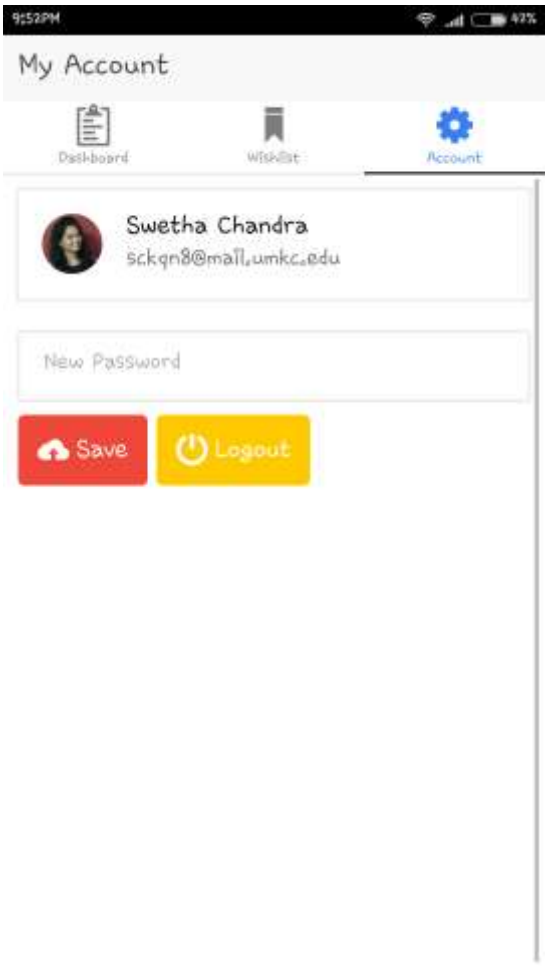




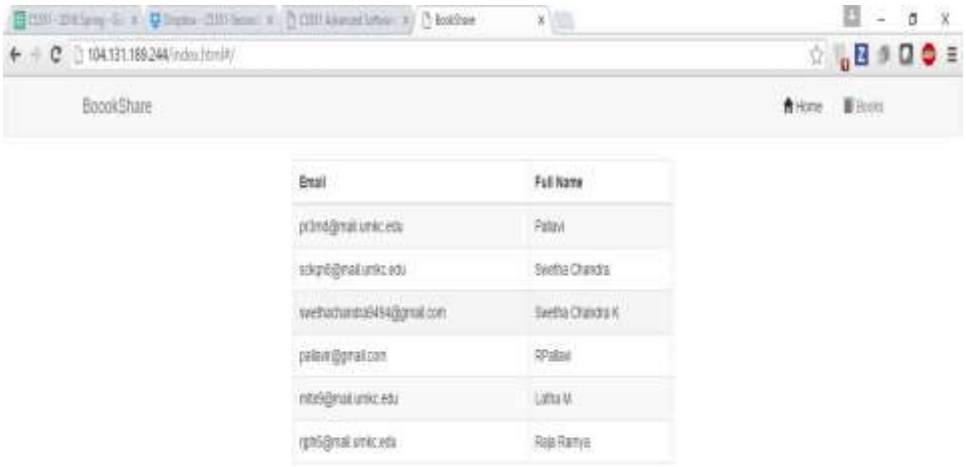
Selected Books can be added to Wishlist:

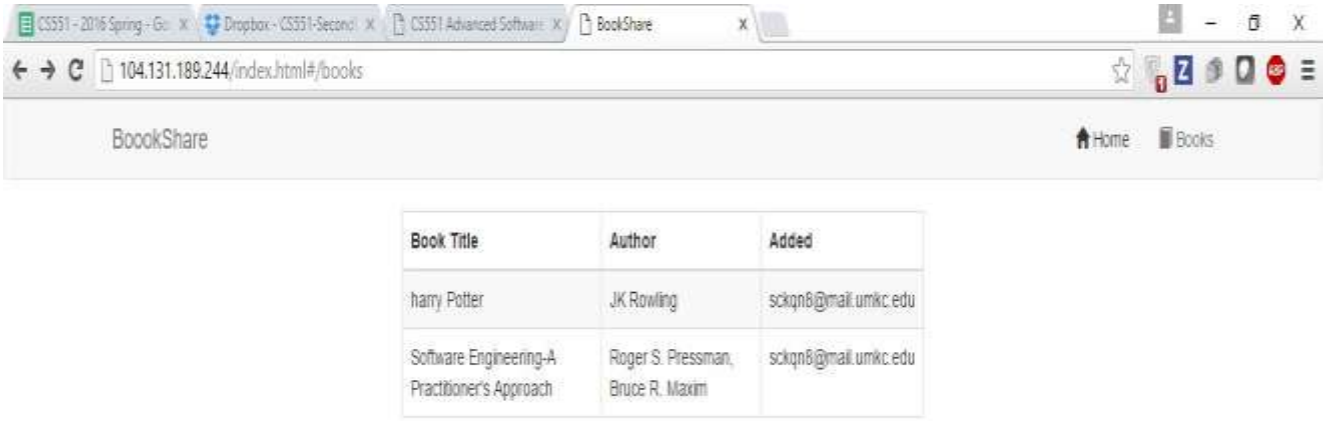
Wishlist and Logout Option:





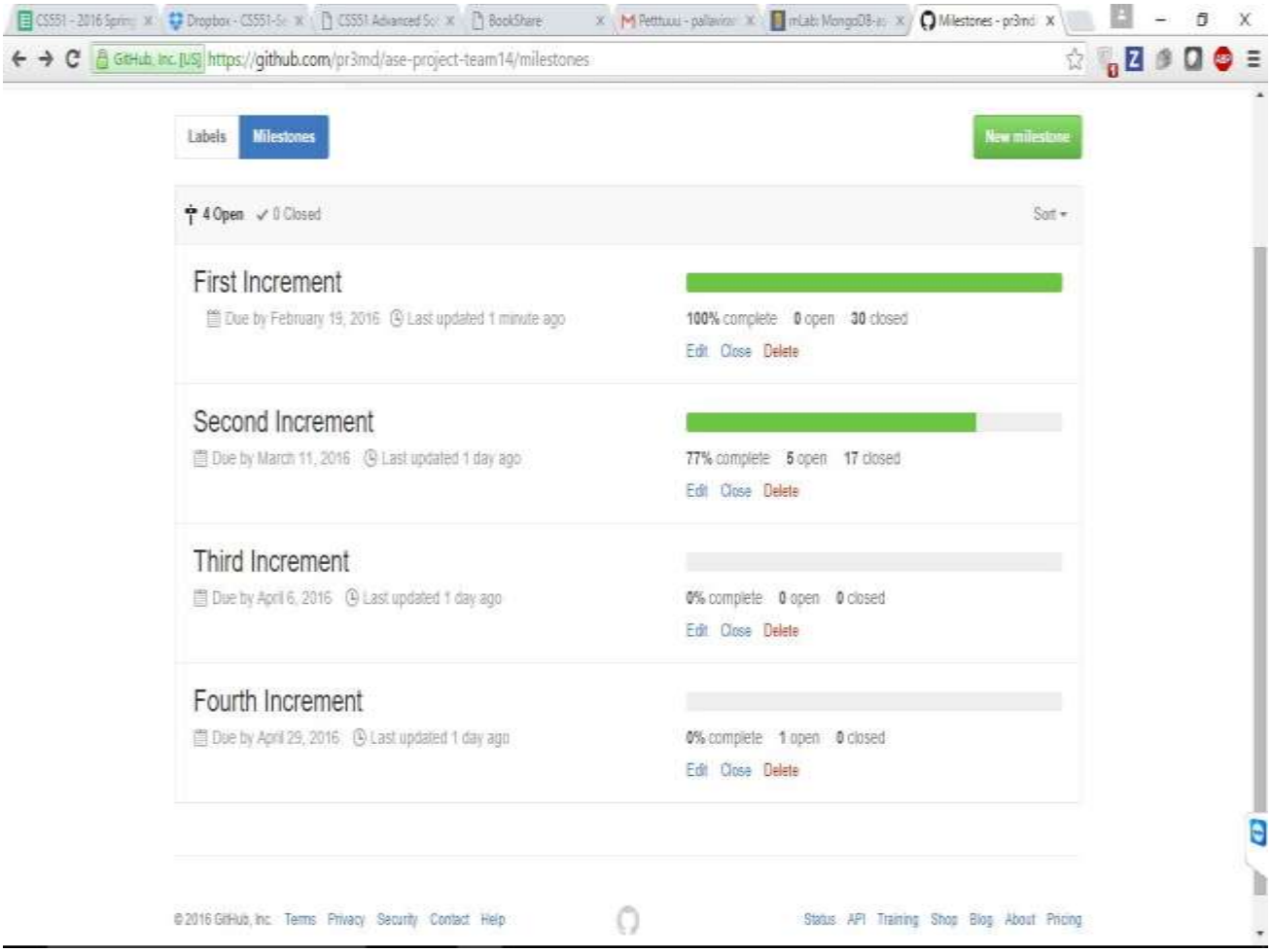
DEPLOYING WEB APPLICATION:

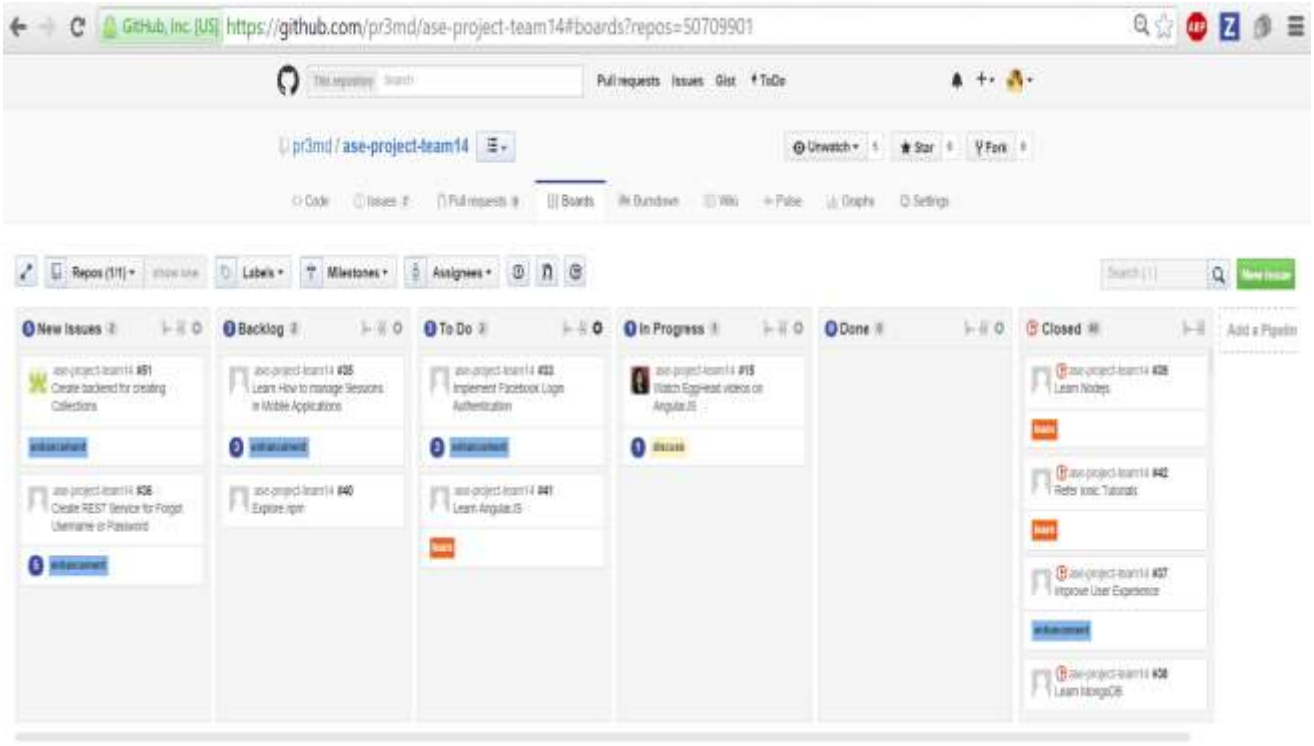


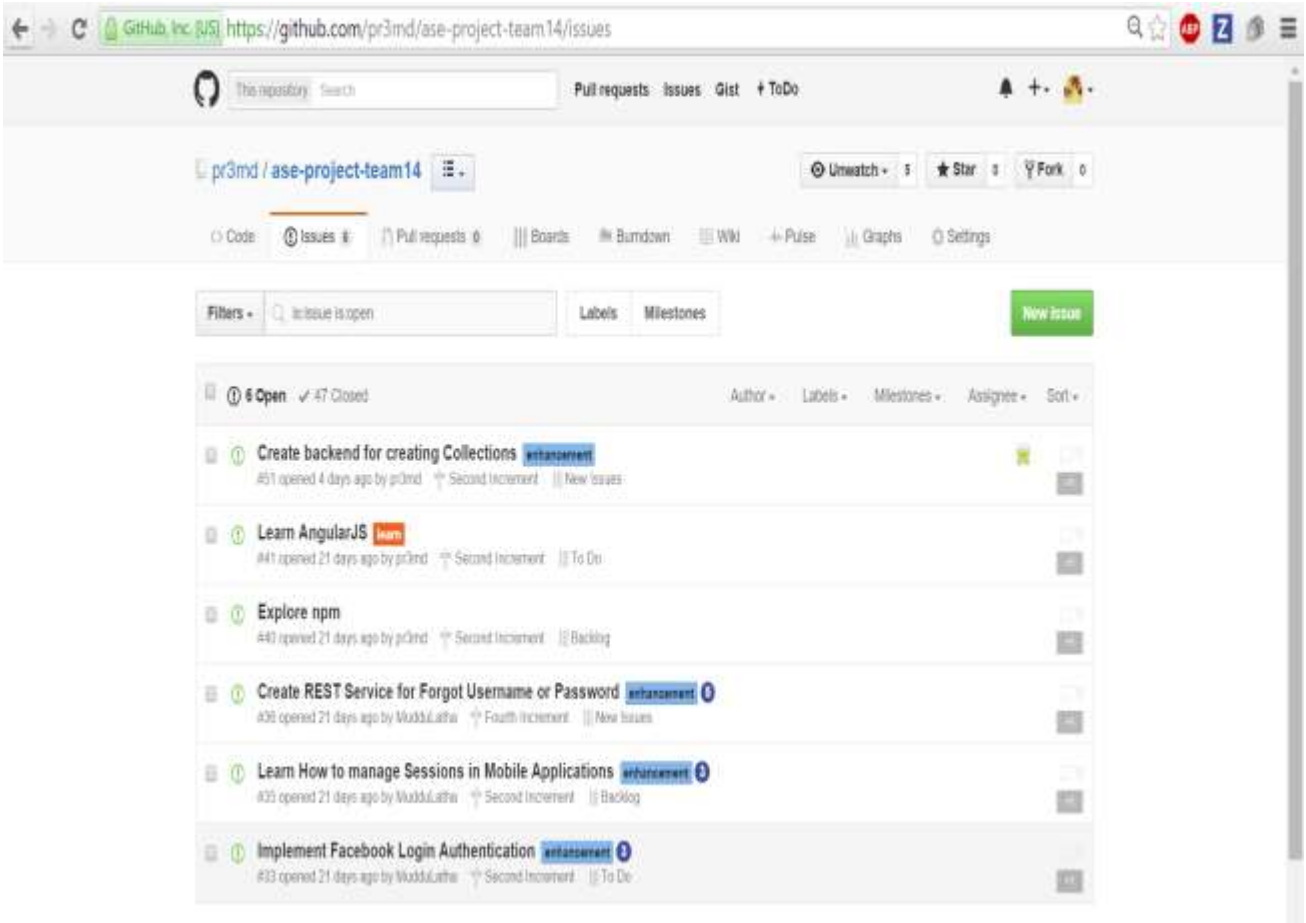


PROJECT MANAGEMENT:

PROJECT TIMELINES/MEMBERS/TASK RESPONSIBILITY:




https://github.com/pr3md/ase-project-team14#boards?repos=50709901
Pull requests
Issues
Gist
+ToDo
pr3md / ase-project-team14
Unwatch
Star
Fork
Code
Issues
Pull requests
Boards
Burndown
Wiki
Pulse
Graphs
Settings
Repos (1/1)
Labels
Milestones
Assignees
New Issue
New Issues
Backlog
To Do
In Progress
Done
Closed
Add a Pipeline


https://github.com/pr3md/ase-project-team14/issues
This repository
Search
Pull requests
Issues
Gist
+ToDo
pr3md / ase-project-team14
Unwatch
5
Star
0
Fork
0
Code
Issues
Pull requests
Boards
Burndown
Wiki
Pulse
Graphs
Settings
Filters
is:issue is:open
Labels
Milestones
New issue
6 Open
47 Closed
Author
Labels
Milestones
Assignee
Sort
Create backend for creating Collections
enhancement
#51 opened 4 days ago by pr3md
Second Increment
New Issues
Learn AngularJS
learn
#41 opened 21 days ago by pr3md
Second Increment
To Do
Explore npm
#40 opened 21 days ago by pr3md
Second Increment
Backlog
Create REST Service for Forgot Username or Password
enhancement
#36 opened 21 days ago by MudduLatha
Fourth Increment
New Issues
Learn How to manage Sessions in Mobile Applications
enhancement
#35 opened 21 days ago by MudduLatha
Second Increment
Backlog
Implement Facebook Login Authentication
enhancement
#33 opened 21 days ago by MudduLatha
Second Increment
To Do

WORK COMPLETED:

Developed RESTful services using Nodejs for handling registration, login and adding new books. Application communicates with the backend using **$http** of AngularJS. Backend communicates with mlab to fetch the data requested by the application and forwards the response back to application. The application is developed using Ionic Framework.

STORIES:

In the first increment, the application interface development has started using Ionic Framework. Basic functionality like User Login, Registration, Forgot Password, Dashboard, Book Details, Search Filter and Account Settings are developed.

Service Design: The base design of this application concentrates on User Interface and Application architecture. All the user screens are designed and backend functionality will be built in further increments.

Service Implementation: At this stage, fake data is being used in **services.js** file to fill the user interface, once backend services are established data will be retrieved from the Database.

WORK TO BE COMPLETED:

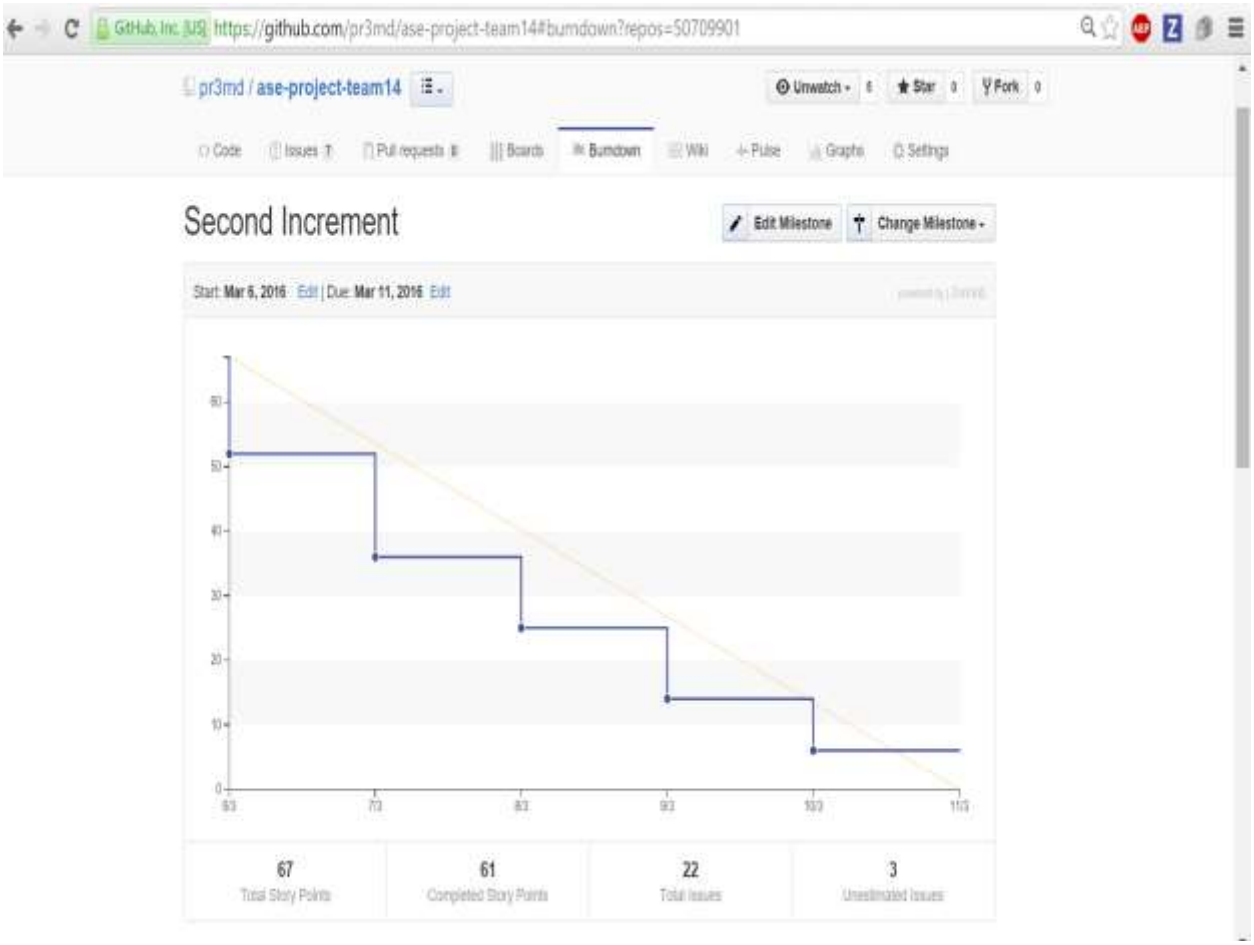
In the next increment, the following features will be added:

1. List of books added by the user.
2. Uploading image for Books.
3. User Profiles.
4. Wishlist implementation.
5. Deployment of Nodejs backend to our own servers.
6. Secure transmission of data over HTTPS.
7. Modification of Account Details.
8. Browsing books by Categories.
9. User session management.

More details about third increment are added here:

https://github.com/pr3md/ase-project-team14/milestones/Third%20Increment#boards?repos=50709901

BURNDOWN CHART:



GITHUB URL: https://github.com/pr3md/ase-project-team14.git

BIBLIOGRAPHY:

Wireframes design: http://draw.io.com/

Android studio: http://developer.android.com/sdk/index.html

ZenHub – Agile project management tool for GitHub: https://www.zenhub.io