Wolf-Share: Book Sharing Application Phase II

Abhinandan Deshpande[1], Deepak Ravindra Patil[2], Kanchan Bisht[3] and Prashant Nagdeve[4]

***Abstract*— Have you ever felt disheartened when you went to the library to check out a book, but due to its limited number of available copies were not granted the request? There are certain books in the library for which the demand is more and the copies are less to meet these demands. Since getting access to these books is rare, so there are chances that once a student gets access to one of these, he may wish to retain it for a longer time (by renewing it the allowed number of times). This would hinder the chances of others trying to get access to that book. But, during the span that the student keeps a book, he may not use it every day. There are days when he may have assignments, midterms, and quizzes due in other subjects. So, he would end up not using this book. This application seeks to exploit this period to help other students gain access to such books that are checked-out but are not being used by the borrower. But, wouldn't you be skeptical to lend a book to another which is issued using your credentials? We have introduced feature of penalty for defaulters of book. Due to this feature, if a borrower returns the book after speculated time or misplaces the book or returns it back in bad condition, user will have to pay charges to the lender. The transaction will take place using PayPal. We have integrated Paypal payment service for this feature to work. Also, we have integrated the feature of geolocation, which allows the borrower to search available books as per their spatial convenience. Apart from this, now lender can also provide information about condition of book user is lending, whether it new, old or torn etc.**

I. INTRODUCTION

Unavailability of books in the library is a common issue for all the students. Along with that, professionals who are seeking reference form other books also don't have access to all the books at all time. Hence, the previous team figured out that there is a great need of an application that will allow the users to share their books temporarily. In order to facilitate this, the previous group had created a web application that lets users to list their books for issuing. On the other hand, the borrowers get to choose from this listing and borrow it for a particular amount of time. This application served its purpose of becoming a platform for listing of books and then contacting the lender. But there were some major functionality flaws in the application.

One of these flaws is that, what happens if the borrower doesn't return the books? Or if the book is damaged? What is the guarantee that the borrower won't run away with the lender's book? In order to address these issues, we have integrated the paypal service. When the user signs up on the wolfshare application, we ask them to signup on paypal as well. If the account is verified, the user is authenticated and allowed to borrow books. Hence, in case of damage, we charge the borrower's paypal account. This gives the lender, the guarantee that even if the borrower doesn't return the book, at least they will get the money back.

Another issue with the existing application was that it only provides the listing and contact information of the lenders but does not have a proper protocol for returning the book. Hence, we created a book returning protocol within the application itself. This protocol serves multiple purposes. Firstly, it addresses any issues that arise while returning the book. And secondly, it tracks the ownership of the book and returns it back to the lender after returning is done on the application. The exact implementation and motivation for this feature will be explained later in the report.

Along with these two major functionality corrections, we had to fix a lot of files before starting off. The database files that we cloned from the github repository of the previous group was not compatible with the MySql version they recommended. We had a hard time setting up the entire project, because the steps they have provided in the Readme file are not enough. We had to meet with them multiple times to get things working in the first place. Later realized that basic functionalities like adding a book for listing was not working. It only required for us to tweak at few places, but took a lot of time for us to debug the code. All such issues with the working code set us back by one week. We could only start development after the original system was working.

After getting the application up, we figured that in order to realize the full potential of the application, there has to be a lot of books on the portal, which was impossible if the listing only consisted of the books owned by the people. Hence, we thought why not have the books issued from the library open for borrowing. This was a major challenge since, nobody will be willing to lend a book issued from the library as it may make them payable for the mistakes of the borrower. We asked the library personnel if they will be willing to change the temporary ownership of the book to the borrower. So, then the late fees will be charged from the borrower's account. But, after a lot of discussion we settled with the fact that it would be a lot of efforts from the library's side for a class project. Hence, we did not implement it, but it certainly is an option for the future work.

So, from the choice of so many possible modifications, which ones to choose and why? In order to figure it out, we conducted a small survey before starting off with any kind

*This work is a part of Software Engineering project.

[1]Abhinandan Deshpande is a MCS student in North Carolina State University, Raleigh, North Carolina. Email: aadeshp2@ncsu.edu

[2]Deepak Ravindra Patil is a MCS student in North Carolina State University, Raleigh, North Carolina.Email: dpatil@ncsu.edu

[3]Kanchan Bisht is a MCS student in North Carolina State University, Raleigh, North Carolina. Email: kbisht@ncsu.edu

[4]Prashant Nagdeve is a MCS student in North Carolina State University, Raleigh, North Carolina. Email: psnagdev@ncsu.edu

of modification.

II. CURRENT SYSTEM

A. Technology Stack:

For Wolf-Share application, current technology stack includes PHP at backend, MySQL as database and AngularJS, JQuery and JavaScript at the front end. There are many advantages of using this technology stack, some of them are as follows:

- PHP is similar to C/C++, and programmers familiar with these languages are able to manipulate and learn it quickly. PHP5 is object oriented which helps us build large and complex applications.
- PHP embeds well in HTML and helps create dynamic websites. It runs on most of servers and it also works well on cross-platforms.
- PHP along with MySQL are excellent choices as, they can be interfaced with each other without any effort.
- Apart from easy interfacing with PHP, MySQL offers unmatched scalability to facilitate management of deeply embedded applications. It also features a distinct storage-engine framework that allows system admin to configure MySQL server for a flawless performance.
- Usage of AngularJS saves time as it binds MVC together automatically. Also, it uses HTML as a declarative language. It does not require any plugins or framework for development.

All these features allowed previous application to be built quickly and effectively.

B. System Overview

In the current application developed by our peers, they provided 3 different options for exchanging books. These are as follows:

- Borrow the book by directly going to lender's house.
- Get booked shipped from borrower's house to lender's house.
- Set up a meeting at some location which is convenient for both the parties involved in exchange (i.e. borrower and lender).

In these methods, lenders are given option to choose between exchange methods. When lender wants to enroll/add book into system, it provides with the choice of options of book exchange out of given options which are mentioned above.

When searching for a book in the system, information about exchange method will also be available for borrower. Depending upon the borrower's choice or convenience he/she may able to select out of different options available for same book according to delivery method. After selection of method for the desired book, borrower will receive information about the lender. After that, they can exchange information about the meeting place via email.

Current application scope does not include any monetary transaction. System is not integrated with any payment gateway and nor does it provide any functionality to handle financial frauds. The application is not responsible for any financial security. Due to this reason, concept of penalty for unwanted or unethical conduct is not applied/implemented. This may include situations where borrower does not return book on time or never returns it back, or returns book in deteriorated condition, or borrower's behavior is not good. There might be a situation where lender showed misconduct or provided wrong information about self or the book. For all these cases owners implemented overall user rating, so that borrower as well as lender will always remain informed about any misconducts occurred previously. User rating involves both borrow rating as well as lending rating. When any misconduct happens, system reduces the rating of the user who is responsible for misconduct from his previous rating. Here, instead of charging money, user rating is deducted as a penalty. Thus, while borrowing book, borrower can have a look at rating of lender before borrowing a book and decide whether lender has good rating and is not penalized for misconduct. Similarly, lender can also take a look at borrowers rating to make sure whether his/her book will be placed in safe hands or not. This feature will force users (lenders or borrowers) to keep their rating high in order to utilize the system to full extent.

The application has two parties- lender and borrower. One user can act as both lender and borrower depending on the situation. User can become lender when he uploads one or more book for lending, he remains lender until the book remains lent. Apart from this time, for rest of the time, user remains as a borrower by default.

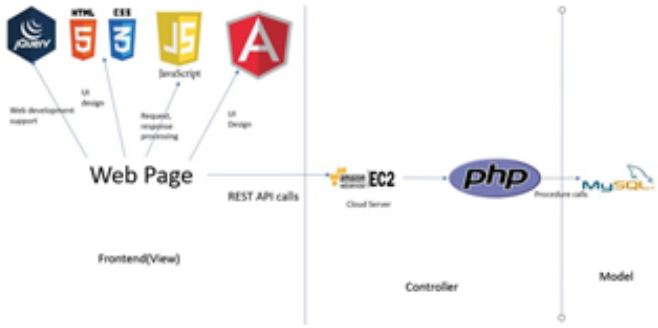

Fig. 1. System Overview

As we can see in Fig 1, the application makes use of AngularJS, which helps build MVC framework automatically. View part of application contains AngularJS, HTML, CSS, JavaScript and jQuery. View receives input from user, then converts it into JSON and sends request using REST API via POST method. The response received from the server is in JSON format and it is then parsed to show desired result. Controller part of system is built using PHP. Controller parses JSON and converts it into structured format which is supported by MySQL. User responses, which are available by querying database are converted to JSON and sent to client as HTTP response. MySQL acts a model for system. It stores required entities and procedures.

C. Application Flow

1) Landing Page, Login and Signup:: Currently, the application only accepts email addresses with ncsu.edu domain names. As we can see in Fig 2, the landing page also gives the facility to search for books even if the user is not registered.

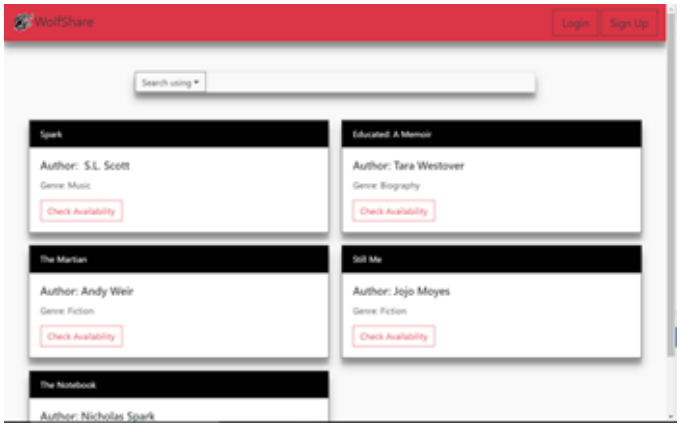

Fig. 2. Pre-login landing page

However, an unregistered user cannot check the availability of a book. When he/she clicks it and is not logged-in a signup/login prompt pops up. A verification code is sent to the email id which needs to be entered by the user while signing up to ensure security. We can see the sign-up and verification pages in Fig 3.

Fig. 3. Sign-up and verification pages

2) Post-login landing page:: The main home page that the user sees after logging in can be seen in Fig 4. It contains a navigation bar on the top which has 5 main tabs:

- Search Tab: This tab redirects the user to the search bar for searching books.
- Add a book Tab: This tab gives the user the ability to add a book for lending.
- Lent Books Tab: This tab is used to display all the books which are currently lent by the user. It also displays the due dates of each book.
- Borrowed Books Tab: This tab is used to display all the books which are borrowed by the user. It also displays the due date of each book.
- Notifications Tab: This tab provides the user with notifications for 3 cases: when a book he has lent is due, when a book he has borrowed is due and when the borrow request has been accepted/denied by the lender.

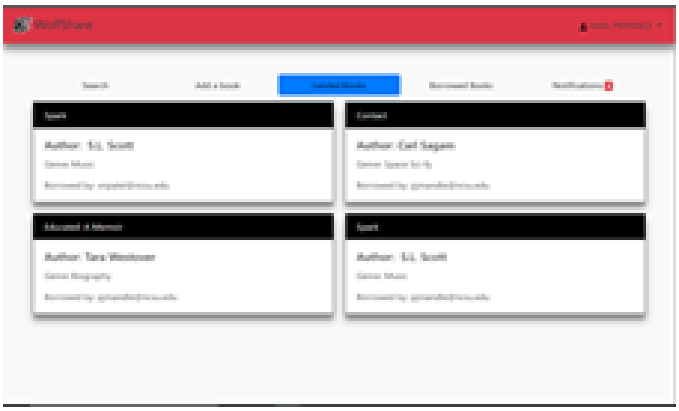

Fig. 4. Post-login landing page

3) Add a book Tab:: This tab redirects the user to a form where he/she can add a book which he/she wishes to lend out. The form contains basic book identification information such as title, author, genre, etc. It also contains a start and due date set by the lender and a mode of delivery. The mode of delivery, currently has 3 options: Meet at commonplace, shipping by lender to borrower and meet at lender's house.

4) Approval Card:: This card-based notification, as we can see in Fig 5, is displayed in the notification tab where a lender gets a request by a borrower and is given the option to approve/disprove a borrower based on the rating of the borrower. This allows a borrower to make an informed decision, whether or not to borrow a book.

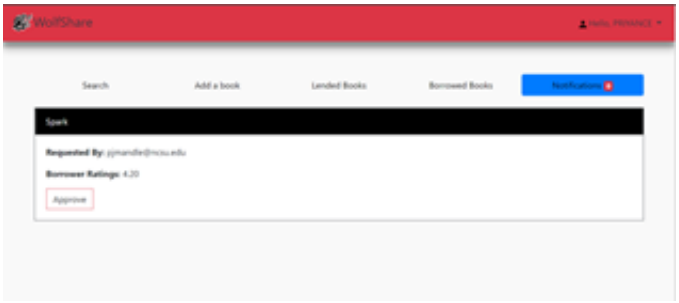

Fig. 5. Approval/Disapproval card

5) Book Card:: This card as seen in Fig 6 displays information about a book which a user has currently searched for and wishes to check its availability. It contains information in two parts. The first part on the left shows information about the book and for what period it is available for borrowing. The second part on the right shows information about the lender, for example his/her name, and ratings/review. This allows a borrower to make an informed decision, whether or not to borrow a book.

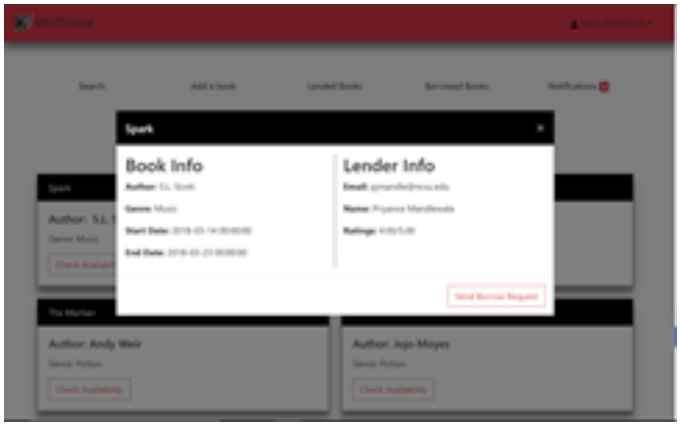

Fig. 6. Book card

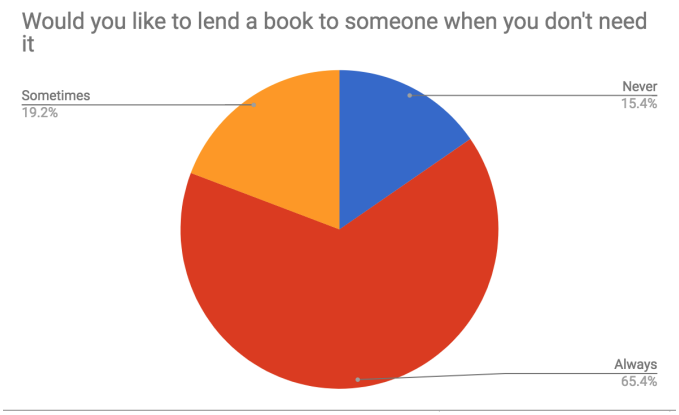

Fig. 7.

III. Initial Survey

In order to learn more about the thinking process of our target audience(university students), we have conducted a small survey to build on the already available product. Before going to the survey we conducted, let us first discuss the survey conducted by the previous group.

- Around 80% people who took the survey were facing difficulties while issuing book from the library since, it was not available.
- About 83% of the students preferred to have a secure web application that could facilitate temporary book lending/borrowing.
- Around 40% of the students were willing to lend it for free. This means that the other 60% were expecting some kind of reward for the service.
- About 84% of the students were comfortable with the rating based system of authenticity.

Let us discuss what we can take from the above found data. Firstly, the students are facing a problem and there are some solutions for this problem which are waiting to be accepted by the community. Secondly, the students did not mind lending a book temporarily. And lastly, people tend to rely on some assuring parameter that indicates the genuineness of the other user(rating of the user in this case).

And now let us discuss the new survey we organized, to see how people responded when we suggested some solutions to them for the above problem.

A. Would you like to lend a book to someone when you don't need it

With this question, we wanted to emphasize on the part when the user doesn't need that book. This will reduce the 'shelf-time' of each book. As we can see from Fig. 7, the people are not sadist in nature. They don't mind lending a book. The 35% of the students who said 'No' for lending must have their reasons. But the most important one of those reasons being, no confidence in the unknown person's ability to return the book on time. Hence we asked the user if they would mind sharing the book which they have issued from the library.

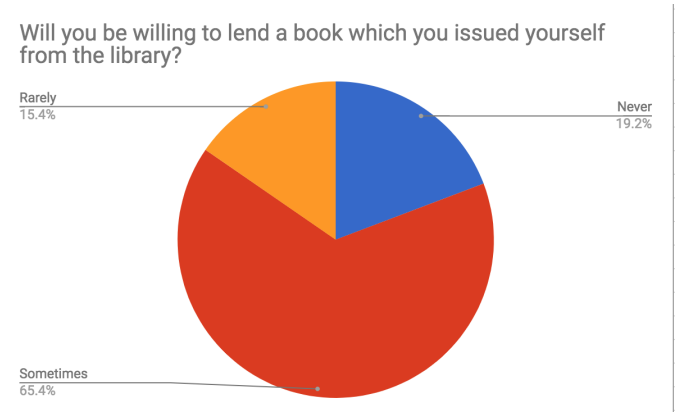

Fig. 8.

B. Will you be willing to lend a book which you issued yourself from the library?

The motive behind asking this question was, to know if people are willing to share a book which they are responsible for. These books (issued from the library) are different from the personally owned books, since the user is responsible for the issued book and would damage his history with the library if he doesn't return it. Hence, as expected, the students are reluctant to lend a book.As seen in Fig. 8, 0% people have said that they would always be available to lend a book issued from the library. Now to counter this problem we suggested them a solution and asked them if they would prefer it.

C. What if the library changes the ownership of that book temporarily (So that late fees will be charged to his account). Then will you lend the book to someone else?

This question was framed in a way that it suggests a solution to the previous problem, and asks about the acceptance possibility of the same. We can see the drastic difference in the number of people saying 'Always' for the same question in Fig.8 and Fig.9, 62% people are willing to lend the book if they are not held responsible for any damages. Now, this

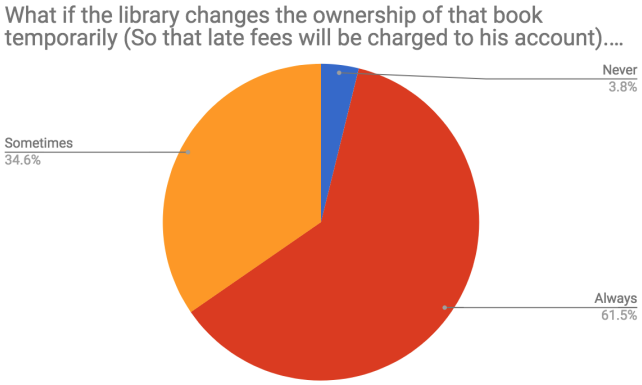

Fig. 9.

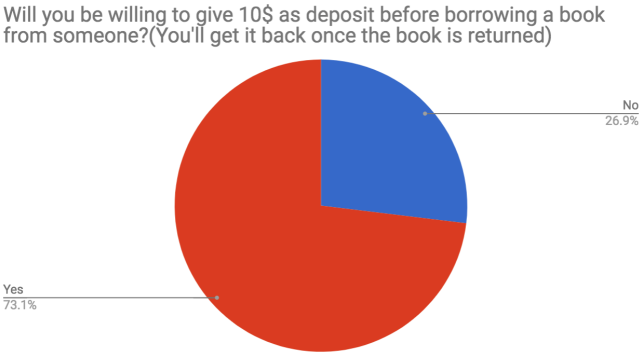

Fig. 10.

can be the basis of our hypothesis that 'People don't mind helping out each other'. Having said that, now we wanted to know, how much money the people are willing to keep as a deposit, in order to borrow a book from an unknown person.

D. Will you be willing to give 10$ as deposit before borrowing a book from someone?(You'll get it back once the book is returned)

Even though we could see around 75% people were fine with the 10$ deposit, but that much amount might not be enough for some cases where the books are expensive. But the Fig.10, demonstrates people's commitment and understanding towards proving their genuineness. Hence, we think that just connecting the user's Paypal account is enough. And if any damage to the book is done, we can charge the user for that amount later.

Hence, we can conclude that adding a payment option is necessary for the users to trust this application.

IV. DEVELOPMENT CYCLE & SYSTEM OVERVIEW

A. Software Development Lifecycle

In development phase we stuck to spiral model of software development. In our previous project we used spiral model to develop prototype of functionality and then iteratively proceeded to main application. Similarly, we proceeded with spiral model this time too as members were comfortable with this development model and also it was not feasible for entire team to follow new development model. Advantage of using spiral model was that it incorporates iterative prototyping process with controlled and systematic aspects of waterfall model. And also it helps build more complete software with each iteration.

Below is the original diagram of spiral model.

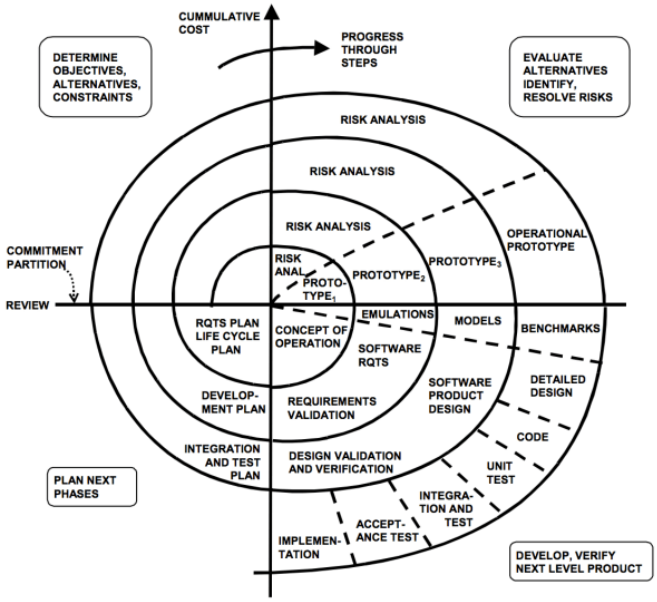

Fig. 11. The original spiral model diagram. Credit Barry Boehm, & Wilfred J. Hansen

1) Requirement Gathering: The first thing we planned to do was to get overall idea of new requirements which user may want or like. In order to achieve this we conducted a survey. Questions were designed to give more insights about users view regarding improvement. Questions were based on whether borrower and lender agree upon transfer of liability or any inclusion of monetary transaction for any misconduct that might rise during lending and borrowing activity. Apart from the conducted survey, we also talked with library personal about librarys role in our application scope. Currently, NCSU library does not allow transfer of liability between users due to various reasons, one of them is high number of hold requests that are made to library for a particular book. And libraries follow first come first serve policy for allocating books. We further made few efforts to talk with library but our efforts were futile as NCSU library does not allow any third party interference. Hence, we scrapped the plan of integrating our system with library. It is not possible in near future as well.

2) Design and Build: We did not change existing system architecture of wolfshare 1.0 nor we changed the technology stack. We integrated current system with Paypal payment service which we are going to explain in detail in upcoming

section. Apart from payment integration we added geolocation feature to system. This feature allows each and every user to search book based on locality/location. Also, we incorporated feature of initiating return request online using our application unlike previous version in which returning of book was done offline.

3) Risk Analysis: We thought that major risk would be security and privacy of user accounts and transactions. But Paypal handles these issues very well and we need not to look into issues of providing encryption and security to transactions.

B. System Architecture

The application has two parties- lender and borrower. One user can act as both lender and borrower depending on the situation. User can become lender when he uploads one or more book for lending, he remains lender until the book remains lent. Apart from this time, for rest of the time, user remains as a borrower by default.

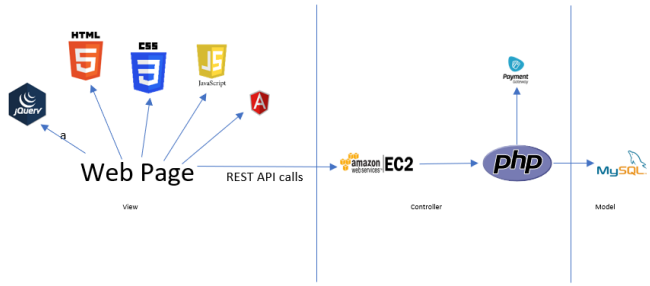

Fig. 12. System Architecture

As we can see in Fig 12, the application makes use of AngularJS, which helps build MVC framework automatically. View part of application contains AngularJS, HTML, CSS, JavaScript and jQuery. View receives input from user, then converts it into JSON and sends request using REST API via POST method. The response received from the server is in JSON format and it is then parsed to show desired result. Controller part of system is built using PHP. Controller parses JSON and converts it into structured format which is supported by MySQL. User responses, which are available by querying database are converted to JSON and sent to client as HTTP response. MySQL acts a model for system. It stores required entities and procedures.

Apart from 'model', 'view' and 'controller' there is one more important thing that needs to be emphasized is PayPal Payment Service. This service is new addition to features of new application.

We deployed our application on amazon EC2 instance so that our application will be available to all users for testing purpose.

V. CHALLENGES FACED

The technology stack which has been used in Wolfshare was familiar to us but learning the existing architecture was a challenge. While setting up the Wolfshare App in our system, we followed the instructions mentioned in the readMe file, but we were not able to set up the application successfully. As mentioned in the readme file, we installed Xampp, but we were getting some errors while using wolfshare because application does not work well with the Xampp. We have tried another option Wampp server and we successfully set up the DB on Wampp server. But unfortunately,the booksharingapplication.sql file which was provided in Github repositiry does not work. We tried to use the application with that file but we were not even able to add a book in database. With that sql file, book sharing application does not work at all. Even the URL to run the app mentioned in the readMe file is incorrect.

We contacted the wolfshare team and took help from them. The previous team provided us with new booksharingapplication sql file which has some data in all the tables of database. This file worked well as it has prepopulated data in it. As we deleted all the data from tables, book was getting added to the table but lender table was not getting populated. As the previous app was not working properly, we had to make changes to all the procedures provided with wolfshare app. Even after user made a borrow request, borrowrequest table in database was not getting populated. After going through all the php files, sql procedures in DB and architecture, we made lot of changes and rectified the errors with the previous system. We had to spend lot of time to make the previous system working.

To make our application reliable, we approached the library with the proposal of integration of wolfshare app to library database. As lot of development was required on the library side, we were not able to integrate wolfshare to NCSU library system.

Implementing a payment gateway was a challenge for us. In the first scenario, we were taking some percentage of book cost before lending the book. As Paypal costs some amount on each transaction, most of the users were hesitating to use our app. In the second scenario, we charged borrower only when there was some penalty or due amount because of faulty transaction. This worked pretty well and users were ready to use our application.

One last challenge that we faced was deploying this application to cloud. None of our team members were familiar with any cloud deployment platforms. Also we were not sure of pricing of each cloud service provider.Our mentor suggested to go with 'heroku' one of the free and good quality cloud hosting service available. Deploying application on heroku seems to be straight forward but there are many small things we need to take care of. We followed few video tutorials and documentation given on heroku website. To deploy any application on heroku we need to have git configured on our system as most of the deployment is done using git bash commands. There are various files such as Procfile, composer.json and .htaccess which are needed by php application to be deployed. And also it requires some 'vendor' files all of which we were not aware of. After all requirements were met we deployed the application on cloud,

but still it was not working correctly. We identified reasons for this. Application given to us was not cloud ready, paths mentioned were local and not global. We fixed these issues but still one issue persisted, which was not able to connect to database as application structure was not what heroku required. We did not wanted to make any changes to current application structure, hence we moved towards the Amazon EC2 Cloud.

Apart from issues related to heroku, we also had some issues with Amazon EC2 deployement. Previous team also did not provide any documentation related to how they have deployed their application and also we did not receive any help from them. Only one link was provided by them which was not explaining all the things which we needed to configure. Amazon RDS configuration was simple task but to access external RDS and application, we need to have VPC (virtual private cloud) and security groups and policies to be configured. After that part was done we followed the steps to deploy it on Amazon Linux AMI distribution as well as the Ubuntu Server 16.04 edition. The problem was same as that of heroku. Application structure was not confined to what was expected by Amazon Linux or Ubuntu. Multiple attempts were made to solve those problems but still errors were not rectified. We also tired Elastic Beanstalk to deploy our application, but we did not had enough time to resolve errors which we got after deployment. Then we tried one of the simple option we thought would be better as we were running out of time. This approach is mentioned in implementation section.

All in all, learning an architecture of application was difficult but the familiarity with technologies involved in development helped us to overcome the challenges. After a careful study about how payment integration works, we successfully implemented a payment gateway for our app which made Wolfshare App more reliable. After facing number of challenges, we successfully deployed our app on the server and now it can be accessed by anyone.

VI. IMPLEMENTATION

1) Payment Integration: **Motivation:** Once the third party borrower (2nd Borrower) acquires the book from original borrower (1st Borrower) for certain time period, he can use that book for that particular time period. But what if 2nd borrower does not return the book? Or what if he has lost the book? In this case, the original borrower has the liability of book and if he is not able to return the book to the library, he will be charged some amount of penalty based on the duration of returning the book. The user who let someone to use his books will incur a loss which is very unfair situation in real life.

Based on the survey conducted, we found out that no one was sure about using previous version of Wolfshare book app because lender was responsible for any late fees. Hence it was necessary to implement this particular feature.

Another limitation of the previous system was that it could only be used for NCSU users as application needed verified

Book Condition	Daily Fine	Grace Period	Penalty if Lost
New	1.5 $	5 Days	100 % of Actual Cost
Normal	1 $	5 Days	70 % of Actual Cost
Bad or torn	1 $	5 Days	50 % of Actual Cost

Fig. 13. Penalty Calculation Rules

Book Condition	Due Date	Actual Return Date	Lost	Penalty
New	4/1/2018	4/27/2018	No	31.5 $
Normal	3/21/2018	3/28/2018	No	5 $
Bad or torn	2/2/2018	4/9/2018	No	61
New	4/10/2018	4/20/2018	No	7.5 $
Normal	3/21/2018	N/A	Yes	20(Cost)*70%=14 $
Bad or torn	2/2/2018	N/A	Yes	34(Cost)*50%=17 $
New	4/23/2018	N/A	Yes	80(Cost)*100%=80 $

Fig. 14. Different Penalty Scenarios

users. Hence user base was limited and application was not open to any users outside NCSU community.

Solution: To address above problems, we have implemented payment gateway using Paypal Integration. When users signs up with our application, he will be asked to fill the details of his/her Paylpal account.We are maintaining the condition of the book. Condition of the book can be New, Normal or Bad. Based on these 3 conditions user will be charged some percentage amount if he misses the deadline. Charge amount varies for different conditions. When any borrower misses the deadline, due amount is calculated based on the following scenarios. This amount is deducted from Paypal account linked with the borrowers account. Fig 13 and Fig 14 shows how penalty is calculated based on the condition of book.

2) Geolocation: **Motivation:** In the previous system, the book searching criteria was based on title, genre and author. There might be the possibility that user searches for a book and requests it but lender cannot give the book because of geographic limitations. This results in wasting time of borrower as he was not able to receive the book. Hence it would be nice if we are able to search only those books which can be delivered to us.

Solution: In this system, we have implanted a Geolocation based search feature which allows users to search for a book using a location. Each book is tagged with a location and saved in a database. Whenever user searches for a book, result will be displayed based on users selected location. Implementing a Geolocation functionality helped user to avoid false request and he could utilize his time in making request to only those borrowers who can deliver books. Borrower also can enter the specific area and can search for users in specific areas. Fig. 15 shows that when user types “ra” in the search field, books whose location start with “ra” are displayed in the search result.

3) Book Return Protocol: **Motivation:** In the previous system, user could borrow a book online but there was no procedure to return the book. Returning book was done offline. As there was no way to communicate with each other, borrowers and lenders were facing issues while returning the book. Lender himself was making a note that this particular

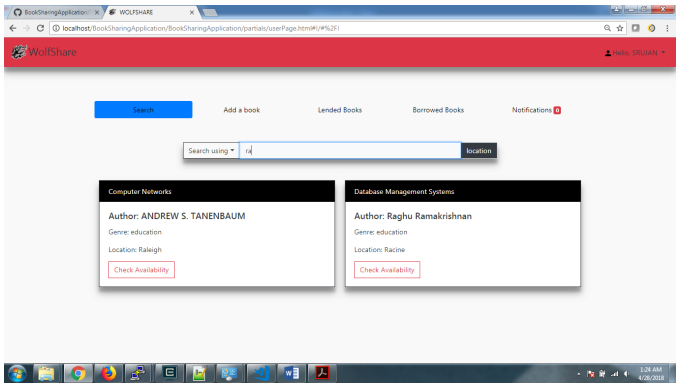

Fig. 15. Results when user searches for location that starts with "ra"

bookId	borrower_email	lender_email	return_status
99	pjmandle@ncsu.edu	sjbarai@ncsu.edu	0
98	sjbarai@ncsu.edu	pjmandle@ncsu.edu	1

Fig. 16. Snapshot of return request table

borrower has returned the book. Previous System did not consider any book return request, it only dealt with book lending.

Solution: In the current System, we have added a book returning protocol which helps user in returning a book. All the book returning transactions are stored in returnrequest table. When any user requests a return, an entry is added in returntable with status 0. Status 0 means return is in progress. When return is completed , returnstatus becomes 1. Either a borrower or a lender can request a return. Fig. 17 shows us how book return transaction is stored in a database.

Cases of returning book: Either lender or borrower can initiate return request. Following are the two cases of return request.

- **Case 1: Borrower requests for a return-figure 17**

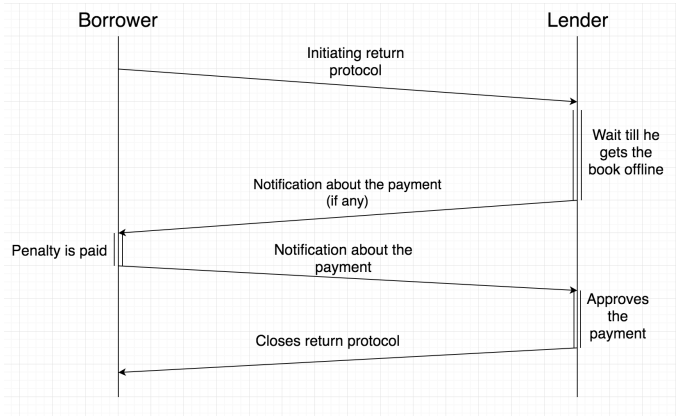

Fig. 17. Case 1:Borrower requests return

  - **Step 1:** Borrower requests for a return.
  - **Step 2:** Lender is notified that the particular borrower wants to return a book.
  - **Step 3:** Lender waits till he gets the book offline. After getting a book, lender checks the condition of a book and a due date, if book is damaged and not in the original condition or return date is missed, due amount is calculated based on the chart mentioned in the payment section.
  - **Step 4:** Borrower is notified that payment of his transaction is pending. If there is a due amount, borrower pays the penalty.
  - **Step 5:** Lender is notified that borrower has paid the required due amount and lender approves the return request.
  - **Step 6:** Borrower gets a notification that lender has approved his return request.
- **Case 2: Lender requests for a return**
  - **Step 1:** Lender requests for a return.
  - **Step 2:** Borrower is notified that the particular lender wants to get the book he/she lent you. Borrower approves a request and return the book offline
  - **Step 3:** Lender waits till he gets the book offline. After getting a book, lender checks the condition of a book and a due date, if book is damaged and not in the original condition or return date is missed, due amount is calculated based on the chart mentioned in the payment section.
  - **Step 4:** Borrower is notified that payment of his transaction is pending. If there is a due amount, borrower pays the penalty.
  - **Step 5:** Lender is notified that borrower has paid the required due amount and lender approves the return request.
  - **Step 6:** Borrower gets a notification that lender has approved his return request.

A. Modification in the previous System

1) Improved Notifications: **Existing Notifications Fix:** Existing notifications were not working properly in the previous system. We have modified existing procedures in the database to get the correct notifications.

Notification for approved book: In the previous system, there was no notification for borrower if lender has approved his book request. There was no way for him to know about his request status. In the current system, borrower is notified when lender approves his/her return.

Lender Info Fix: Lender info for a book was not working as lender table in a database was not getting populated.Figure 18 shows that lender info showed nothing in the previous system for new lenders. We have fixed the procedures to add new lenders to lender table in database. Now lender notifications are displayed correctly. As you can see in figure 19 that lender info is now displayed properly.

2) Improvements in Borrower functionality: **Fix 1:**In the previous system, user could borrow a book from himself.

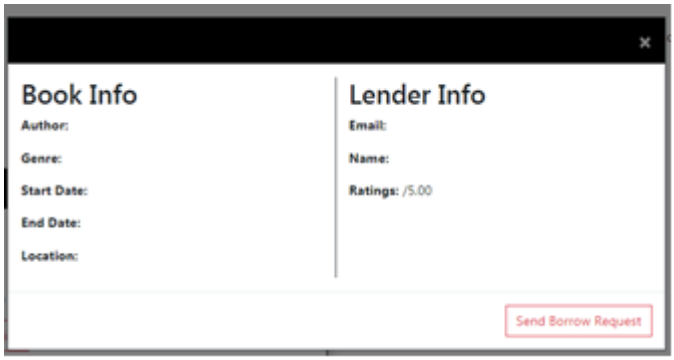

Fig. 18. Lender Info in previous System

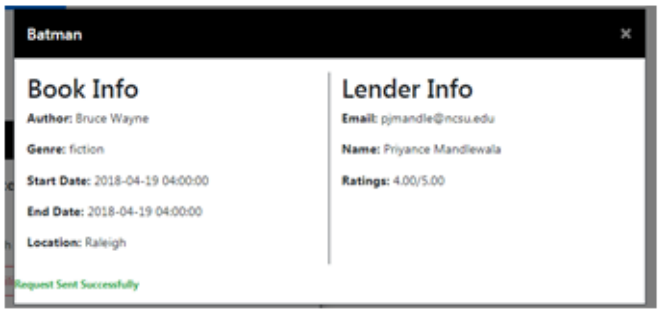

Fig. 19. Lender info in current System

This did not make any sense as user cannot borrow a book from himself. Hence we have modified the borrow book function to tackle the above mentioned problem. Before allowing a borrow request, we only allow borrower request to only those book whose owner is different from the current borrower.

Fix 2: Previous system allowed a multiple borrow request from a same user for a single book even though he/she has requested for that particular book. In our system, we only allow one request for a single book. User cannot make multiple request for a book twice if his previous request has not been processed.

Fix 3: In the previous system, User could request for a book even if it is currently borrowed by some other user. In our system, we have maintained an availability of a book with it. Figure 20 shows a snapshot of book table. In book table in database, we save the availability of book. If it is 0, that means book is not available and hence you cannot request it. If availability is 1 that means any user can borrow a book.

3) Improvement in Search:: In the previous system, user could see all the books including books from himself/herself. In the current system, user can now see only those books who are not added by him/her. This help user in searching relevant books. Location based search is added in the current system to facilitate the search of a book using location.

methodOfDelivery	book_status	location	availability
HD	BAD	Racine	1
PK	NEW	Raleigh	1
PK	NEW	Raleigh	0
HD	BAD	Racine	1

Fig. 20. Snapshot of book table in DB

B. Cloud Deployment

As mentioned in challenges faced section, our efforts failed multiple times. We tried various ways but were not be able to succeed. If we would have had more time, we might be able to fix those issues with help of some experienced people.

For EC2 deployment, we first created EC2 instance which was allowed in free tier services. We created Microsoft Windows Server 2012 R2 instance. Once all required configurations were done we created key-pair to connect to our instance securely. We connected to newly launched instance using Remote Desktop and key-pair we created earlier. Using decrypted key we logged in to remote desktop (our instance). We then downloaded google chrome, WAMP server with php 7.2.4 and mysql 5.7.21. All these application were installed and we followed all mentioned steps which we mentioned on our project's github readme.md page. It includes, installing WAMP server, placing application folder into www folder, starting apache and mysql server. Then creating and importing database. After all these steps our application was ready to be used by public networks. Our main concern while using WAMP was scalability and security of application. As for now, we wanted our application to be tested by only few users and number of users accessing application were not too huge we were getting desired results within speculated time. We are planning to deploy our application on linux distribution and will make it production ready in future.

VII. EVALUATION

For evaluation purposes, we asked the students to use both the old as well as the new version of the WolfShare application. We took a note of the time taken by the users in trying to access the application, trying to find a book, requesting for it and after request approval, returning it back. All these observations helped us evaluate our application overall. After all, who can better tell about the usability and efficiency of an application than a user himself!

First of all, we asked the users to access the old version of the application. For that the instructions were given in the old group's Readme.md file. Also, all of us were available to guide the evaluators. We summarized the steps involved

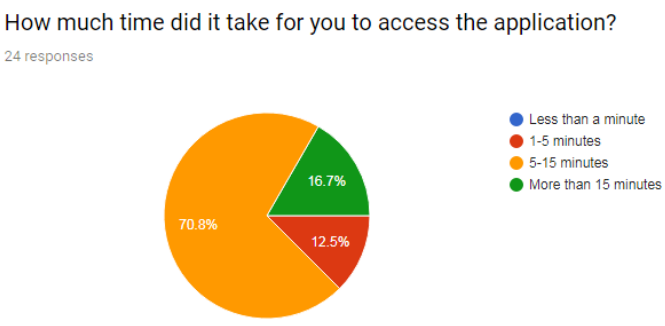

Fig. 21. Old version Question 1

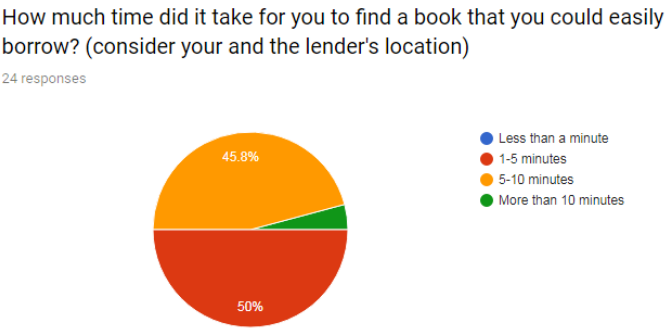

Fig. 22. Old version Question 2

and asked the users to set up the system in order to access it. As soon as the users started this task, we timed them. On finishing, we asked them to fill the first question of the survey.

After this, we asked the user to sign up, log in and then find a book. We again timed them for this operation. The next figure shows the results obtained.

Since the users had gone through the book searching process, we asked them if the searching process could be enhanced by addition of a location filter. The next figure shows the results obtained.

After the users had borrowed a book, we asked them the following:

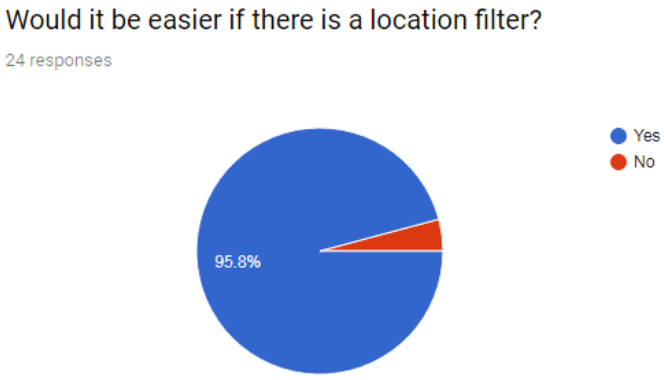

Fig. 23. Old version Question 3

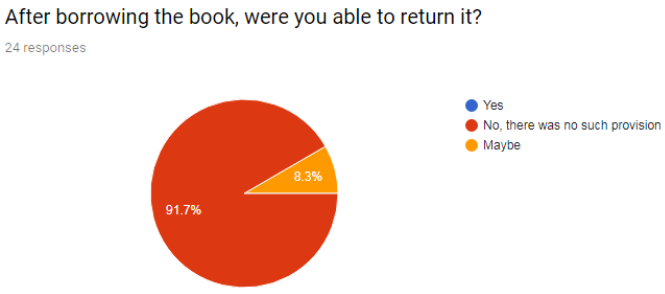

Fig. 24. Old version Question 4

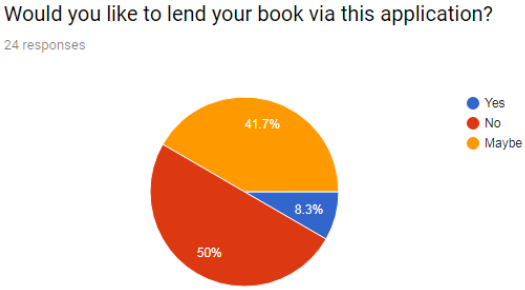

Fig. 25. Old version Question 5

In the end, we asked the users if they would like to lend their books via this application. We can see that only 8.3% users agreed.

We also had an option question in the survey for the old system. Only 37.5% users responded to the optional question. Their responses are as follows.

After this, we asked the user to access the new system. For this we instructed them to use the link to the deployed application. We again timed them for this task. As we can see in the results displayed below, 66.7% users were able to access our application within a minute.

We then asked the users to sign up, login and then find a book. We again timed them for this operation. The next figure shows the results obtained.

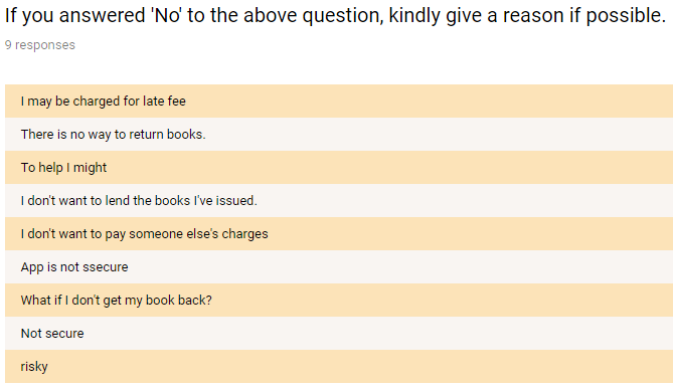

Fig. 26. Old version Question 6

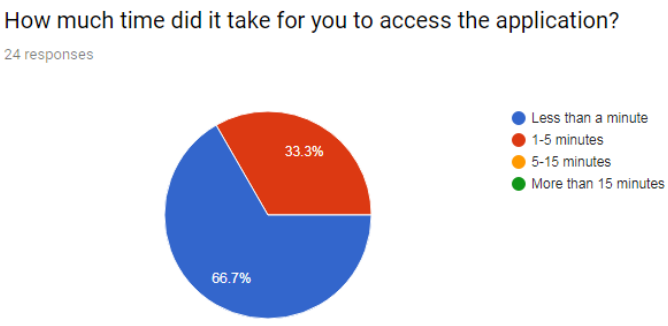

Fig. 27. New version Question 1

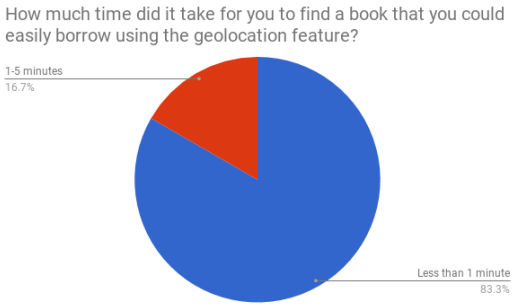

Fig. 28. New version Question 2

We asked the users to figure out how to return the book. The following questions demonstrate the results.

Once they had it figured, we asked them to return the book.

We finally asked the users what option they would prefer if they wanted some book urgently that was not available in the library. The following pie chart demonstrates the results.

1) Evaluation Conclusion: By the use of timer and by making the users perform the same tasks via the old and new versions of WolfShare, we were able to analyze the results and come to the following conclusions.

Our deployed application could be accessed on a single click within a minute most of the times. This increased it's

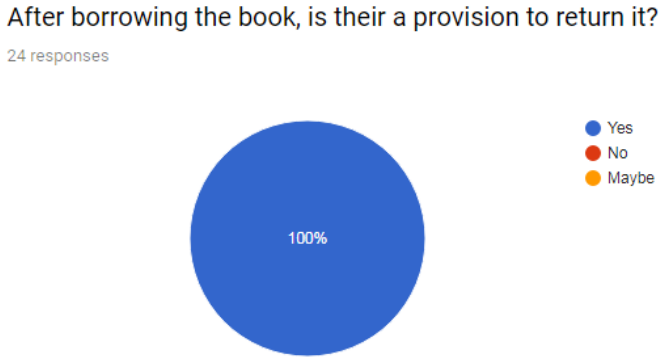

Fig. 29. New version Question 3

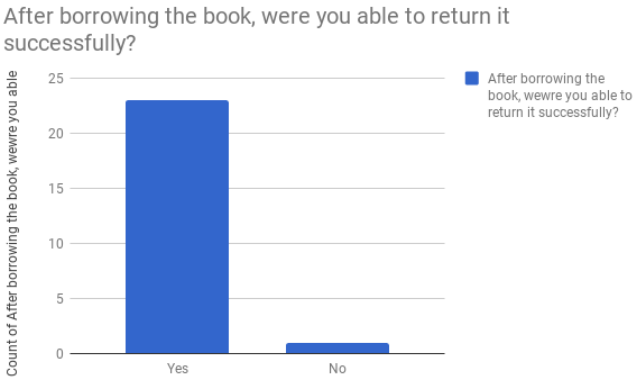

Fig. 30. New version Question 4

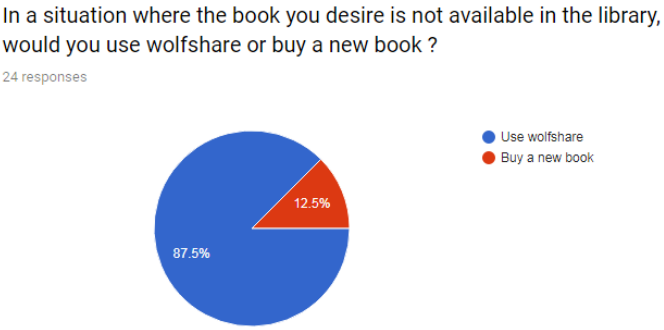

Fig. 31. New version Question 5

accessibility by a huge extent as now the users didn't have to go through the tedious local manual setup prior to accessing the application.

The addition of geolocation feature enabled users to search for their required book with respect to spatial preferences easily. This enhanced the application's usability by making it more convenient and faster.

We carefully examined the optional comments of the users in the survey. We concluded that the old system was not reliable as it did not handle the charges that may be incurred due to mishandling of borrowed book. Imagine you are helping someone by lending them your book and are paying for the damage that they have caused too! That wouldn't be nice, right! Therefore, we added features to enhance accountability of our application.The addition of book returning protocol and payment integration via Paypal for payment of charges in case of an fee on the book, damage to the book etc, made our application more reliable and 87.5% users positively responded to making use of this application in need.

VIII. CONCLUSIONS

Based on conducted user survey and received feedback, we tried to implement all the necessary functionalities which were suggested by the users. In previous system, there was no

strong measure to prevent users from any kind of misconduct such as bad handling of book or miss-placing/losing of borrowed book, damaging of book. After implementing the payment functionality, we hope to prevent this misconduct and provide better usability.

IX. FUTURE SCOPE

A. Faster Mail API Integration

Though we have improved and added new features into the existing application, there is need of improving one basic and important feature which helps in user registration, that is e-mail. Current system is using PHP mailer which was already implemented in wolfshare 1.0. But this mailer does not perform well. It takes approximately 45-60 mins to send verification code to user. We will try to incorporate new mail API which will be able send verification mail within few seconds, because no one waits for this huge amount of time just for registration.

B. Chatbox

Once user (borrower) places borrow request, and it is approved by lender, lender's and borrower's e-mail are shared with each other so that they can discuss about place of exchange and any other matter related to exchange using e-mail communication. This mode of communication does not encourage use of application more and more, rather it makes both users to go away from application for communication. To overcome this issue we can implement 'chatbox' functionality. With chatbox available for communication, users will not need to go away from application, login into mail account and communicate using e-mails. With this feature, users can discuss exchange point, time and other related details with ease. As multiple chat applications are used by lots of people and are preferred by users this feature will surely boost liking towards our application. Also, it will encourage users more and more to use our application platform.

C. Book Recommendation

Most of the times, students struggle with textbooks required for their curriculum as they dont know which book in particular they should refer to, for their subjects. Students navigate on web and find something which may not be good. If user wants to read books related to specific topic, he has to search for best books before he requests for any book on our application. In future we can provide a feature where user will get a suggestion about which book he should read for particular topic and subjects. User also can request other users to recommend books. This will result in more active involvement of users. As a user would know that he will easily find which book he will need for a particular topic, users will use our system very often.

D. Android Application

Our wolfshare 2.0 as well as wolfshare 1.0 are web applications. Nowadays, current generation prefers mobile applications whether android or iOS. It is also convenient for users to access mobile application rather than web application. Also, people will prefer using chatbox on mobile app rather than web application. Also, mobile application will make our user interface more interactive. Mobile application will allow users to access our platform on the go.

E. Review Mining

In current as well as previous version of application ratings are provided using stars range from 0 to 5, 0 being lowest to 5 being the highest. By using this method, we can calculate overall user rating by just simple math and provide overall rating of each user. But this method does not provide any insight about that rating, meaning what was the reason behind low or high rating. Application currently provides a simple textbox which allows users to record their experience and reasons behind the rating, also application related comments. Information provided by each and every user is stored in database but not currently used to draw any kind of conclusion. In future, we can make use of this information to perform review mining by making use of natural language processing. By making use of various methods such as segmentation, normalization, language identification, classification and relationship extraction we can generate sentiment insight from the reviews which will help us understand reviews given by every user to every other user and also provide much better information about each user and transaction done by that user (either lending or borrowing).

ACKNOWLEDGMENT

We express our sincere gratitude towards our Software Engineering professor, Dr. Timothy Menzies, for giving us the opportunity to freely change the assigned project, implement and test it using the tools and technologies of our choice. We also thank our Teaching assistant Amritanshu Agrawal for his valuable guidance and our fellow peers for helping us in requirement gathering by participating in the survey made by us.

REFERENCES

[1] PHP: https://github.com/php
[2] AngularJS: https://github.com/angular
[3] HTML5: https://www.w3.org/html/
[4] PHPMailer:https://github.com/PHPMailer/PHPMailer
[5] CSS3: https://www.w3.org/Style/CSS/
[6] JS:developer.mozilla.org/enUS/docs/Web/JavaScript
[7] MySQL: https://github.com/mysql
[8] Software Engineering: A Practitioners Approach 8th Edition, Roger S. Pressman, Ph.D., Bruce R. Maxim, Ph.D.
[9] https://www.lib.ncsu.edu/borrow/fines
[10] https://github.com/DexterousMe/BookSharingApplication/blob/master/Report/MarchRe
[11] https://github.com/VishrutPatel/BookSharingApplication

X. CHITS

- OPY
- MAY
- DJD
- WOK
- SCI
- DMO
- MLY

- CRX
- MII
- ZYU
- JWU
- ZYX
- CMN
- GNM
- ZZK
- WIM
- BKP
- PNZ
- FJP
- VXL
- PZF
- YCL
- WJS
- QXL