Wolf-Share: Book Sharing Application Phase II

Abhinandan Deshpande[1], Deepak 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 books after speculated time or misplaces the book or return it back in bad condition, user will have to pay charges to lender. The transaction will take place using PayPal. We integrated Paypal payment service for this feature to work. Also, we integrated feature of geolocation, which allows borrower to search available books from their spatial convenience. Apart from this, now lender can also provide information about condition of book user is lending, whether it new, old or torned etc.**

I. INTRODUCTION

YET TO COMPLETE

II. PREVIOUS SYSTEM : WOLFSHARE 1.0

Previous version of wolfshare application, we call it as wolfshare 1.0 was developed using PHP, MySQL, HTML, CSS and AngularJS.

Their application was working on a principle of good ratings and reviews. If user have to borrow or lend particular book from or to another user, they have to look for user rating and trust that ratings to go ahead with lending or borrowing process. There was no penalty or charge in terms of money even if books are mishandled or returned late or lost. Also, return request initiation was not provided in system. Return request was done offline in previous application.

*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

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

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. Let us see how potential users responded to our intial survey when we suggested some solutions to them for problems which were present in previous system.

A. Would you like to lend a book to someone when you dont need it?

With this question, we wanted to emphasize on the part when the user doesnt need that book. This will reduce the shelf-time of each book. As we can see from Fig. 1, the people are not sadist in nature. They dont 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 persons 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.

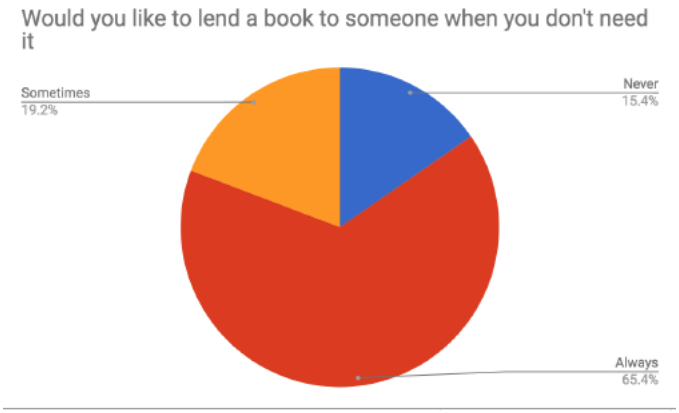

Fig. 1.

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 doesnt return it. Hence, as expected, the students are reluctant to lend a book.As seen in Fig. 2, 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.

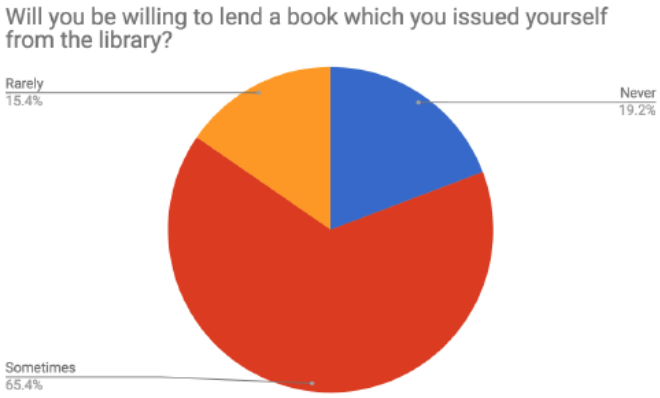

Fig. 2.

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.1 and Fig.2, 62% people are willing to lend the book if they are not held responsible for any damages. Now, this can be the basis of our hypothesis that People dont 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.

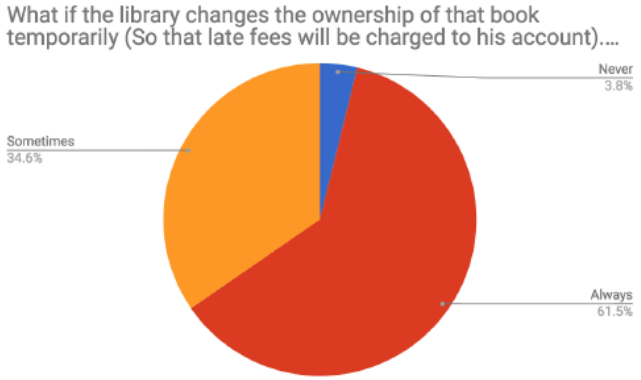

Fig. 3.

D. Will you be willing to give 10$ as deposit before borrowing a book from someone?(Youll 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.4, demonstrates peoples commitment and understanding towards proving their genuineness. Hence, we think that just connecting the users 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.

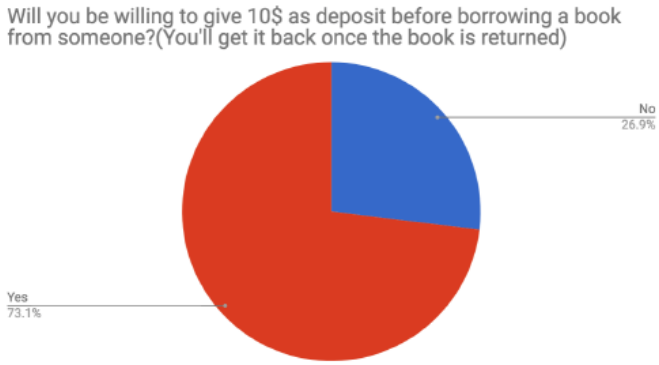

Fig. 4.

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.

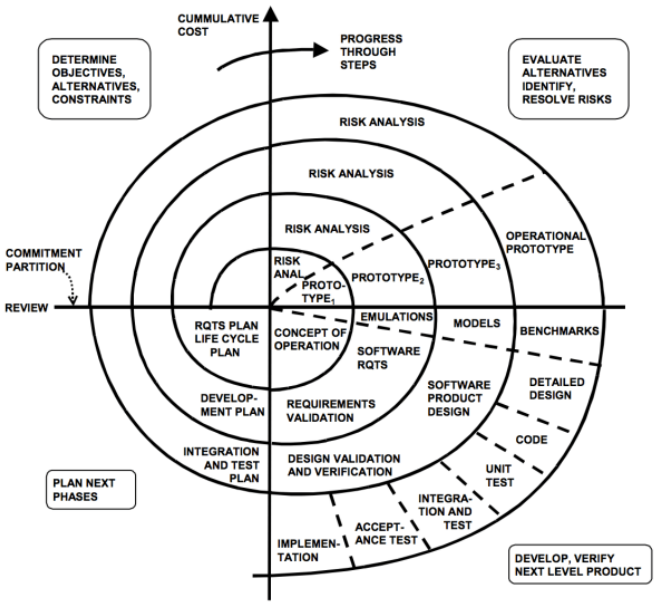

Fig. 5. The original spiral model diagram. Credit Barry Boehm, & Wilfred J. Hansen

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.

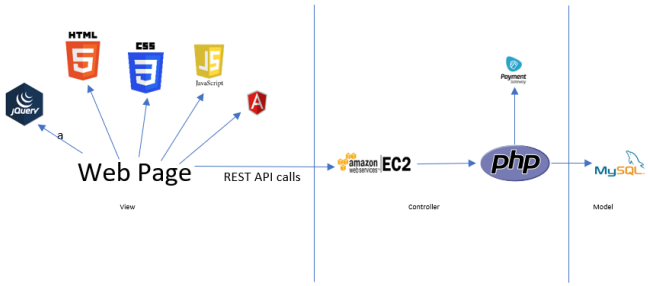

Fig. 6. System Architecture

As we can see in Fig 6, 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

YET TO COMPLETE

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.

VI. IMPLEMENTATION

YET TO COMPLETE

A. Payment Integration

B. Geolocation

C. Online Return Request

D. 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 projects github readme.md page. It includes, installing WAMP server, placing application folder into www, 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

YET TO COMPLETE

VIII. CONCLUSION

YET TO COMPLETE

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/MarchReportPhase2.pdf
[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