Borrowing System

Borrowing Management Web Application

Feb 20, 2017

Author: [Hesam Faraji]

Contents

1.Introduction

Developing web applications that satisfy client needs depend on clear and thorough understanding of client requirements and expectations from the desired system. For example, developing complicated website with lots of extra functionalities may confuse the users, and it would be hard or impossible for them to achieve their desired tasks, since users don't know how to work with such system that provides too many unnecessary features. Indeed, all functionalities that client expect from the system should be provided properly, while any missing functionality may decrease usability of system. Therefore, in this project there has been a research performed to elicit the requirements of the system comprehensively before the development of system.

Regarding the requirements of Fh-Kiel laboratory in this master project, relative website has been developed to fulfill almost all functional and nonfunctional requirements provided with a simple and user-friendly user-interface. In this project, unnecessary and extra functionalities were avoided to maintain the usability and simplicity of website.

The borrowing system website provides the possibility that student can easily view available items in laboratory and send request to borrow the items while they can view pervious and current items they borrow from lab as well. Lab technicians can add, edit, and delete items respectively while there are provided with suitable features to manage user`s requests.

To implement this website, PHP programming language has been used for server side development which is popular in various web template systems, web content management systems and web frameworks. CakePHP is the chosen framework to develop this system. CakePHP is an open-source web framework and it follows the model–view–controller (MVC) that make code clean and speed up the development process. for database MySQL is used to manage the data in system that is the best option for the system regarding requirement. HTML5 CSS3 and bootstrap are used for client side development and UI design.

1.2. Motivation

The main goal of the project is to provide a web application for students to let them borrow items from FH-Kiel. The app should be easy to use and understandable to users while it provides all functionalities that are needed for borrowing activities. Implementing comprehensive stakeholder's analysis is key point to reach objectives. One of the other key factors of this project for me is to learn the principals and fundamentals of web application development using php language which in turn brings me a great opportunity to learn a lot about PHP language itself and CakePHP framework which is an MVC web development framework.

1.3. Problem Statement

There is a library system in FH-Kiel that is used to lend books to students, but this system is not adequate since there is no possibility for students to borrow items for research and development purposes, they can only borrow the book from library. Respecting to large number of students in FH-Kiel who study with in different faculties and considerable number of research projects, there is a need for a system to make it feasible to borrow items to students and keep tracking of available items in the university. On the other hand, the lab technicians who are responsible to lend items to students need system to manage borrowed items and record all data about the students and items to keep tracking borrowed items and available items.

Per current situation in FH-Kiel, there is an evident lack of suitable borrowing system, so developing an efficient web application that provides a platform for students and university admins (lab technicians) to borrow and manage borrowed items is crucial. this project tries to develop this system to solve current problems to ease the borrowing process. developing such system needs comprehensive research about the requirements, therefore a research has been conducted amongst students, professors, lab technicians and other stockholders. previous documents are precisely reviewed, and both pervious web applications are also analyzed.

1.4. Existing System

There were two systems developed to manage borrowing activities. first one is current system that is used to borrow items to students but this system is not commonly used because some deficiencies exist in this system that make this system useless.

The current system doesn't support transactions for multiple users, and it also cannot be used for admin user, and is not suitable for students to view the available items and make requests to book items. The user interface is poor so users are totally confused on how to do their desired actions. Furthermore, items are not categorized, so finding the right item in the site is a difficult task to do. The quantities of items are not indicated, so users do not know whether item is available, or how many of that item may exist at all. in general, management of borrowing activities is a daunting task and it would create troubles while managing items.

The second system that developed by student provides lots of functionalities that are not necessary for FH-Kiel and it just makes the system complicated and unclear to users, some features such as offers to groups, chat between students and so on. In addition, none of the functionalities of developed system is working properly.

1.5. Proposed System

A new system developed to satisfy requirements of the stakeholders and solve problems associated with current system. The new system provides possibility for students to browse among items and borrow the desired items. Finding desired items for student is totally easy since all items are categorized in different categories and they just need to refer to desired category. in addition, students can search for items by just typing item name to find that. Students can review the borrowed items in system. they also get confirmation mail when they borrow items, therefore they will be aware of details of borrowing and time of returning item.

In new system admin user, can lend items to student by just confirming the user request or just going to user profiles and assign the item to users. returning the item is also easy admin should enter return date of item, then item will return to system. admin can post new items in system, manage items whether edit or delete, and keep track of items to know which item is borrowed to which student.

There is possibility for admin user to verify that students return all items before their graduation admin needs just to view the user profile to make sure students have returned all items. The developed website offered in two languages. user can choose language between English and German, since there is lots of international student use this system it is essential for web site to be Multilanguage. the system also has possibility to integrate with LDAP server to verify student authenticity.

1.6. Distributed Borrowing System

1.6.1. Introduction

"Distributed system is a collection of independent computers that appears to its users as a single coherent system" [12]. The computers are independent from each other. they communicate with each other via the messages over the network, the messages can convey information and instruction. Computers can tell other computers to execute procedures or send and receive packets of data via the messages.

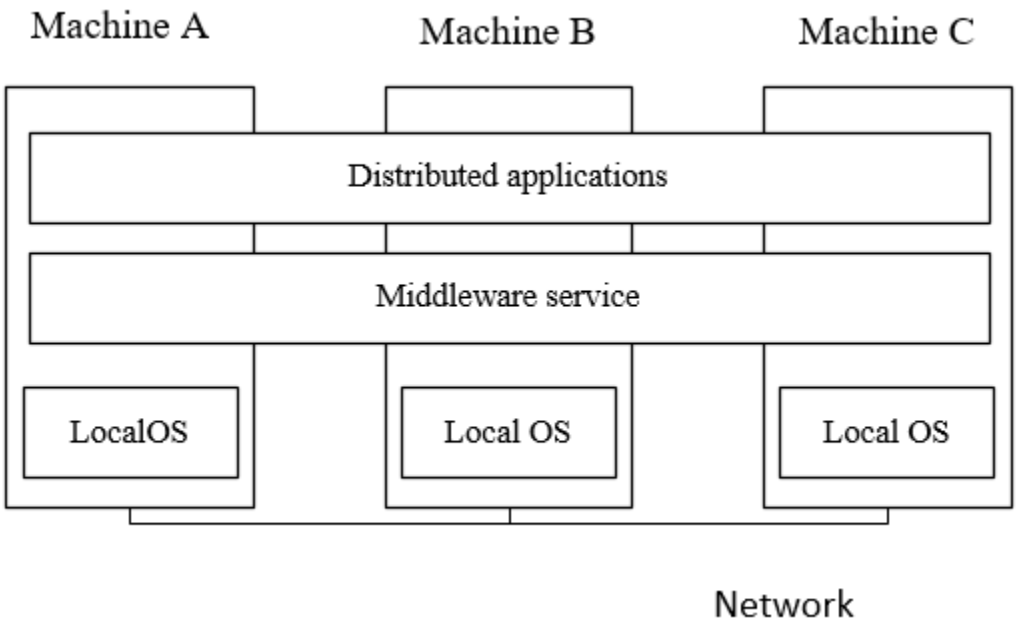


Figure 1 Distributed System

In distributed system, multiple computers working on single problem, the problem is divided between the computers, therefore each computer has certain tasks, they are communicating to each other through the network until they solve the problem.

The main goals of distributed system are to achieve high level of the scalability, sharing resource, distribution transparency, openness and provide fault tolerant system that enables the resources to be accessible in case of one of component failure.

1.6.2. Borrowing System

Using distributed computing in borrowing system helps to maximize the performance of the system by connecting users and resources in a cost efficient and reliable manner. with such a system, it is easier for users to access the system and share the resources with each other. This would also make system scalable, so we can serve to a very large number of users and if the number of users increase dramatically this would be no problem for system to serve. Another advantage of using such system in borrowing system is in case of hardware and software failure in any component system would not be failed, other component can do the failed part task, so there is no concern for system availability and system will be more reliable [14].

In conclusion, if we have distributed borrowing system our system would be more powerful and reliable, but for now as per our requirement and number of the current users, this would not be cost effective to develop the distributed system. But for future this may be applied to current system .in my opinion current system will satisfy all the requirements of the current users and system would work properly to serve the current users.

One of the successful distributed systems is eBay with 276 million registered users that are being served in different countries eBay is an eCommerce website that a user can browse to its website eBay.com and search for anything they want to buy. This procedure is quite like borrowing procedure that we have in borrowing system. Users of borrowing system can browse to website and book their desired items instead of buying item. The website like eBay that should handle large volumes of the requests every day and respond these requests in timely fashions needs to incorporate distributed computing system concepts to provide high performance and high availability to their users [15].

1.6.3. Examples of Distributed Systems

In this section, there are some examples about current distributed system.

Web Search

There are considerable growth in using web search in last decade. the recent statistics show that the number of searches have risen to over 10 billion per months. Web search engine indexes the whole content of the world-wide web, that contains the wide range of information style which include web pages, books and multimedia sources. this is very difficult task, since world wide web is huge, it contains over 63 billion pages and one trillion unique web addresses. Analyzing entire world wide web content by search engine is sophisticated process that represents major challenges to design distributed system [13].

Google, a pioneer in web search technology has designed sophisticated distributed system to support millions of search requests in day. in below, there are some highlights about google distributed system [13].

1. There are large number of networked computers used in each data center all around the world.
2. There is a distributed file system which supports large files and optimized for usage required by search
3. There is a distributed storage system that is associated to search engine which offers access to large data set.
4. There is lock service which offers the distributed system functions including the distributed looking and agreement
5. There is a programing model which supports distributed computations within physical infrastructure

Massively Multiplayer Online Games (MMOGs)

In massively Multiplayer games there are large number of users interacting through internet in game environments. Sony's EverQuest II and EVE online are two leading providers in market that serve to huge number of users around the world. For example, EVE game has capability to support more than 50,000 simultaneous players. Designing such a game is a great challenge for

designers since such system needs fast response times to keep user experience of the game. Another challenge is to real time propagation of events to many users and maintain consistent view of the game environment. These are some good example of main challenges being faced by distributed system designer [13].

The largest online game, EVE online, uses the client-server architecture that centralized server accessed by client programs running on user's PC. To serve this large number of users, complex server is used which consists of a cluster architecture with hundreds of computers[13]. Management of the game environment and maintain consistency with this type of architecture is easier. The main goal in the EVE online is to have fast response through optimizing the network protocols, and make rapid response to incoming events to achieve this goal. the load is partitioned within computers in cluster, incoming event goes to the right computer [13].

Most of the MMOGS use more distributed system that the load is partitioned across number of servers that are located in different locations around the world .users are dynamically redirected to server. the system can be expanded by adding new servers.

Most of the companies utilize the two models that discussed above but there is new approach that is completely decentralized which is based on peer-to-peer technology where every participant contributed in resource sharing which include storage and processing game data.

Financial Trading

In financial industry, there is need to have real time access to the wide range of the information. distributed system technology is the best solution to provide the reliable services to the users. The emphasis in Financial trading systems are based on communication and processing item of interest known as event, the event should deliver reliably and timely to the potential clients that have interest in such information. For example, drop in share price, the latest unemployment figure are events that should deliver to client on time [13].

2.Project Management

Project management is the art and science of planning and leading software projects, that indicates how software project are planned, implemented, monitored and controlled. In this project, agile project management strategies have been chosen, therefore agile strategy has been followed in every stage of the project. it provides a flexible software development model that helps to address the agility of requirements of borrowing system project.

In this project, each project task is divided into smaller tasks that would be helpful to develop the project as fast as possible and modification will be easier to incorporate. as it is mentioned previously CakePHP has been used in this project (MVC framework), so there is modular design in application as it separates the application code into three parts: model, view, controller that each of them also are divided into smaller sections. this helps to classify related parts of applications together to have a better management over project.

In addition, there have been several contacts made to stockholders during the development phase of the project to gather requirements and getting feedbacks on system being developed, so if some changes needed to be applied, they will be considered and planned as soon as possible.

As first step in the project some meetings with stakeholders conducted to identify their desired features that they want to see in the application which are called user stories. when the list of final features is identified, then using agile estimation techniques, size of each user stories is indicated through an approximation of each user story on how long it may take to implement. Now, it is time to prioritize the user stories by stakeholders. After prioritization, development is started and stakeholders are asked to provide feedbacks after implementing each stage of project, if some changes are required, they will be considered in system.

3.System and Application Development

3.1 Introduction

As it was mentioned before in this project, CakePHP framework has been used as the main framework to develop borrowing system application, CakePHP makes development straightforward and faster while requiring less code. CakePHP follows the MVC software design pattern that Crafts application tasks into separate models, views, and controllers which make application very light and manageable, New features can easily be added, designed, and development can be done simultaneously. Changing in one part of application doesn't affect other part of application.

NetBeans IDE is used to write PHP code, phpMyAdmin is used to create and manage database. XAMPP is a free and open source cross-platform web server solution stack package that is used in this project to utilize Apache server and MySQL database.

3.2. Application Structure

The following folders are existing in the application. these folders show the structure of application. Each folder is a component of application that do certain task in application. for better understanding and future recusing code there is explanation below for whom to have depth understanding about the project [1].

- bin
- config
- logs
- plugins
- src
- tests
- tmp

- vendor
- webroot
- .htaccess
- composer.json
- index.php
- README.md

The *bin* folder is responsible to holds the application console executables.

The *config* folder contains some configuration files. Database connection, bootstrapping, core configuration files and another configuration store here.

The *plugins* folder contains all Plugin that used in application.

The *logs* folder contains log files of application,

The *src* folder contains the application file and place that application code was written.

The *tests* folder contains test cases for application.

The *tmp* folder is responsible to store temporary data. This folder normally used to store model description and session information.

The *vendor* folder is place that other application dependencies will be installed. This file strongly recommended to don't edit file in this folder.

The *Webroot* directory is the public document root of application. It contains all the files that public ally reachable.

The src Folder:

Application's src folder is place that most of the development process where done here. therefore we have closer look on folders inside the src folder:

Console

Contains the console commands and console tasks of application.

Controller

All application controller and their component are placed here

Locale

Stores language files.

Model

Applications tables, entities and behaviors are placed here

View

Presentational classes like cells, helpers, and template files, are store here

Template

All presentational files are store in template file like elements, error pages, layouts, and view template files [1].

3.3. Application Architecture

The architectural pattern of the applications follows the MVC software design pattern, that separates application into three main parts:

The Model layer represents part of application which implements business logic. it is responsible for retrieving data and convert it to meaningful concept for application. This layer includes validation and data processing [1].

The View layer is responsible to produce presentational interface using modeled data. for example, when model contains some data the view could use it to display this data in HTML page. The view can be used to represent a wide variety of data formats such as document, video, music and any other format [1].

The Controller layer handles the user requests. it prepares the response using model and view layer. controller acts as a manager to make sure that all the resources are available to correct worker to prepare the product for client [1]

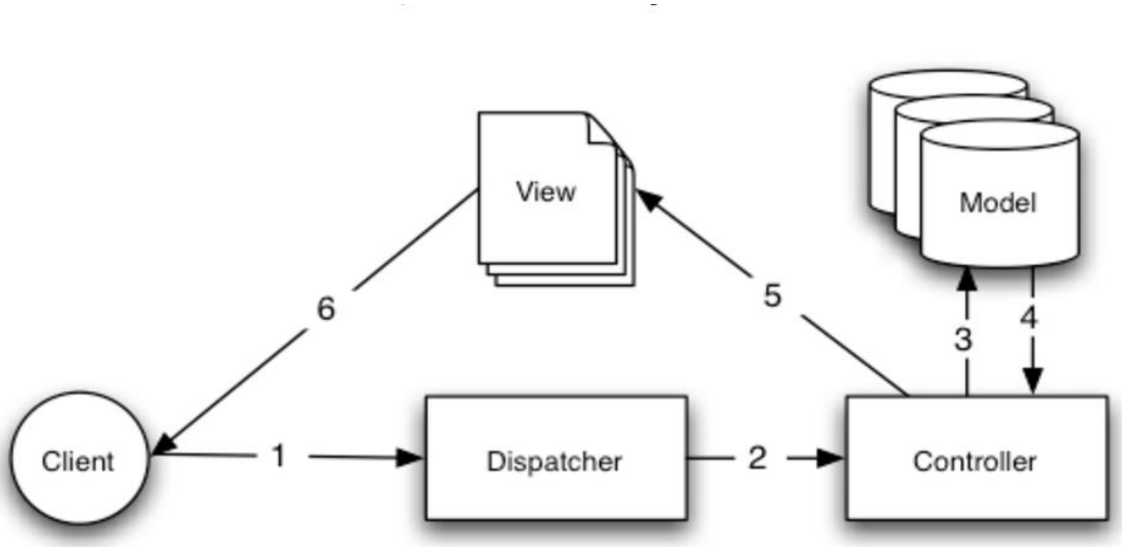


Figure 2 MCV Request in CakePHP

Above figure demonstrates CakePHP request cycle, the cycle starts when client requests a page or resource from application. Request is passed to dispatcher to decide which controller should handle request, after that dispatcher finds the correct controller and request is forwarded to that controller [1].

Once controller receives request, it communicates to model to process any data fetching or saving operation depending on request needs, then controller finds the correct view object to output the result to user [1].

3.4. System Analysis

3.4.1. Stakeholders Identification and Requirement Elicitation

As first step in requirement analysis stakeholders are identified, these people are whom that have valid interest in Borrowing system. they may have affected whether directly or indirectly. these people play important role in requirement analysis, the requirement of system declare base on

this stakeholder's interest. The following stockholders are identified for borrowing system application

- Student
- Laboratory Technicians
- Professor
- Financial Manager
- System Admin
- Dean

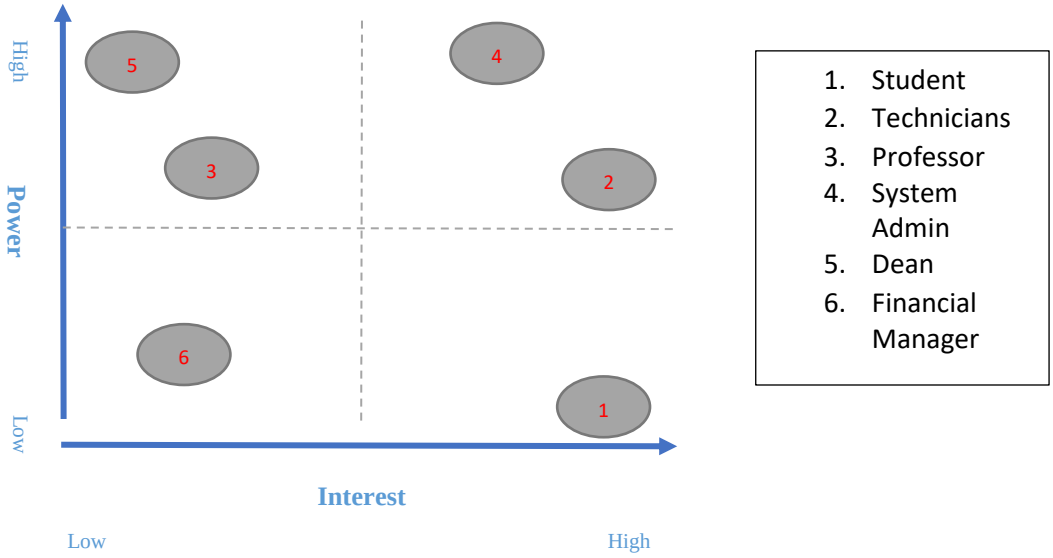
The Controller layer handles the user request .it prepares the response by using model and view layer. controller act as manager to make sure that all the resource is available to the correct worker to prepare the product for client.

In this section, all requirement gathered from the identified stakeholders through the various elicitation techniques. in following table shows that which techniques used to elucidate requirement from stakeholders.

Table 1Stackholders and Elicitation Technique Used

Stockholders	***Elicitation Technique***	***Reason***
Student	Questionnaire, Prototyping	Large Number of student high involvement, require good validation
Laboratory Technicians	Interview	High involvement, require direct contact
Professor	Interview	High knowledge and experience, require direct contact
Financial Manager	Questionnaire	Low involvement, low experience and knowledge
System Admin	Interview	High involvement, high experience
Dean	Questionnaire	Low involvement, lack of time

3.4.2. Stakeholders Matrix



3.4.3. Functional and Nonfunctional Requirements

This document is to be interpreted as described based on RFC 2119 standard

Definition:

Shall: Required feature

Should: recommended feature

Functional Requirements:

1. Every user shall be authenticated by system.
2. The system shall define different roles for users (Admin, Normal (student)).
3. The system shall provide permissions and access rights based on user's role.
4. The system shall allow admin to offer items to students.
5. The system shall accept image for offered item.
6. The system shall allow admin to manage offered items.

7. The system shall allow admin to lend items to students and store all data about student, item, borrow and return date.
8. The system shall allow admin to receive item from student and update database respectively.
9. The system should send confirmation email to the student upon borrowing confirmation.
10. The system shall allow students to sign-up in system.
11. The system should allow students to edit their data.
12. The system shall allow admin to manage users.
13. The system shall support localization for English and German languages.
14. The system shall allow students to send requests for borrowing items
15. The system should allow students to extend due dates.
16. The system shall show all available items to the students.
17. The system shall display the available quantity per item.
18. The system shall update database after each transaction (borrow and receive items).
19. The system should classify items categories.
20. The students should be able to view the history of their transactions (borrowed items)
21. The admin should be able to track all student's transactions to verify students have return items before their graduation.
22. The students should be able to search for their desired items.
23. The admin should be able to search through students in the system.
24. The system should record all user activities.
25. The system shall validate all user inputs.
26. The system should send periodic reminders to students regarding their borrowed items due dates

Non-Functional Requirements:

1. The system shall be accessible through the web.
2. The system shall be secure.
3. The system shall be available 24/7 with no down time.
4. The system shall be reliable.
5. The system shall be manageable.
6. The system shall be responsive.

7. The system should be scalable.
8. The System should have documentation
9. The system should be integrated with the current LDAP

3.4.4.UML Use Case Diagram

The following diagrams are usecase diagram of the Borrowing System.this diagrams represent a set of the actions that borrowing system performed in collbration with users of system.

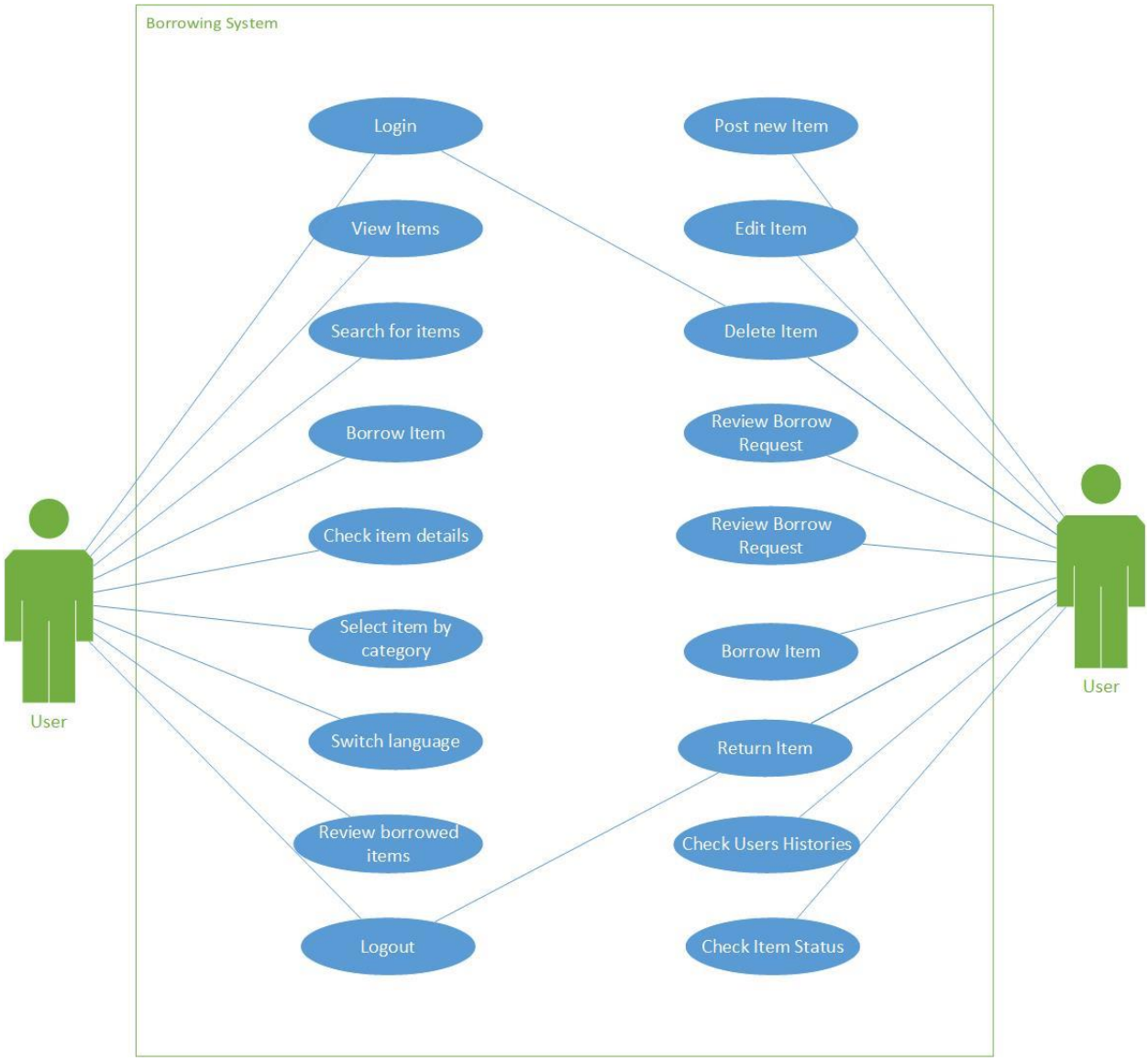


Figure 3 Use Case Diagram for whole Borrowing System

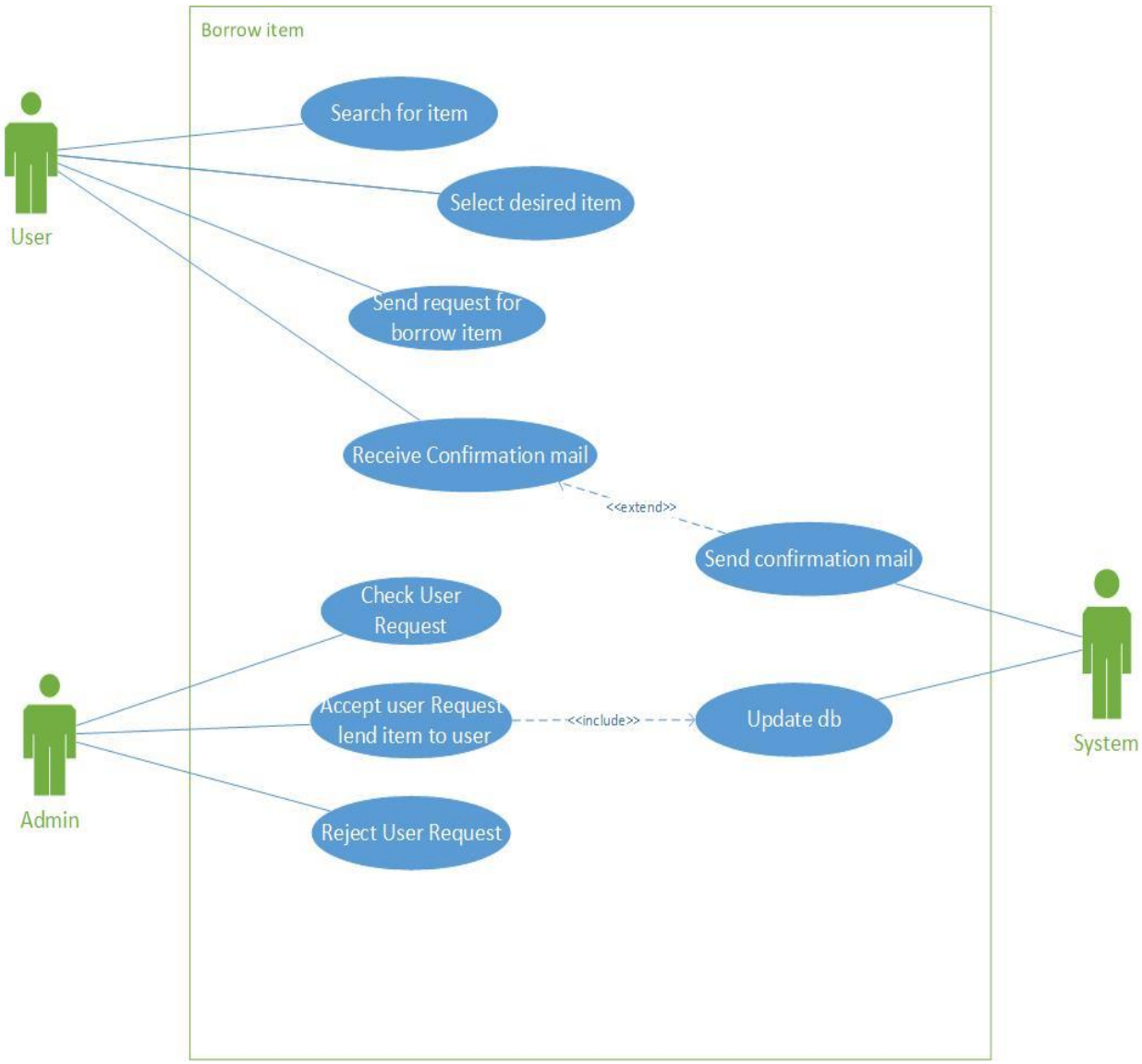


Figure 4 Use Case Diagram for Borrow Item

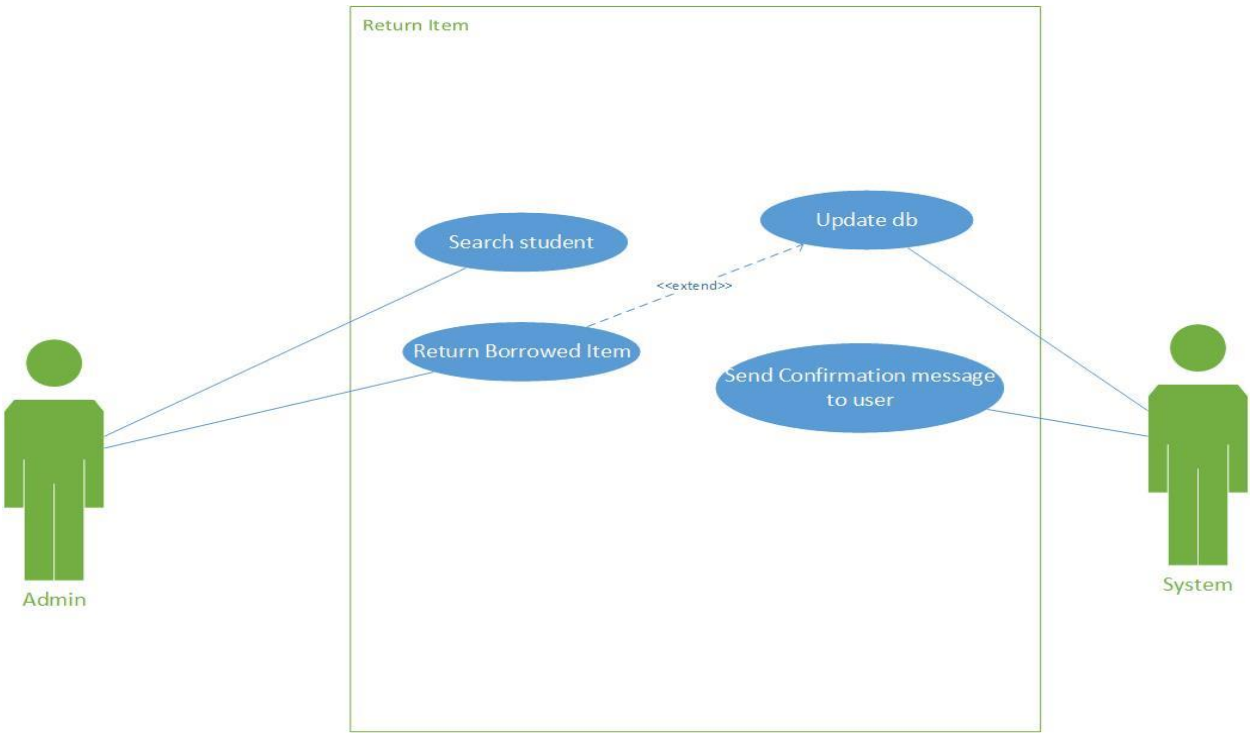


Figure 5 Use Case Diagram for Return Item

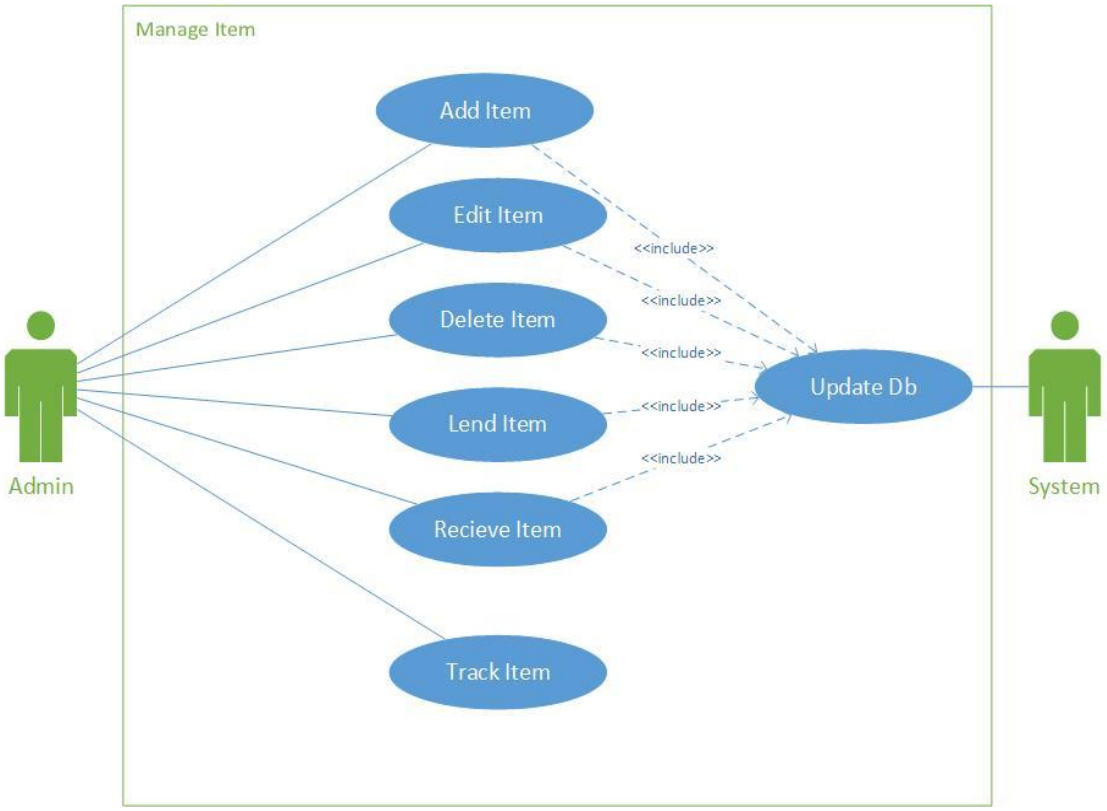


Figure 6 Use Case Diagram for Manage Item

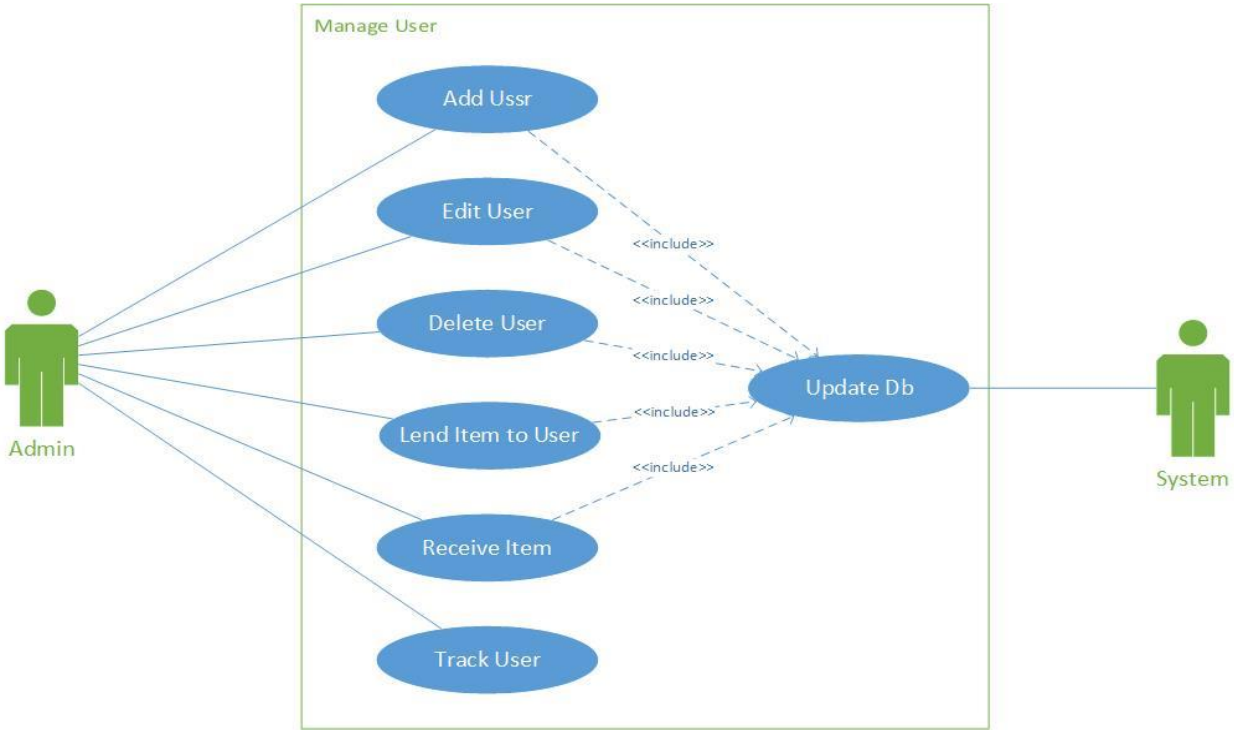


Figure 7 Use Case Diagram for Manage User

3.4.5. Activity Diagram

Activity diagram is basically flow chart that represent flow from one activity to another activity in system the flowing activities diagrams shows how different operation is done in system .

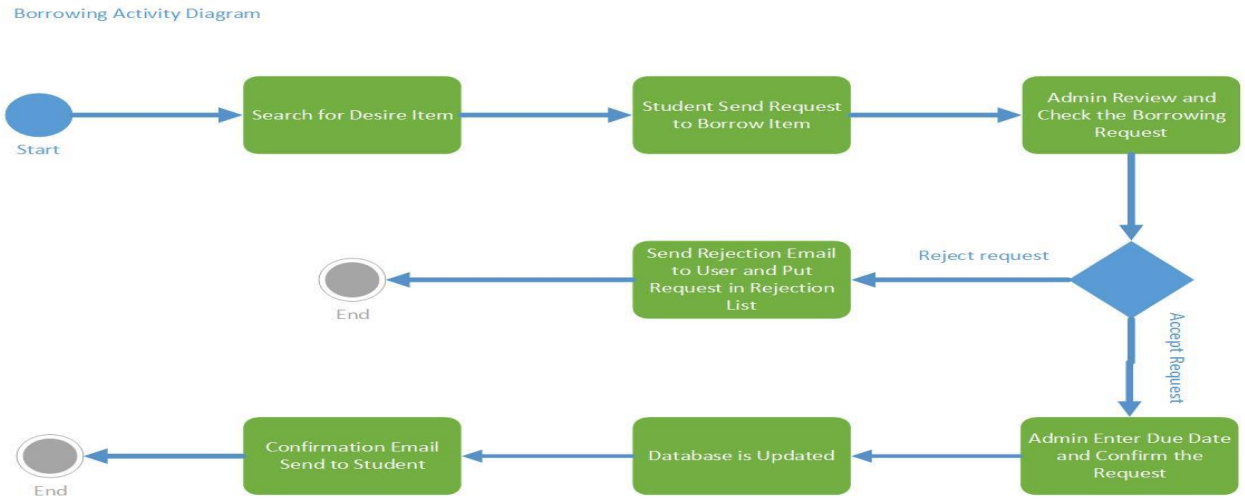


Figure 8 Activity Diagram for Borrowing Item

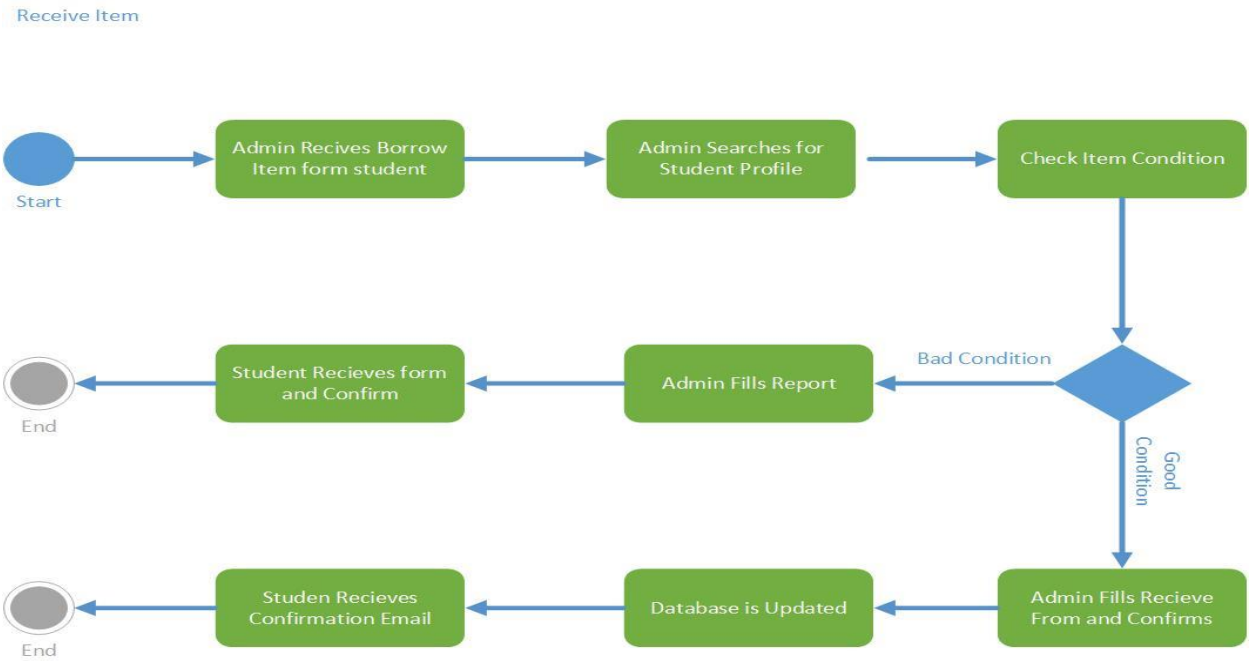


Figure 9 Activity Diagram for Receive Item

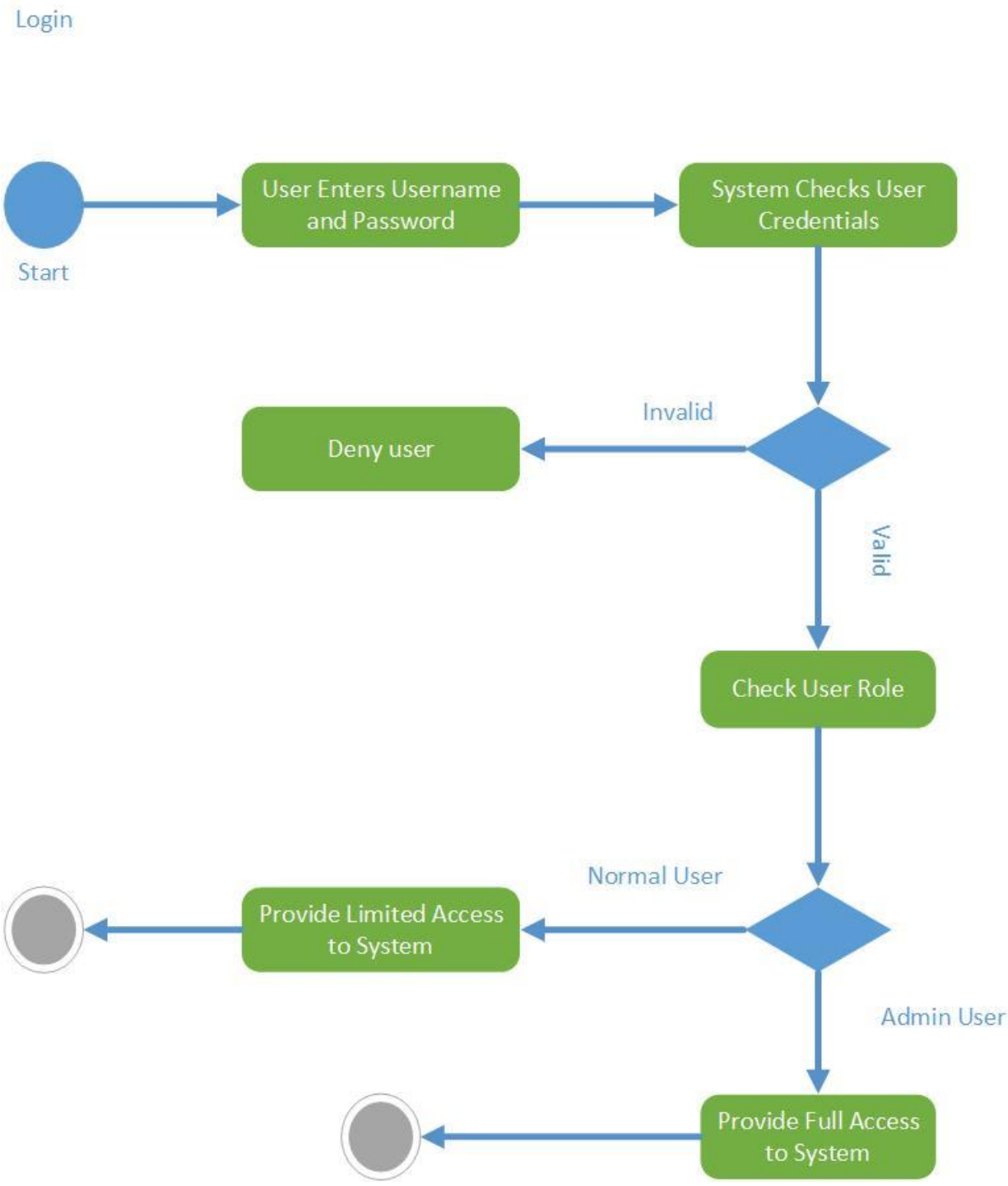


Figure 10 Activity Diagram for Login

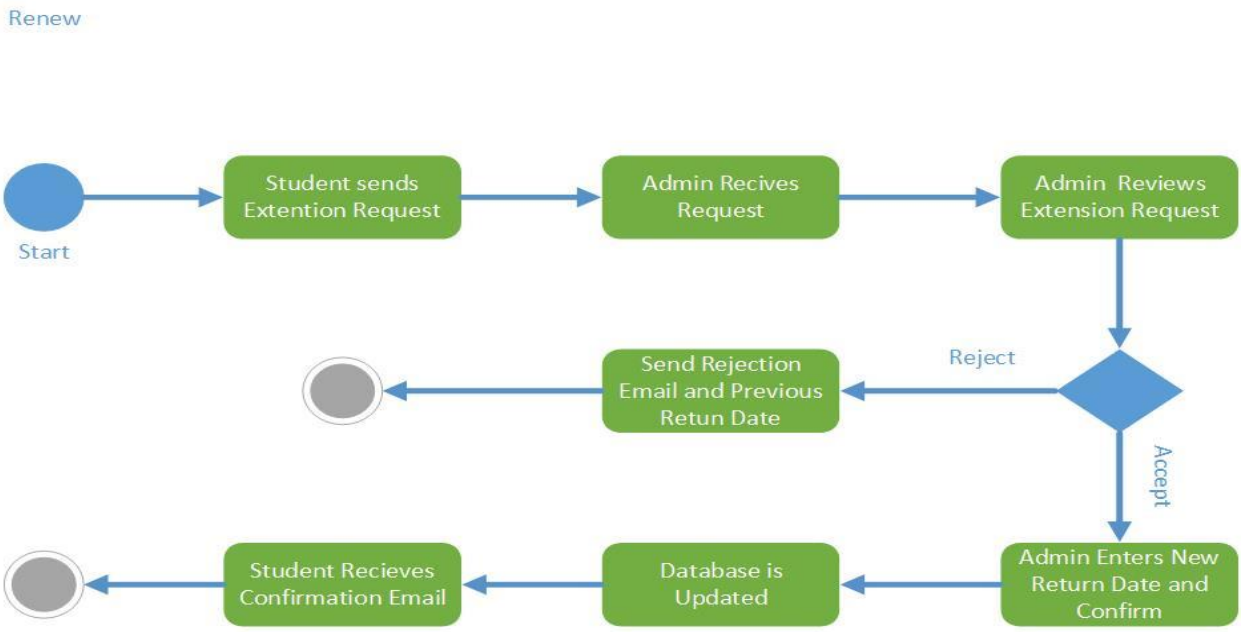


Figure 11 Activity Diagram for Renew

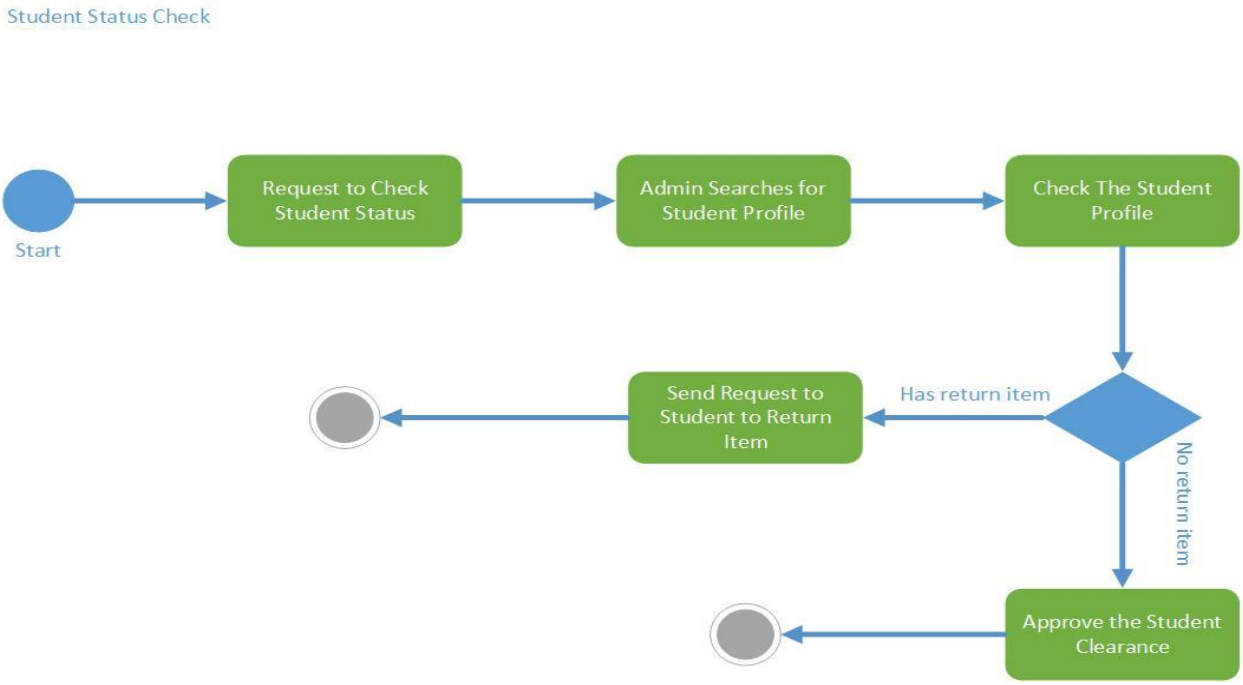


Figure 12 Activity Diagram for Student Statues Check

3.5. System Design

3.5.1. User Interface Design

User interface design is the process of arrangement and placement of user interface elements on the most external layer where users are intended to dealing a system through it. for machines and software, like computer, mobile device and other electronic devices with focus on maximizing the usability and user experience. the main goal of the user interface design is to make user interface as simple as possible and be efficient to accomplishing the user goals.

The key point in user interface design is to knowing the user very well, designer should have clear knowledge about the user's goals, skills, preference and tendencies unless he cannot make suitable user interface design.

In this project, frequent and regular contact with users have been made to understand their requirements and preferences to make user interface as efficient as possible.

In this project HTML5, CSS3 and bootstrap are used to design borrowing system app. all view codes for app are placed in template files that are stored in src/template in a folder named after the controller that uses the files, and same name with correspondent action. For example, the view file for the Items controller's "add" action, would normally be found in src/Template/Items/add.ctp. and you can find all bootsrap and css file in webroot/css and it would be loaded for desired page with $this->Html->css('flie.css') statement.

Layout are used to reuse the common sections in different pages of the project .it contains presentation code that wraps around view.

Default layout is located at src/Template/Layout/default.ctp. overall look of the application contains in default layout, but there are some different layouts in the app that can be founded in src/Template/Layout.

3.5.2. Database Design

Per the requirements, MySQL has been chosen as database platform for the Borrowing System application. MySQL is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL). It is cross platform that can be run in all platforms, including Windows, UNIX and Linux. MySQL commonly used in web-based applications. however, it can be used in a wide range of applications.

PHPMyAdmin is used to perform the administrative tasks for MySQL in this application. it supports a wide range of operations on MySQL such as Managing databases, tables, relations, users, indexes, user permissions etc. These operations can be run either through sql command line or via user interface.

As it is depicted in the following ERD diagram:

- A role can be assigned to many users while each user may hold one and one role at a time
- A user may have many loans at any certain moment and each loan may have one or more items with it.
- An item should belong to one category, and each category may have one or more items
- A user may have many orders at any certain moment and each order belongs to one and only one user while each order may have one or more items with it.

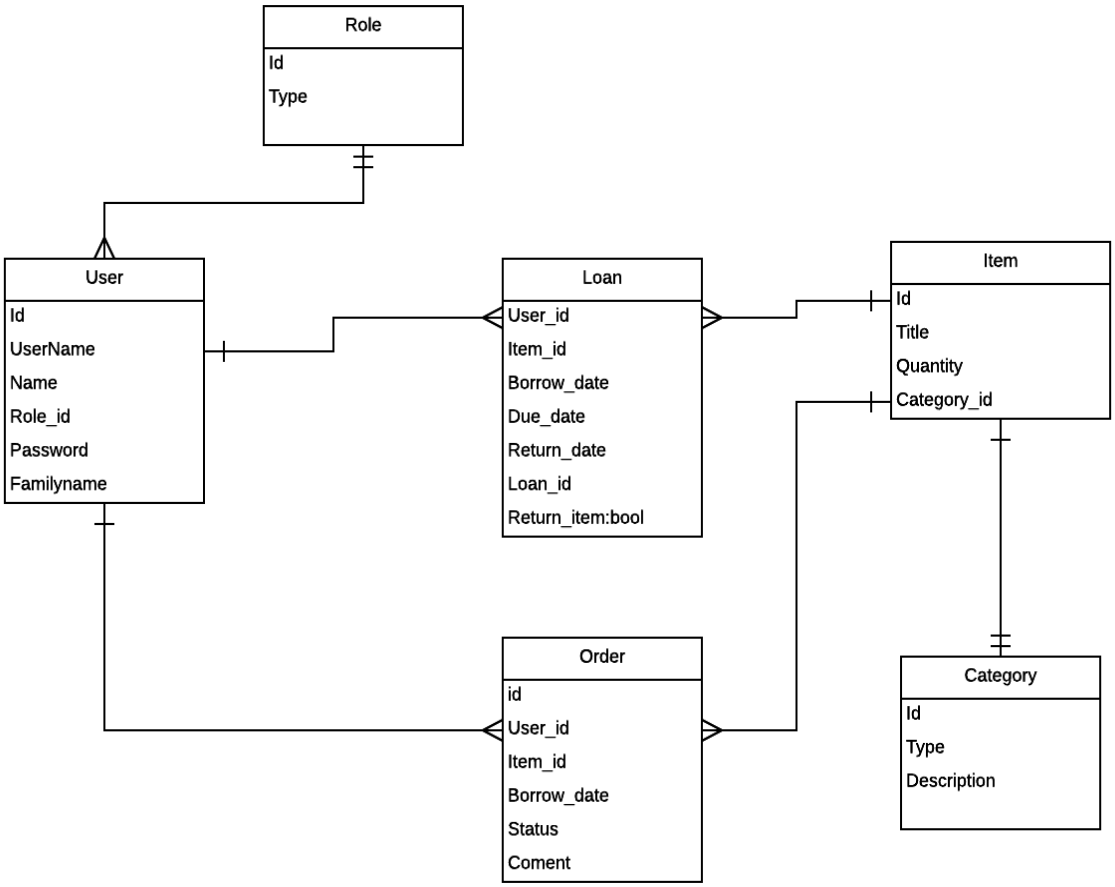


Figure 13 ERD Diagram for Borrowing System

Table	Create Table
items	CREATE TABLE `items` (`id` int(11) NOT NULL AUTO_INCREMENT, `title` varchar(100) NOT NULL, `quantity` int(11) DEFAULT NULL, `category_id` int(11) DEFAULT NULL, `created` datetime DEFAULT NULL, `modified` datetime DEFAULT NULL, `photo` varchar(255) DEFAULT NULL, `photo_dir` varchar(250) DEFAULT NULL, PRIMARY KEY (`id`), KEY `category_key` (`category_id`), CONSTRAINT `category_key` FOREIGN KEY (`category_id`) REFERENCES `categories` (`id`)) ENGINE=InnoDB AUTO_INCREMENT=2777 DEFAULT CHARSET=latin1

Figure 14 SQL Code for Items Table

Table	Create Table
categories	CREATE TABLE `categories` (`id` int(11) NOT NULL AUTO_INCREMENT, `type` varchar(255) NOT NULL, `description` varchar(255) NOT NULL, `created` datetime DEFAULT NULL, `modified` datetime DEFAULT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB AUTO_INCREMENT=179 DEFAULT CHARSET=latin1

Figure 15 SQL Code for Categories Table

Table	Create Table
orders	CREATE TABLE `orders` (`id` int(11) NOT NULL AUTO_INCREMENT, `user_id` int(11) NOT NULL, `item_id` int(11) NOT NULL, `borrow_date` date DEFAULT NULL, `comment` text, `created` datetime DEFAULT NULL, `modified` datetime DEFAULT NULL, `status` int(11) DEFAULT NULL, PRIMARY KEY (`id`), KEY `user_id` (`user_id`), KEY `item_id` (`item_id`), CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`), CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`item_id`) REFERENCES `items` (`id`)) ENGINE=InnoDB AUTO_INCREMENT=109 DEFAULT CHARSET=latin1

Figure 16 SQL Code for Orders Table

Table	Create Table
loan	CREATE TABLE `loan` (`id` int(11) NOT NULL AUTO_INCREMENT, `item_id` int(11) NOT NULL, `user_id` int(11) NOT NULL, `borrow_date` date DEFAULT NULL, `due_date` date DEFAULT NULL, `return_date` date DEFAULT NULL, `Return_item` tinyint(1) DEFAULT NULL, `created` datetime DEFAULT NULL, `modified` datetime DEFAULT NULL, PRIMARY KEY (`id`), KEY `user_key` (`user_id`), KEY `item_key` (`item_id`)) ENGINE=InnoDB AUTO_INCREMENT=168 DEFAULT CHARSET=latin1

Figure 17 SQL Code for Loan Table

Table	Create Table
users	CREATE TABLE `users` (`id` int(11) NOT NULL AUTO_INCREMENT, `email` char(250) NOT NULL, `firstname` varchar(256) DEFAULT NULL, `role_id` int(11) DEFAULT NULL, `created` datetime DEFAULT NULL, `modified` datetime DEFAULT NULL, `password` char(96) DEFAULT NULL, `lastname` varchar(256) DEFAULT NULL, PRIMARY KEY (`id`), UNIQUE KEY `email` (`email`), KEY `role_key` (`role_id`), CONSTRAINT `role_key` FOREIGN KEY (`role_id`) REFERENCES `role` (`id`)) ENGINE=InnoDB AUTO_INCREMENT=102 DEFAULT CHARSET=latin1

Figure 18 SQL Code for Users Table

Table	Create Table
role	CREATE TABLE `role` (`id` int(11) NOT NULL AUTO_INCREMENT, `type` varchar(100) NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB AUTO_INCREMENT=105 DEFAULT CHARSET=latin1

Figure 19 SQL Code for Role Table

3.6. System Implementation

To develop this system PHP programming language has been used. To make the development process more straightforward and more manageable MVC design pattern is used which in the realm of PHP, CakePHP is one the frameworks that supports MVC to create web applications.

As the amount of code and its complexity due its requirement is at a level that constructing and modifying them using normal text editors is a cumbersome and error prone task to do, an IDE (Integrated Development Environment) is used to make the development process more manageable. NetBeans has been used as the main development IDE for this project which is an open source and free IDE.

To have a server solution to run and test the application during the development XAMPP which is a free and open source cross-platform web server solution has been used as this stacked package provides many services such as hosting using Apache HTTP server database support and interpreters for scripts written in PHP and perl programming languages.

Mysql is the database used for this system. It is an RDBMS system which in this system can entirely supports the requirement needs. XAMPP makes it easier to access MySQL service to create, modify, query etc. visually through its web interface at a specific url and port.

3.7. Comparative Technology

3.7.1. Introduction

When it comes to have a comparison between these those programming languages, it is worth to talk a bit about their root and actual goals of these languages. PHP was created to run on server, in on other word it is intended to process the client requests and generate their corresponding responses while JavaScript language was introduced as a language to be run by browser to assist programmers on the client agents(browsers). There are so many powerful web application utilizing the successful collaboration of these two-programming language to make a comprehensive web application to satisfy all users need which at the end brings a better experience to them. PHP has been a server technology and still it is while it has gone through a lot of improvements but JavaScript has changed part of its path and now it be used to write application to can run on server to server client requests and as its obvious with this evolution in JavaScript it is now possible to make a web application completely in JavaScript while it does all its client and server side tasks. Node js is the framework that handle server side JavaScript programming aspects. In the following, I will try to have a comparison over these two-great frameworks [21].

3.7.2. Performance Analysis

As we know JavaScript is using well known V8 engine which is faster than PHP, but it can't be defined to a certain degree that how much it is faster. On the other hand, when we are talking about PHP performance, its performance is directly related to the web server whose is hosting it. It is hard to find a reliable and solid benchmark that compares these two frameworks together since a simple test application cannot prove it completely. Another important aspect on this topic is the way or better to say the paradigm that two frameworks follow. For example, a typical PHP request is handled as following [21]:

1) The request is received by a web server (Apache, Nginx, etc…)

2) The request should be dispatched to the right php handler

3) The PHP process should run the script

4) The necessary steps (startup/init) start to take place

5) Finally, the script is processed and hands over to the web server

6) Web server will forward to response to the client

7) PHP process will be recycled for another request

While if we want to have this whole process in node.js the followings steps are done in a typical case:

1) The request is received by a web server (Ngingx, this step is optional as it is only used as a load balancer, http or https, or web socket bridge) and it is dispatched to one of node.js workers.

2) Node.js service process the request while all the initial codes have already been executed upon service initialization time

Even driven and non-blocking IO

There are differences non-blocking mechanism of these two frameworks. While PHP only may handle a process at a time node.js handle as many requests as it can while a blocking IO is involved, in other word in the case that there is an IO which may take time to get completed, node.js accepts other requests to process while PHP does not. With node.js a lot of processing can be save as there will be as many as needed service processes created based on available resources on the system and event driven nature of node.js let the requests to register callbacks which will be triggered upon completion.

3.7.3. MySQL vs MongoDB

These are completely two different database systems in their nature, one is an RDBMS system while the other one is NOSQL. There are many advantages and disadvantages over these two different systems which led each to be suitable for specific scenarios. MySQL is an RDBMS database system which is pretty to for the data that can be normalized and it has a good support for transactional operations in the scope of the database. NoSQL which are a new generation of database have different categories but mongo dB is the one called document based. In the

document based database the data are not necessarily normalized as join operations are expensive in these types of databases, so the whole data is organized and saved in BSON format internally by database. As it is mentioned there are different scenarios that these systems are best for. For instance, if there is a financial system that need to have a high level of data granularity and transaction should be supported in the system, MySQL could be a good choice as it is powerful enough to make required join to build the final queries and transaction are supported by this system. But on the other side, there could be a blog or even a chat system. In the case that for instance an article may have several comments and each comment may have many replies, it is not desirable to store these types of data in a relational database as it should be partitioned into several tables and joins are needed in the reading time. In a document based database, the who article and its comments can be stored into a single document, hence the data can be retrieved with less process and faster comparing to its RDBMS counterpart. Apparently, as the whole related data is stored in a single document in these types of database, the database can be scaled with less effort as there is no dependency to other document (in most cases) and the document is independent. As each of these database technologies are right to address different requirements, they can be mixed and a hybrid solution can be designed to get the best of each world [24].

4.Application Testing

To ensure the quality of the developed application, some tests performed to on the quality of application under test. This helps to prove the usability of the borrowing system to the stakeholders to have them convinced that application meets their requirements properly.

PHPUnit is selected as the testing framework to perform testing within the application. it is the de-facto standard for testing in PHP. It offers a set of powerful features to test application and make sure that app behaves as expected. it uses assertions to verify performance of the specific component of application. PHP unit is well integrated with CakePHP and can be installed using either PHAR package or Composer.

4.1. Unit Testing

To test individual units of source code unit test has been performed in this application. Units are entities like model and table classes in this system. Therefore, unit tests are implemented in each table class in app to perform unit test. As the first step, test case should be written for each function in the class. for clear understanding and code reuse for future, there is an explanation about the unit testing in the *items table class*. But for other classes you may refer to source code in the tests/TestCase/Model/Table directory. To perform unit test in items table, there are some functions available in tests/TestCase/Model/Table/ItemsTableTest.php file to test the main functionality of the Items table.

The first test is item addition test in items table to make sure that user can add item into the table without any problem and error, so the following function tests whether new item is added to the database successfully or not.

```
public function testItemAdd(){
    $items = TableRegistry::get('items');
    $query = $items->find('all');
    $categories = TableRegistry::get('Categories');
    $firstCat = $categories->find()->first();
    $expected = ($query -> count()) + 1;
    $item = $this->Items->newEntity();
    $item->Photo = "Test";
    $item->title = "title";
    $item->category_id = $firstCat->id;
    $item->quantity= 1;
    $this->Items->save($item);
    $this->assertEquals($items->find('all')->count(), $expected);
}
```

Figure 20 Code for Testing Adding Item in Database

In code above, first the total number of available items in database were calculated. so, expected result after successful addition of an item to the database should be equal to the total number of available items plus one. new instance of the item was created and added to database. by using assertEquals() method we identify whether total number of items in database is equal to expected value. If it is equal, then the add method in items controller works properly otherwise there is a problem in this method.

The following code is for testing addition of large number of the items to database and monitor the capability of the application when large number of the items are added to the system. fortunately, application passed this test and 1000 item added to database successfully without any errors.

```
public function test1000ItemAdd(){
    $items = TableRegistry::get('items');
    $query = $items->find('all');
    $categories = TableRegistry::get('Categories');
    $firstCat = $categories->find()->first();
    $expected  = ($query -> count()) + 1000;
    for ($x = 0; $x <= 1000; $x++)
    {$item = $this->Items->newEntity();
    $item->title = "test1000";
    $item->category_id = $firstCat->id;
    $item->quantity= 1;
    $item->Photo = "Test";
    $this->Items->save($item);}
    $this->assertEquals($items->find('all')->count(), $expected);
}
```

Figure 21 Code for Testing Adding 1000 Items to Database

The next test is testing delete function to make sure delete function work properly and item is removed from database

```
public  function testItemDelete() {
    $items = TableRegistry::get('items');
    $query = $items->find('all');
    $last= $items->find()->last();
    $expected = ($query -> count()) - 1;
    $entity = $this->Items->get($last->id);
    $this->Items->delete($entity);
    $this->assertEquals($items->find('all')->count(), $expected);
}
```

Figure 22 Code for Deleting Item from Database

To test delete method the last item in database is identified, then it is removed from database. expected result for deleting should be total number of available item in database mines 1. so by using assertEqual() method the database was tested to make sure the current item is deleted from database.

Following is a snapshot of the unit test result performed on all models. As it is shown, the –testsuite switch is used to only perform the unit tests having 12/12(100%) success.

```
hesam@DESKTOP-NMD7TTH c:\xampp\htdocs\BorrowingSystem
# phpunit --testsuite=unit
PHPUnit 5.7.0 by Sebastian Bergmann and contributors.

............                                                      12 / 12 (100%)

Time: 695 ms, Memory: 10.00MB

OK (12 tests, 12 assertions)
```

Figure 23 Unit Test Result

4.2. Integration Test

After unit testing, integration testing has been performed to test the individual components of the application. for Integration Test CakePHP offers a specialized IntegrationTestCase class, so this class will be inherited by the controller which are going to incorporate integration test.

Integration test in CakePHP simulates an HTTP requests which are handled by application. It tests controller and any model, component and helpers that are involved in handling any given request.

In this project, integration test has been implemented for all controllers to ensure that application works properly but in this section, there is description for an Items controller .to view the code for other controllers please refer to test directory.

As it is shown in the test functions below, it is obvious that each test function is correlated to its counterpart function in the corresponding controller e.g. testIndex is the test function testing the Index function in the corresponding controller and this convention has been used through all testing classes.

testIndex: At this point if we concentrate on testIndex class, there are 2 lines of codes which are almost self-explanatory as *get* function tries to simulate a get request on the /Items/Index route and assertResponseCode tries to evaluate its response code against a certain status code as it is defined in its argument which in this case is 200 where it is a standard http status code stating a successful response. If we try to have a definition for this function, we may define it as following:

Make a request to /Items/index route and if the response is success (200), mark the test as passed otherwise any other response having any other code other than 200 is a failed test.

testView: In this test, the goal is to check an individual item page health while it is being requested by the user. The strategy used here is self-explanatory as well. First, the items table is queried against its first member, then the specific page for that chosen item is requested. At this stage is the page for that specific item exists a 200 will be sent back as the response otherwise if the page does not exist the response will be anything other than 200 which identifies the status of this test.

```php
public function testIndex()
{
  $this->get('/items');
  $this->assertResponseCode(200);
}
public function testView()
{
    $items = TableRegistry::get('items');
    $firstitem = $items->find()->first();
    $this->get('/items/view/' .  $firstitem->id);
    $this->assertResponseCode(200);
}
```

Figure 24 Code for Testing Index and view functions

```php
public function testEdit()
{
    $items = TableRegistry::get('items');
    $firstitem = $items->find()->first();
    $firstitem-> title = 'test title';
    $path = 'items/edit/'.$firstitem->id;
    $data = [
        'id' => $firstitem -> id,
        'title' => 'test title',
        'quantity' => $firstitem->quantity,
        'category_id' => $firstitem->category_id ,
        'photo' => $firstitem->photo,
    ];
    $this->post($path, $data);
    $this->assertResponseCode(302);
}
```

Figure 25 Code for Testing Edit Function

testAdd: In this test, the goal is to check if a new item can be successfully to the database. In this case a reference to a specific existing category is needed as the new item must be added to an existing category. For the sake of having a valid category id a query has been made against the first category in the categories table. After having a valid category id, a new instance on an item has been created using the fetched category id then the newly created item is sent to /items/add route as a post request so the payload of the request is wrapped in the body of the request. It is worth mentioning that the add action hosted at /items/add url will redirect the request maker (browser or any other agent) to another url which is a confirmation page to inform the the request maker of its requested operation. As this redirection only happens in the case of successful operation the 302(standard http redirect code) is used to evaluate the success of request.

testDelete: In this test, the goal is to check if an item can be successfully deleted from database. In this case a reference to a last item is needed. After finding last item in database. by using the last item id then the last item is deleted by using the /items/delete route as a post request so the payload of the request is wrapped in the body of the request. It is worth mentioning that the delete action hosted at /items/delete url will redirect the request maker (browser or any other agent) to another url which is a confirmation page to inform the the request maker of its requested operation. As this redirection only happens in the case of successful operation the 302(standard http redirect code) is used to evaluate the success of request.

```
public function testAdd()
{
    $categories = TableRegistry::get('Categories');
    $firstCat = $categories->find()->first();
      $data = [
        'title' => 'text item',
        'quantity' => 10,
        'category_id' => $firstCat->id ,
        'photo' => 'new photo',
    ];
     $this->post('items/add', $data);
     $this->assertResponseCode(302);
}
```

Figure 26 Code for Testing Add Function

```
public function testDelete()
 {
    $items = TableRegistry::get('items');
    $lastitem = $items->find()->last();
    $this->post('/items/delete/' .  $lastitem->id);
    $this->assertResponseCode(302);
 }
```

Figure 27 Code for Testing Delete Function

Following is a snapshot of the integration test result performed on all controllers. As it is shown, the –testsuite switch is used to only perform the integration tests having 34/34(100%) success.

```
hesam@DESKTOP-NMD7TTH c:\xampp\htdocs\BorrowingSystem
# phpunit --testsuite=integration
PHPUnit 5.7.0 by Sebastian Bergmann and contributors.

..................................                                34 / 34 (100%)

Time: 1.98 seconds, Memory: 20.00MB

OK (34 tests, 134 assertions)
```

Figure 28 Integration Test Result

4.3. System Testing

System testing perform in whole application to evaluate compliance of app to the identified requirements.as first stage Usability testing is performed to find that is it system easy to use and understandable to user and is application meets the project objective, therefore system tested with some users and feedback collected from them. The tested result showed that most of the users are willing to use this application and they are not confused about how to work with app.

Functional testing also performs to identifying any missing function, so in this project after doing it, some missing functions are identified and added to system.

Application also tested with various input and outputs checked to ensure that app work correctly with valid data and it will deny the invalid data.

4.4.A/B Testing

A/B testing is method of comparing two variations of a web page to find out which one performs better. In A/B testing two variations of a webpage are shown to the user, then the one that gets the higher conversion rate, wins [18].

The main goal of every website is to converting their visitors to something else. the rate that websites can achieve this goal is known as conversion rate.

By using A/B testing you can decide about changes in your website through real experiments. The decision made by A/B testing is based on result that is obtained directly from users, the data is not based on prediction, therefore data is more reliable. so, it helps you to decide whether to confirm or discard changes. A/B testing helps to optimize website by increasing the conversion rate. A high conversion rate shows high level of success in website to attract users, this would be also cost effective since you don't pay so much money for traffic acquisition [19].

In this project A/B testing performs to increase the number of successful borrowings which are completed by the borrowing system application. to achieve this goal the second version of the main page of the website that shows the available items to users is developed, this version is known as variation and the original web page is known as control.

The new version is developed based on the idea of items information rearrangement that is shown to users. based on current analysis, there are some weak points identified that may cause to bounce rate in website. the current user interface might be weak to show the items information to the user in a way that attracts user attention, therefore it should be improved to represent information in a better way that makes the user more comfortable with new user interface.

The control version of the web site shows the item information in a smaller space comparing with its variation version as it represents two items in a single row while its corresponding variation version represents one item in a single row, therefore user can see the item information in a larger space. On the other hand, in the control version the total width for item is less but user can see more items information in single page.

For purpose of testing 40 students participated in the survey. the two versions are represented to them and they are asked to surf the website to get feedback if they are comfortable with the provided UI for borrowing procedure. In A/B test, normally 50/50 traffic split between the control page and variation, therefore in this project the control version page is represented to 20 students and variation version is represented to the remaining 20 students.

After all students participated in research, the collected data is analyzed, the result shows that 80 percent of the students make successful booking with variation version while only 50 percent of students tend to book items in control version.

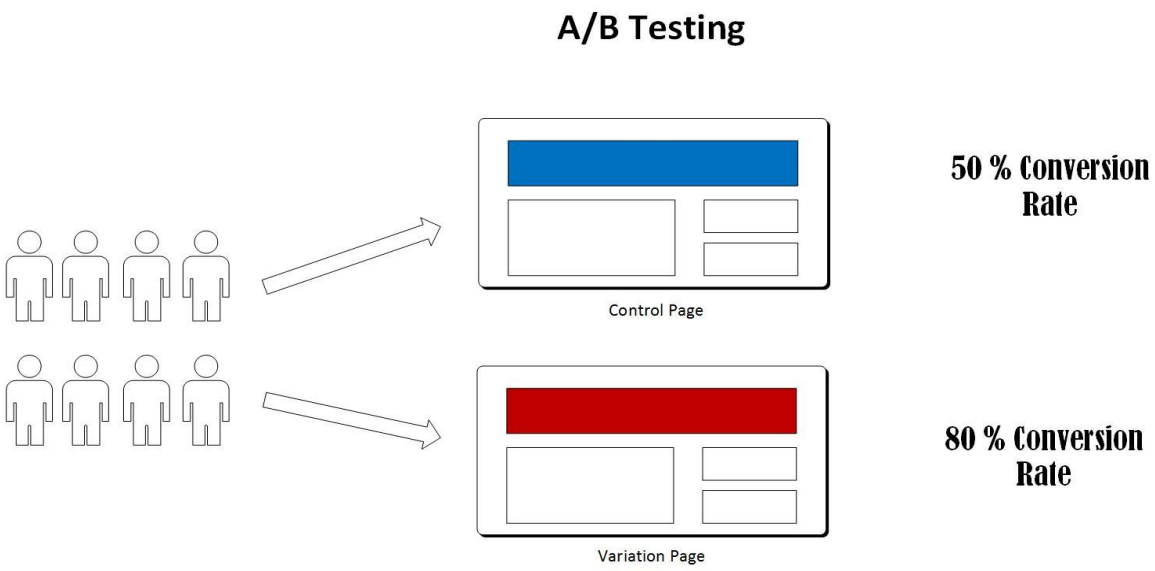


Figure 29 A/B Testing Result

As indicated in figure above, variation version of the website wins the testing, therefore changes have been made in user interface to increase the conversion rate in borrowing system website.

5.Deployment

At this stage when the development is completed, there are the following issues that should be taken care of before the web application may be hosted on the application server which is called the deployment process.

Config/app.php adjustment

All the application wide configurations are kept in the app.php and as we have been developing the application there have been many settings configured for only development time as following:

Debugging messages: the debug variable should be set to false so all the calls for debugging using pr(), debug(), and dd() will be automatically disabled[1].

Caches should be set to flush properly as its default value is to flush every 365 days while this value should be set to 10 seconds when it is serving clients.

There should be generic error views instead of detailed ones as the users do not to know the details of any error and a general error would be more helpful secure [1].

Exception stack should be disabled so nothing about the error on any specific stack will be shown to the users.

Security

There are some security concerns as following that should be addressed in the deployment process:

LDAP integration should be addressed and resolved before the final hosting the application.

Make sure that models and views have proper validation rules.

Check that only your webroot directory is publicly visible, and that your secrets (such as your app salt, and any security keys) are private and unique as well.

Set Document Root :

An important step in the deployment process is setting the document root properly. CakePHP applications should their document root set to the application Webroot as this makes the application and configuration file is accessible through a url. This way we make sure that no file outside the Webroot directory get executed [1].

6.Conclusion

The aim of this project is to provide the modern web application that replaces the old web app in Fh-Kiel which manages borrowing items in university. there are some reviews at first step, which were done in pervious app to identify main problems of system and finding possibility to improve system then stakeholders meeting conducted to gather and identify requirements and their expectations.

To reach project objective of the project that is satisfaction of the stockholders, the modern web technology is used to provide responsive and user friendly system that user feels comfortable when use system.

The coding of the app is clear and easy to understand since MVC framework is used in this project and all project tasks are broken up in smaller jobs. in addition to the documentation it would be helpful to understand whole system.

All expected functional and nonfunctional requirements of the app are successfully implemented in this project that fulfill everything that is needed in borrowing system. After developing app all functions of the app were tested with Phpunit to identify any problem and error. The result of the test shows that application work properly in different test case scenarios.

At the end, this project provides great experience for me in term of the web development, designing, implementing and testing web application in php language which are all new things that I learnt during this project.

7.Bibliography

[1] CakePHP Documention. Retrieved November 10, 2016, from CakePHP, https://book.cakephp.org/3.0/en/index.html

[2] Project Managment. Retrieved January 15, 2017, from Wikipedia, https://en.wikipedia.org/wiki/Project_management

[3] User Interface Design. Retrieved January 20, 2017, from Wikipedia, https://en.wikipedia.org/wiki/User_interface_design

[4] Usability. Retrieved January 15, 2017, from Usability.gov, https://www.usability.gov/whatand-why/user-interface-design.html

[5] PHPUnit. Retrieved February 01, 2017, from PHPUnit, https://phpunit.de/

[6] Unit Test. Retrieved February 01, 2017, from Wikipedia, https://en.wikipedia.org/wiki/Unit_testing

[7] Integration Testing. Retrieved February 02, 2017, from SoftwareTestingFundamentala, http://softwaretestingfundamentals.com/integration-testing/

[8] System Test. Retrieved February 01, 2017, from Tutorial Point, http://www.tutorialspoint.com/software_testing/software_testing_quick_guide.html

[9] Agile Methodology. Retrieved February 02, 2017, from agilemethodology, http://agilemethodology.org/

[10] Tutorialpoint. Retrieved January 10, 2017, from https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

[11] Authentication. Retrieved January 15, 2017, from Search Security, http://searchsecurity.techtarget.com/definition/authentication

[12] Kangasharju, J. (2008). Distributed Systems. (Kangasharju, 2008)

[13] Coulouris, G., Dollimore, J., Kindberg, T. and Blair, G. (2012). *Distributed systems*. 3rd ed. Harlow, England: Addison-Wesley.

[14] Chapter 4: Distributed and Parallel Computing. (2017). Wla.berkeley.edu. Retrieved 9 February 2017, from http://wla.berkeley.edu/~cs61a/fa11/lectures/communication.html

[15] Usman Ahmed, M. (2016). eBay - eCommerce Platform.

[16] eBay Architecture - High Scalability -. (2017). Highscalability.com. Retrieved 1 March 2017, from http://highscalability.com/ebay-architecture

[17] *What is a Distributed Computing System? - Definition from Techopedia*. (2017). *Techopedia.com*. Retrieved 1 February 2017, from https://www.techopedia.com/definition/7/distributed-computing-system

[18] Optimizely: Optimize digital experiences for your customers. (2017). Optimizely.com. Retrieved 9 Feb 2017, from https://www.optimizely.com/ab-testing/

[19] A/B Testing - The Complete Guide. (2017). Vwo.com. Retrieved 1 February 2017, from https://vwo.com/ab-testing/

[20] UNBOUNCE, (2016). The Ultimate Guide to A/B Testing.

[21] From PHP to Node.js - Part. II: Performances. (2017). Soulserv Team. Retrieved 1 March 2017, from http://blog.soulserv.net/from-php-to-nodejs-part-ii-performances/

[22] Wayner, P. (2017). PHP vs. Node.js: An epic battle for developer mind share. InfoWorld. Retrieved 1 March 2017, from http://www.infoworld.com/article/3166109/application-development/php-vs-nodejs-an-epic-battle-for-developer-mind-share.html

[23] an analysis of node js. (2017). Retrieved 1 March 2017, from http://www.cs.fsu.edu/~xyuan/cop4020/written.pdf

[24] MongoDB and MySQL Compared. (2017). MongoDB. Retrieved 12 March 2017, from https://www.mongodb.com/compare/mongodb-mysql

[25] Dipina Damodaran, Shirin Salim, & Surekha Marium Vargese,. (2016). MONGODB VS MYSQL. A COMPARATIVE STUDY OF PERFORMANCE IN SUPER MARKET MANAGEMENT SYSTEM.

8.List of Figure

9.Appendix

```
public function add() {
    $item = $this->Items->newEntity();
    if ($this->request->is('post')) {
        $item = $this->Items->patchEntity($item, $this->request->data);
        if ($this->Items->save($item)) {
            $this->Flash->success(__('The item has been saved.'));

            return $this->redirect(['action' => 'index']);
        } else {
            $this->Flash->error(__('The item could not be saved. Please, try again.'));
        }
    }
    $categories = $this->Items->Categories->find('list', ['limit' => 200]);
    $this->set(compact('item', 'categories'));
    $this->set('_serialize', ['item']);
}
```

Figure 30 Code Snippet for Add New Item

```
public function delete($id = null) {
    $this->request->allowMethod(['post', 'delete']);
    $item = $this->Items->get($id);
    if ($this->Items->delete($item)) {
        $this->Flash->success(__('The item has been deleted.'));
    } else {
        $this->Flash->error(__('The item could not be deleted. Please, try again.'));
    }
    return $this->redirect(['action' => 'index']);
}
```

Figure 31 Code Snippet for Delete Item

```
public function edit($id = null)
{
    $item = $this->Items->get($id, [
        'contain' => []
    ]);
    if ($this->request->is(['patch', 'post', 'put'])) {
        $item = $this->Items->patchEntity($item, $this->request->data);
        if ($this->Items->save($item)) {
            $this->Flash->success(__('The item has been saved.'));

            return $this->redirect(['action' => 'index']);
        } else {
            $this->Flash->error(__('The item could not be saved. Please, try again.'));
        }
    }
    $categories = $this->Items->Categories->find('list', ['limit' => 200]);
    $this->set(compact('item', 'categories'));
    $this->set('_serialize', ['item']);

}
```

Figure 32 Code Snippet for Edit Item

```
  public function userview() {

      $this->viewBuilder()->layout('foruser');
      $q = null;
      if ($this->request->query('title') != null) {
          $q = $this->request->query['title'];
      }
      if ($q != null) {
$items = $this->paginate($this->Items->find()->contain(['Categories'])->where(['items.title' => $q]));
      } else {
          $items = $this->paginate($this->Items->find()->contain(['Categories']));
      }
      $id = $this->Auth->user('id');
      $this->set(compact('items', 'id'));
      $this->set('_serialize', ['items']);
  }
```

Figure 33 Code Snippet for Displaying Items to Users

```
public function delete($id = null) {
    $this->request->allowMethod(['post', 'delete']);
    $loan = $this->Loan->get($id);
    if ($this->Loan->delete($loan)) {
        $this->Flash->success(__('The loan has been deleted.'));
    } else {
        $this->Flash->error(__('The loan could not be deleted. Please, try again.'));
    }

    return $this->redirect(['action' => 'index']);
}

}
```

Figure 34 Code Snippet for Deleting Loan

```
public function add($id = null) {
$this->paginate = [
    'contain' => ['Items', 'Users']
];
$loan = $this->Loan->newEntity();
if ($this->request->is('post')) {
    $h = $this->request->data['item_id'];
    $userId = $this->request->data['user_id'];
    $loan->Return_item = false;
    $loan = $this->Loan->patchEntity($loan, $this->request->data);
    if ($this->Loan->save($loan)) {
        $this->Flash->success(__('The loan has been saved.'));
        $lo = $loan->id;
        $itemsTable = TableRegistry::get('Items');
        $item = $itemsTable->get($h);
        $item->quantity = $item->quantity - 1;
        $itemsTable->save($item);
        $usersTable = TableRegistry::get('Users');
        $user = $usersTable->get($userId);
        $userDataArr = array(
            'userData' => $user,
            'itemData' => $item,
            'loanData' => $loan

            $email = new Email();
            $email->template('confirm', 'default')
                    ->emailFormat('html')
                    ->to($user->email)
                    ->from(['farajihesam166@gmail.com' => 'Fhkiel'])
                    ->subject('Booking Confirmation')
                    ->viewVars($userDataArr)
                    ->send();
            return $this->redirect(['action' => 'index']);
        }
    } else {
        $loan->selectedValue = $id;
        $items = $this->Loan->Items->find('list', ['limit' => 200]);
        $users = $this->Loan->Users->find('list', ['limit' => 200]);
        $this->set(compact('loan', 'items', 'users', 'lo'));
        $this->set('_serialize', ['loan']);
    }
}
```

Figure 35 Code Snippet for Adding New Loan and Sending Confirmation Email

```
public function edit($id = null) {
    $loan = $this->Loan->get($id, [
        'contain' => []
    ]);
    if ($this->request->is(['patch', 'post', 'put'])) {
        $loan = $this->Loan->patchEntity($loan, $this->request->data);
        if ($this->Loan->save($loan)) {
            $this->Flash->success(__('The loan has been saved.'));
            return $this->redirect(['action' => 'index']);
        } else {
            $this->Flash->error(__('The loan could not be saved. Please, try again.'));
        }
    }
    $items = $this->Loan->Items->find('list', ['limit' => 200]);
    $users = $this->Loan->Users->find('list', ['limit' => 200]);
    $this->set(compact('loan', 'items', 'users'));
    $this->set('_serialize', ['loan']);
}
```

Figure 36 Code Snippet for Editing Item

```
public function isAuthorized($user) {
    if ($this->request->action === 'index') {
        return true;
    }
    if (in_array($this->request->action, ['controler' => 'loan', 'action' => 'view'])) {
        $loanId = (int) $this->request->params['pass'][0];
        if ($this->Loan->isOwnedBy($loanId, $user['id'])) {
            return true;
        }
    }



    return parent::isAuthorized($user);
}
```

Figure 37 Code Snippet for Showing Each User Own Borrowed Item

```
public  function returnitem($id = null)
{
  $loan = $this->Loan->get($id, [
        'contain' => []
    ]);
  $x=$loan->item_id;
    if ($loan->Return_item==true)
  {
        $this->Flash->success(__('The loan has allready returned.'));
        return $this->redirect($this->referer());
  }
    if ($this->request->is(['patch', 'post', 'put'])) {
    $loan = $this->Loan->patchEntity($loan, $this->request->data);
    if ($this->Loan->save($loan)) {
    $itemsTable = TableRegistry::get('Items');
    $item = $itemsTable->get($x); // Return article with id 12
    $item->quantity = $item->quantity+1;

    $itemsTable->save($item);
            $this->Flash->success(__('The loan has been saved.'));
            return $this->redirect(['action' => 'index']);
        } else {
            $this->Flash->error(__('The loan could not be saved. Please, try again.'));
        } }
    $items = $this->Loan->Items->find('list', ['limit' => 200]);
    $users = $this->Loan->Users->find('list', ['limit' => 200]);
    $this->set(compact('loan', 'items', 'users'));
    $this->set('_serialize', ['loan']);
```

Figure 38 Code Snippet for Return Item

```
public function reminder() {

    $loan = $this->Loan->find()
            ->where(['Loan.item_id' => 102]);
    foreach ($loan as $it):
        $t = new Time($it->due_date);
        $userId = $it->user_id;
        $itemId = $it->item_id;
        $itemsTable = TableRegistry::get('Items');
        $item = $itemsTable->get($itemId);
        $usersTable = TableRegistry::get('Users');
        $user = $usersTable->get($userId);
        $userDataArr = array(
            'userData' => $user,
            'itemData' => $item,
            'loanData' => $it
        );

        if ($t->isWithinNext(1) and $it->Return_item == false) {
            $email = new Email();
            $email->template('reminder', 'default')
                    ->emailFormat('html')
                    ->to($user->email)
                    ->from(['farajihesam166@gmail.com' => 'Fhkiel'])
                    //->from('farajihesam166@gmail.com')
                    ->subject('Booking Confirmation')
                    ->viewVars($userDataArr)
                    ->send();
        }
    endforeach;
    $this->set(compact('times'));
    $this->set('_serialize', ['times']);
}
```

Figure 39 Code Snippet for Send Reminder