AUTOMATED LIBRARY MANAGEMENT SYSTEM

THIRD YEAR DESIGN PROJECT

NSS COLLEGE OF ENGINEERING

Roll No.	Name	University Registration Number
5	Abhijith M V	NSS16EE004
6	Abhilash P	NSS16EE005
7	Abhinav R	NSS16EE006
8	Abhishek P	NSS16EE007

ACKNOWLEDGEMENTS

We would like to express our heartfelt gratitude to our guides Professor Jeema K Cherian, Dr. Priya G Das and Professor Neethi S.Pillai for their continued support and guidance throughout this venture. We would also like to thank our HOD , Dr.Devi V for giving us the opportunity to do this project. We also thank our Principal Dr. T. Sudha , without whose support this project would not have been possible.

ABSTRACT

This project proposes an automated system for the management of a library, the EEE department library of NSS College of Engineering specifically. The design targets reduced human interaction in the day-to-day working of the library. It will make it flexible in adapting to the changes in information exchange. We are slowly moving towards a paper free world and e-libraries are just around the corner. In a time when the average teenager is spending more time online, taking the library experience online seems sensible. The design will include a web application for imparting information. It will also contain systems to monitor the usage of the library, assess the health of the books and prevent theft. Reducing the amount of human interaction will help increase the fluidity of the system.

CONTENTS

CHAPTER.....PAGE

CHAPTER1
INTRODUCTION

Library interest in automated circulation control is, in large part, based on a long-standing awareness of the problems inherent in manual circulation systems. These problems include labor-intensive and time-consuming recordkeeping work routines, inaccuracy, high personnel turnover, an inability to generate statistics about circulation activity, and the lack of an interface between circulation files and other library files which contain much the same bibliographic data. Circulation control is one of the most widely automated library operations, and it is often the first and simplest activity to be automated in a given library, possibly because circulation control systems bear an obvious resemblance to inventory management, retail charge card operations, and other transaction processing activities which have been successfully automated in general business applications.

An analysis of the working of the *EEE department library* of *NSSCE* , brought ot the following key features about its working :

i. All books in the library are given a unique, 5 digit identification. Of the five digits, the first two signify the section of the library to which the book belongs.
ii. Any user entering the library has to make a record of the time of his/her arrival and departure in the *entry register.*
iii. The user issues the books himself , by making an entry under his name in the *issue register.*
iv. When the user returns the book, the teacher in-charge verifies the health of the books and makes a note of the safe return of the book in the *issue register.*
v. A user can keep a book issued under his name for a 14 days. He/She can reissue the book before this period ends and keep it for an additional period of 14 days.
vi. Return of books past their due date leads to imposition of fines on the user. The total fine in a users name is collected at the end of the semester.
vii. There is no infromation retrieval system for the users to search for the availability of books.

The analysis of the library circulation process helped us identify certain key areas which could be automated. These are:

i. The book issue and return process.
ii. The calculation of fine.
iii. The entry register.
iv. Infromation retrieval system

Another fact we noticed about the library is a complete absence from cyber space. In a time when the average person is spending increasing amounts of time online, it is imperative for the user community that the library goes online as well. This will enable the library to give the students a wide veriety of new services.

i. An online digitral library can be created which can considerably increase the number of books in the catalogue. This can help bypass the bottleneck created by the space scarcity. This will also help the more digital savvy among students make better use of the library.
ii. An online information system can help the users check for the availability of a particular book.
iii. It can also be used for passing information to the users.

Our design addresses all the above mentioned problems and uses them as an opportunity to build a robust and flexible management system for the *department library.*

CHAPTER 2
TOOL BOX

Before delving into the design we would like to familiarize the reader with the tools we have used in our design.

We have used HTML5, CSS3, JavaScript and Bootstrap to build the front end of the website. The backend was created using the opensource Django web application framework written in Python. We have used Django version 2.0.2 and Python version 3.6.1.

The web app is hosted on *pythonAnywhere.com* which is a hosting service for web applications written in Python and Python frameworks like Django and Flask.

We plan to use the Arduino Uno R3 microcontroller platform, ESP8266 wifi module and RFID tagging during the second phase of the project to create the library management hardware.

2.1 PYTHON AND DJANGO

Python is an interpreted, high-level , general-purpose programming language , Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. The reference verion of Python , Cpython , is open source and has a community-based development model. It has vibrant developer community and following.

Django is an open source web development framework written in python. A web framework is a set of components that helps you to develop websites faster and easier. It comes with an object-relational mapper in which you describe your database layout in Python code. More than that, the data-model syntax provides many rich ways of representing your models. The migrate command glances over all available models and builds tables in your database for whichever ones that don't exist. Its user authentication system provides a safe way to manage user accounts and passwords alongside ensuring developers stray away from making common mistakes such as: cross-site scripting, cross-site request, click jacking and forgery.

Some other webdevelopment frameworks similar to Django are Flask(Python), Node.js(JavaScript), Express.js(JavaScript) etc.

2.2 SQLite

SQLite is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages.

2.3 ARDUINO UNO R3

The Arduino Uno R3 is a microcontroller board based on theAtmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16MHzcrystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. It has 32Kb of flash memory and ADC on each of its analog pins with 10-bit resolution.

The Arduino IDE , provided by Arduino.cc, makes the job of programming the board easy. The board is programmed using a variant of C written for arduino.

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol

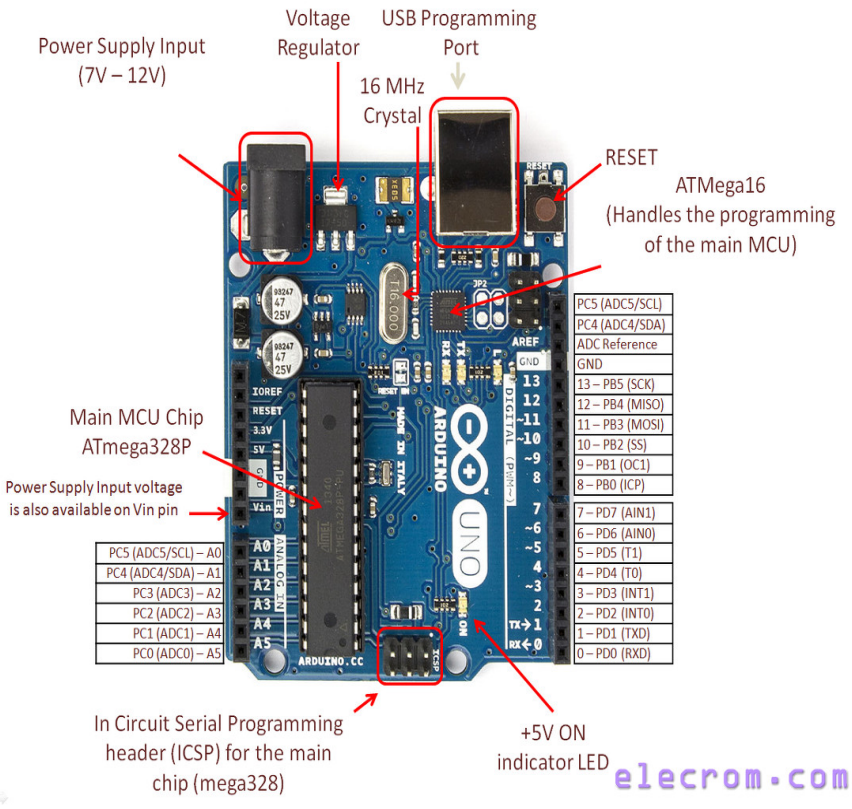

Illustration 1: Arduino Pinout

2.4 ESP8266 WiFi MODULE

The Arduino Uno R3 does not have inbuilt wifi. In order to give our application the ability ro connect to a network , we add an ESP8266 WiFi module. As an aletrnative we could have used a NodeMCU microcontroller board which comes with built-in ESP8266. But it is not as robust as the arduino.

The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by Shanghai-based Chinese manufacturer Espressif Systems.

The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at the time there was almost no English-language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module, which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.

The ESP8285 is an ESP8266 with 1 MiB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi.

The successor to these microcontroller chips is the ESP32.

FEATURES:

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 Mhz.
- Memory:
    - 32 KiB instruction RAM
    - 32 KiB instruction cache RAM
    - 80 KiB user-data RAM
    - 16 KiB ETS system-data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
- IEEE 802.11 b/g/n Wi-Fi
    - Integrated TR switch, balun, LNA, power amplifier and matching network
    - WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins
- SPI
- I²C (software implementation)
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit ADC (successive approximation ADC)

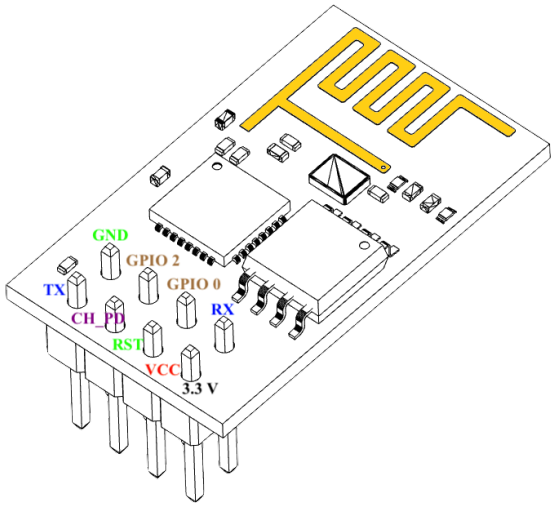

Illustration 2: ESP8266 Pinout

2.5 RFID

Radio-Frequency Identification (RFID) is the use of radio waves to read and capture information stored on a tag attached to an object. A tag can be read from up to several feet away and does not need to be within direct line-of-sight of the reader to be tracked.

A RFID system is made up of two parts: a tag or label and a reader. RFID tags or labels are embedded with a transmitter and a receiver. The RFID component on the tags have two parts: a microchip that stores and processes information, and an antenna to receive and transmit a signal. The tag contains the specific serial number for one specific object.

To read the information encoded on a tag, a two-way radio transmitter-receiver called an interrogator or reader emits a signal to the tag using an antenna. The tag responds with the information written in its memory bank. The interrogator will then transmit the read results to an RFID computer program.

There are two types of RFID tags: passive and battery powered. A passive RFID tag will use the interrogator's radio wave energy to relay its stored information back to the interrogator. A batter powered RFID tag is embedded with a small battery that powers the relay of information.

In a retail setting, RFID tags may be attached to articles of clothing. When an inventory associate uses a handheld RFID reader to scan a shelf of jeans, the associate is able to differentiate between two pairs of identical jeans based upon the information stored on the RFID tag. Each pair will have its own serial number.

With one pass of the handheld RFID reader, the associate can not only find a specific pair, but they can tell how many of each pair are on the shelf and which pairs need to be replenished. The associate can learn all of this information without having to scan each individual item.

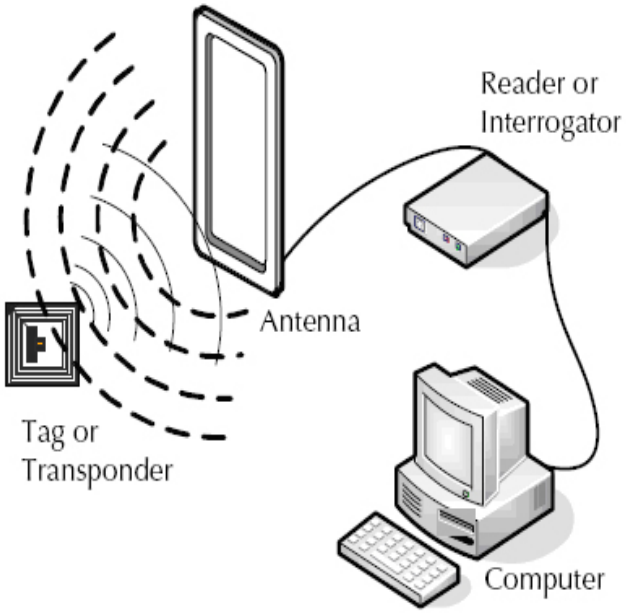

Illustration 3: An RFID system

CHAPTER 3

DESIGN

The automated library management system has two main components:

i. The Software: It is a web application written in the Django web development framework and hosted on *pythonanywhere.com*.
ii. The hard ware: It is an Arduino based system that takes care of the book circulation process

The design of the Automated Library Management System is devided into two phases. The first phase deals with the design, development and production of a working website for the library. The second phase deals with the design, simulation, implementation and testing of the hardware.

3.1 PHASE 1: WEB APPLICATION

The web application is written in Django. It uses a SQLite databased that is embedded inside the application, unlike MySQL or Postgres which runs as a service on the server. The site is hosted on *pythonanywhere.com.*

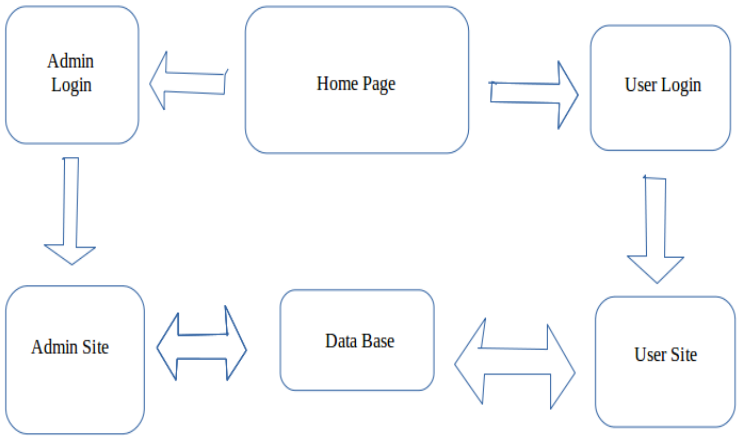

Illustration 4: Site Model

The site model (Illustration 4) gives a graphical view of the relationship between different elements of the site. The home page is a landing page from where the user or admin can jump to their particular site. The admin site contains a built-in django interface which gives the admin power over the different database objects. The user site gives the user information about the books he is currently holding, the outstanding fines againt his name etc. Both the sites communicate with the inbuilt SQLite database through functions called views.

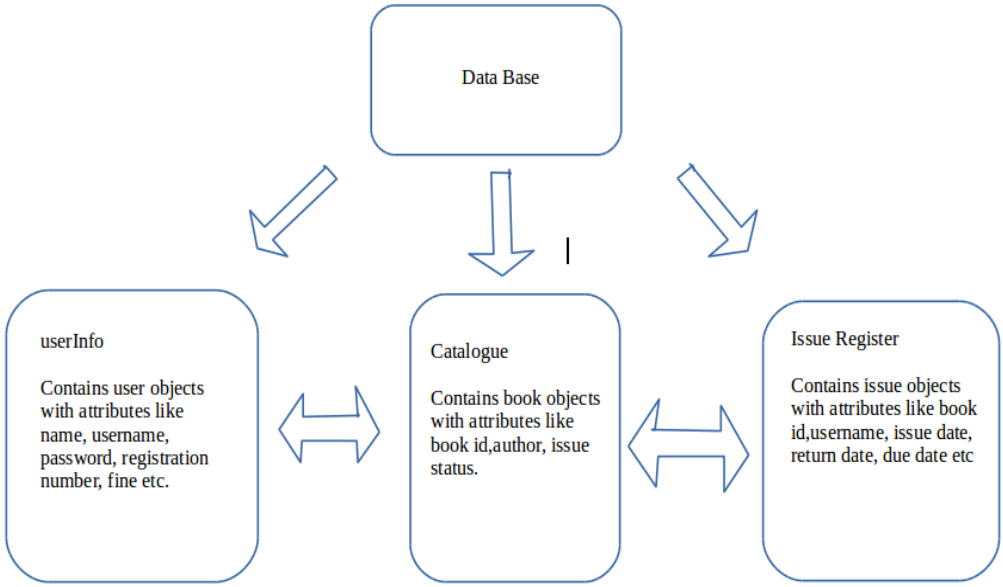

Illustration 5: Database Model

In Django the heavylifting of database creation and querying is done quite easily by creating instance of the django *models.Model* class in the app *models.py* file. By doing so the site can be migrated from one database to the other with relative ease.

The data base model is given in illustration 5 . There are three relational tables in the database:

i. Userinfo:
   . This class defines the user objects. Each user is given attributes like their username, password, registration number, semester ,batch etc. Instances of this class represent the users.

ii. Catalogue:
   . This class defines the collection of books in the library, with each book being an instance of the catalogue object. Each book has attributes like book_id, issue status, current user, author, book name etc.

iii. Issue Register:
   . This class defines a database table similar to the *issue register* used in the library. The objects of this class have attributes like the id of the book, the id of the person who issued the book, the issue date , the due date , the date of return etc.

Using a feature called template tagging , the amount of repetitive code in the software can be considerably reduced. Template tagging allows one to use html pages like objects, with one page able to inherit part of its structure from anothe page. All the functionality of the app can be mapped to one of the many functions called views in the apps views.py file.

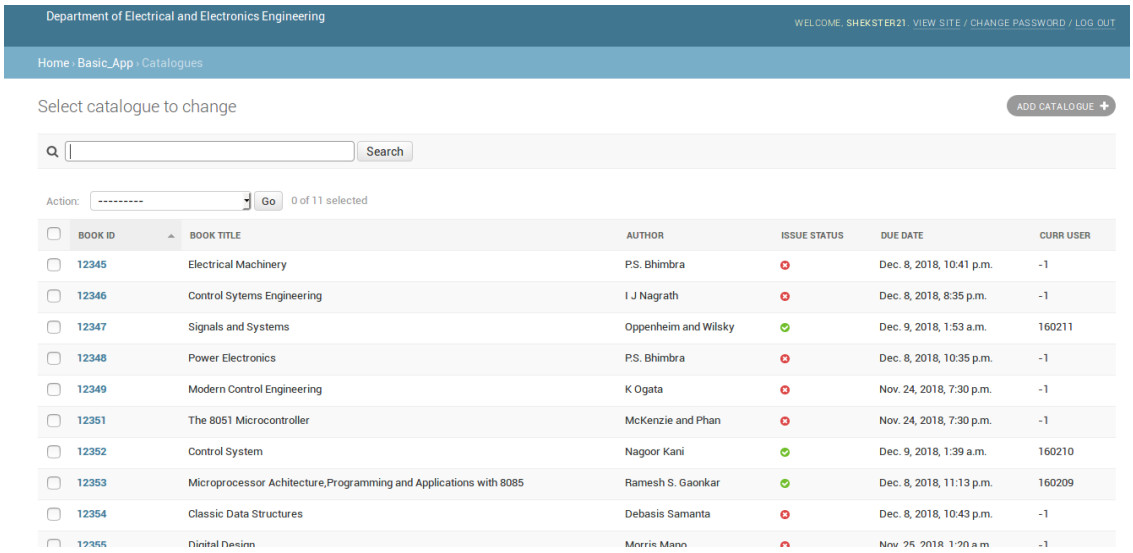

Illustration 6: A view of the admin interface

The homepage of the user gives the information regarding the books they are holding, their due date, the total amount of fine in their name.

Illustration 7: View of the Users home page

3.2 PHASE 2: HARDWARE

An Arduino Uno based take care of the based sytem is used for taking care of the book circulation process of the library. The circuit is given below along with an explanation of its working.

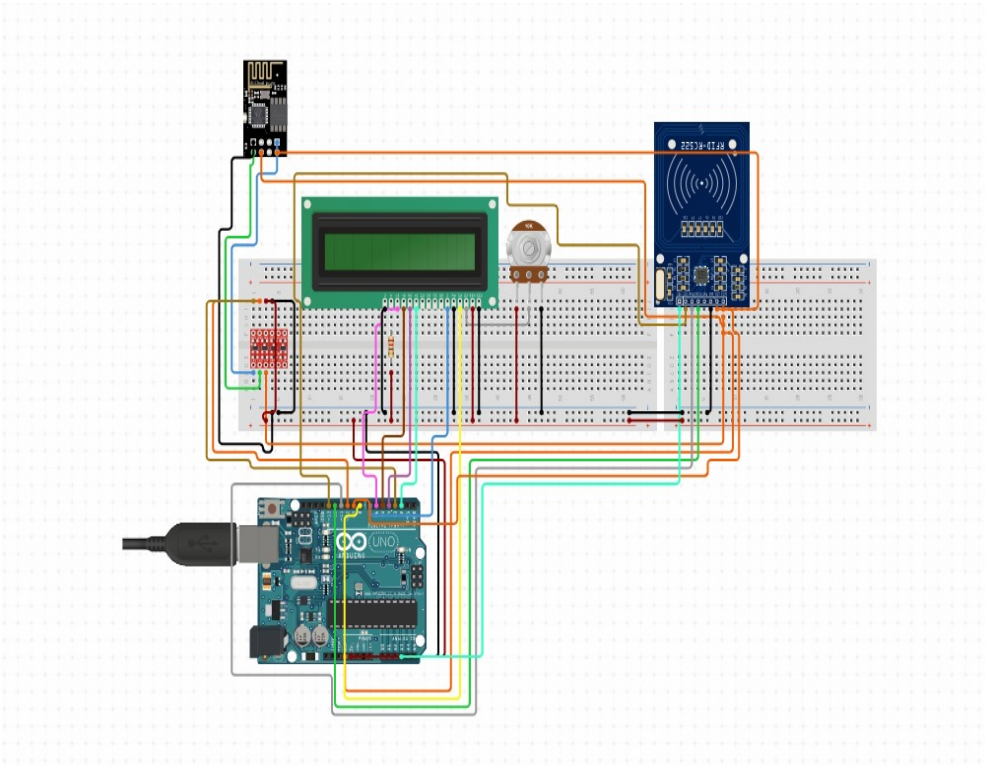

Illustration 8: Hardware

Working:

i. Every book and identity card has an RF-id Sticker attached to it. If a user wants to take a book, he needs to tap his identity card and the RF-id part of the book, with the RF-id reader.
ii. Each RF-id contains the data about the book in a matric format.
iii. When it is tapped, the RF-id reader reads the data. This data/serial number is send to the Arduino along with the data of identity card.
iv. The user name and book details are shown in the display attached to the Arduino. It also notifies the user about the success or failure of the issue or return process.
v. The Arduino is connected with a Wi-Fi module, which is connected to a server.
vi. Socket programming library in C is used to code the microcontroller to communicate with the web server.
vii. The web site is hosted in *pythonanywhere*.com, which provides a set of security features to the websites. The hardware is coded according to these restrictions.

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

At the time of preparation of this report, a working website and hardware for communication with the website has been created and found to be functional. The project has helped the authors gain experience in design of complex hardware and software systems. The opportunity to solve a real life problem was nurturing.

The following features could be added to the design in the future:

i. Hardware system to check the health of returned books.
ii. Theft prevention system.
iii. Integration of a digital library into the website.
iv. Addition of dynamic learning content.
v. A blog for students to share information and knowledge.

REFERENCES

The following sources have been used during the design and implementation of this process:

1) Saffady, William. 1989. “Library Automation : An Overview”.*Library Trends* vol.37. Winter 1989.
2) Sadanand Y. Bansode, Shamin Pariera. 2008. “ A Survey of Library Automation in College Libraries”. University of Nebraska-Lincoln.
3) Saenz, Jaime. 1975. “Computerised Inventory Mangement System”. University of California-Northridge.
4) “The Definitive Guide to Django: Web Development Done Right”.2007.Adrian Holovaty and Jacob Kaplan-Moss. *Apress.*
5) “Lightweight Django: Using REST, WebSockets, and Backbone”.2014.Julia Elman and Mark Lavin.*O’Reilly.*
6) “Arduino Cookbook”. 2011.Michael Margolis.*O’Reilly.*
7) The official documentation of django - *https://docs.djangoproject.com/en/2.1/*