Smart Library Management System Using RFID

B.E. Dissertation

Submitted in partial fulfillment of the requirement of

University of Mumbai

For the Degree of

Bachelor of Engineering
(Electronics & Telecommunication Engineering)

by

Ansari Khalid	**15DET51**
Shaikh Mohammed Sakib	**15DET78**
Inamdar Mohammad Arbaz	**15DET93**
Shaikh Sohail	**15DET79**

Under the guidance of

Prof. Mazhar Malagi

Department of Electronics and Telecommunication Engineering
Anjuman-I-Islam's Kalsekar Technical Campus,
Sector 16, New Panvel , Navi Mumbai
(Affiliated to University of Mumbai)
April 2017

Anjuman-I-Islam's

Kalsekar Technical Campus

(Affiliated to the University of Mumbai)

Plot 2 and 3,Sector 16, Khandagaon, Near Thana Naka, New Panvel, Navi Mumbai 410206.

Certificate

This is to certify that, the dissertation titled

"Smart Library Management System Using RFID "

is a bonafide work done by

Ansari Khalid (15DET51)
Shaikh Mohammed Sakib (15DET78)
Inamdar Mohammad Arbaz (15DET93)
Shaikh Sohail (15DET79)

and is submitted in the partial fulfillment of the requirement for the degree of

Bachelor of Engineering
in
Electronics and Telecomunication Engineering
to the
University of Mumbai.

Guide	Project Coordinator
Head of Department	Principal

Certificate of Approval by Examiners

This is to certify that the dissertation entitled **"Smart Library Management System Using RFID"** is a bonafide work done by **Ansari Khalid Shaikh Mohammed Sakib Shaikh Sohail Inamdar Mohammad Arbaz** under the guidance of **Mr. Mazhar Malagi**. This dissertation has been approved for the award of **Bachelor's Degree in Electronics and Telecommunication Engineering**, University of Mumbai.

Examiners:

Name:	Name:

Acknowledgments

We are highly grateful to the Prof. Afzal Shaikh, HOD of Electronic and Telecommunication Department, Kalsekar Technical Campus (New Panvel), for providing this opportunity to carry out the project. We would like to express our gratitude to other faculty members of Electronics and Telecommunication Engineering Department for providing academic inputs, guidance and encouragement throughout this period. We would like to express our deep sense of gratitude and thank to prof. Mazhar Malagi, for the wise council and able guidance it would have not been possible to carry out this project in this manner. Finally, we express our indebtedness to all who have directly or indirectly contributed to the successful completion of this project.

Abstract

Applicability of Radio Frequency Identification (RFID) system which is a new generation of Auto Identification and Data collection technology in a future Smart Library Management System is presented in this paper. It helps to automate business processes and allows identification of large number of tagged objects like books, using radio waves. In existing system barcode and token card system were used. Barcodes have no read/write capabilities; they do not contain any added information such as expiry date etc. and it needs line of sight, less security and it also can easily damaged. By using token card system, they are very labor intensive and work process for the librarians was more. By considering the above demerits in the existing systems, the proposed Smart RFID system , which is a wireless non-contact system that uses radio frequency to transfer data from a tag attached to an object, for the purpose of automatic identification and tracking. RFID doesnt need the line of sight, it remove manual book keeping of records, improved utilization of resources like manpower, infrastructure etc. Also less time consumption as line of sight and manual interactions are not needed for RFID Tag reading. RFID based Library Management system would helps to allow fast transaction flow for the library and will prove immediate and long term benefits to library in traceability and security.

List of Figures

Contents

Chapter 1

Introduction

1.1 Library Management System:

A library management system (LMS) can be considered as an enterprise resource planning (ERP) system for a library. It is formed from a suite of integrated functions to manage a diverse range of processes within a library. These modules' typically include: cataloging (classifying and indexing materials), acquisitions (ordering, receiving, and invoicing materials), circulation (lending materials to users and receiving them back), serials (tracking journal, magazine and newspaper holdings), OPAC ('Online Public Access Catalogue'–the public interface for users).

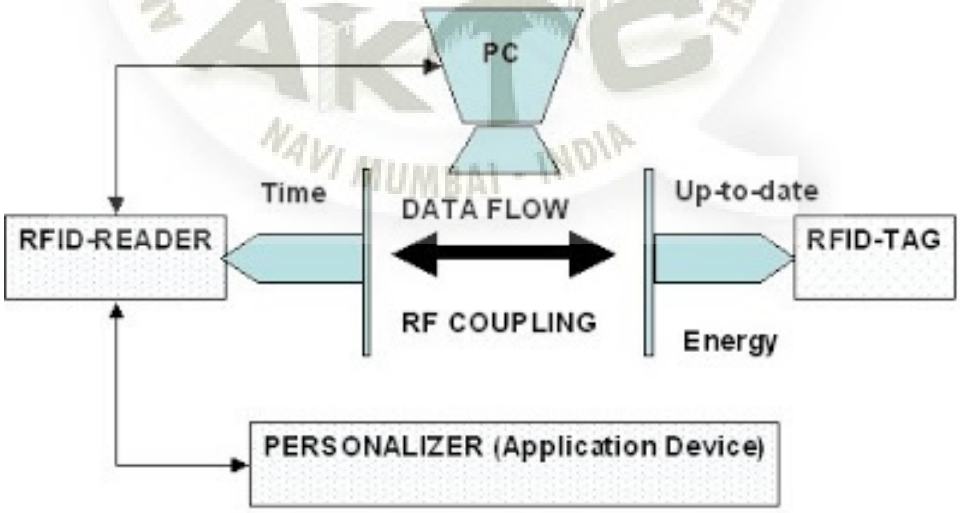

Figure 1.1: Items personalised model for RFID system

1.2 RFID:

It is the wireless non contact system that uses radio frequency EM waves to transfer data from a tag attached to an object, for automatic identification and tracking. A Radio-Frequency Identification system has three parts that are a scanning antenna ,a transceiver with a decoder to interpret the data, a transponder - the RFID tag - that has been programmed with information. The scanning antenna puts out radio-frequency signals in a relatively short range. The RF radiation provides a means of communicating with the transponder (the RFID tag) and provides the RFID tag with the energy to communicate (in the case of passive RFID tags).The scanning antennas can be permanently affixed to a surface, handheld antennas are also available. They can take whatever shape you need; for example, you could build them into a door frame to accept data from persons or objects passing through. When an RFID tag passes through the field of the scanning antenna, it detects the activation signal from the antenna. That "wakes up" the RFID chip, and it transmits the information on its microchip to be picked up by the scanning antenna. The RF low frequency range 120-150 KHz is used for the data transmission.

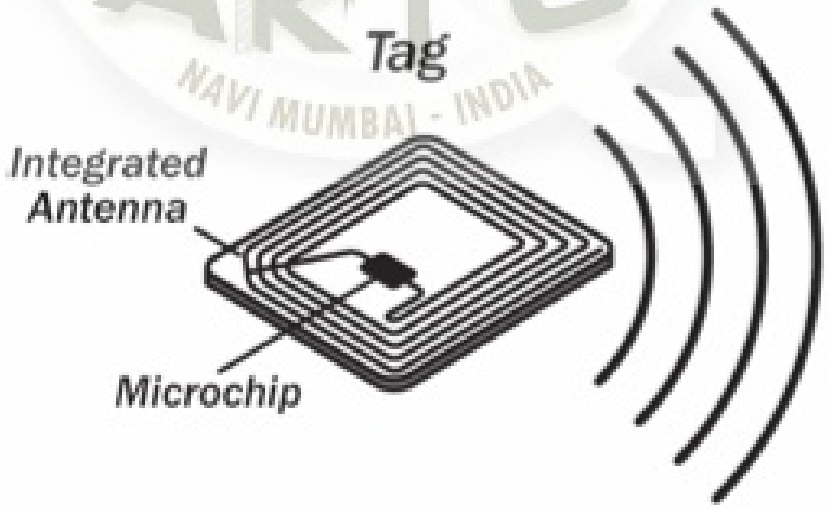

Figure 1.2: RFID tag

Chapter 2

Literature Survey

2.1 "AN IOT BASED SECURED SMART LIBRARY SYSTEM WITH NFC BASED BOOK TRACKING"

A lot of attention. But the application of the internet technology in library management is at its infancy. In a library, books are arranged in shelves based on their classification of subject. There is a lot of movement of these books in and out of the library and so there is every possibility that these books get misplaced. When such misplacement of books occurs, it becomes a tall order for both the librarian and the user to search for the books. Though the OPAC gives the rack and the access number of the books, it is always an uphill task to even find the rack in huge libraries. The scenario becomes worse if the books are misplaced. So a system that could lead the user directly to the place where the book is placed would come in very handy and it also saves a lot of time. It also helps in finding the misplaced books. a connected library system where the user can utilize his mobile phone to connect to the library system and also find the position of the book through a local positioning system would prove to be very useful.

2.2 Conclusion

The internet every day, the IoT is definitely a promising technology for the future. In this paper, we have exploited IoT and mobile technologies for easy and efficient library

management. The major goal of this proposed work is to reduce the burden of the library user to track a book and to fetch it from its location. Here we have used the local positioning system and embedded tags on the book to communicate with each other and with the user's smartphone. With much ease, the user can interact with the library server to check the book more accurately.

2.3 Smart Library Management System using RFID

Library Management System: A library management system (LMS) can be considered as an enterprise resource planning (ERP) system for a library. It is formed from a suite of integrated functions to manage a diverse range of processes within a library. These modules' typically include: cataloging (classifying and indexing materials), acquisitions (ordering, receiving, and invoicing materials), circulation (lending materials to users and receiving them back), serials (tracking journal, magazine and newspaper holdings), OPAC ('Online Public Access Catalogue'–the public interface for users).

RFID: It is the wireless non contact system that uses radio frequency EM waves to transfer data from a tag attached to an object, for automatic identification and tracking. A Radio-Frequency Identification system has three parts that are a scanning antenna ,a transceiver with a decoder to interpret the data, a transponder - the RFID tag - that has been programmed with information. The scanning antenna puts out radio-frequency signals in a relatively short range. The RF radiation provides a means of communicating with the transponder (the RFID tag) and provides the RFID tag with the energy to communicate (in the case of passive RFID tags).The scanning antennas can be permanently affixed to a surface, handheld antennas are also available. They can take whatever shape you need; for example, you could build them into a door frame to accept data from persons or objects passing through. When an RFID tag passes through the field of the scanning antenna, it detects the activation signal from the antenna. That "wakes up" the RFID chip, and it transmits the information on its microchip to be picked up by the scanning antenna. The RF low frequency range 120- 150 KHz is used for the data transmission.

2.4 Conclusion

RFID implementation in libraries has been discussed. The whole system was designed to overcome the disadvantages of barcode systems and thus demonstrated. The entire project was planned to reduce the need of skilled librarians. Though the system is more expensive than the barcode systems, security is ensured and is more efficient. RFID technology is also applicable in various fields like: Asset tracking, people tracking, healthcare, animal tracking, document tracking, object tracking in stores, building access control, airline

baggage tracking and toll collection at toll booths

Chapter 3

Problem Statement

3.1 Problem Statement

This project idea was taken up with a vision to create a Smart Library Management System using RFID to minimize the requirement of a librarian.Here is a list of some features of Library Management System which AmpleTrails offer:

Keep record of different categories like; Books, Journals, Newspapers, Magazines, etc. Classify the books subject wise. Easy way to enter new books. Keep record of complete information of a book like; Book name, Author name, Publishers name, Date/ Year of publication, Cost of the book, Book purchasing date/ Bill no. Easy way to make a check-out. Easy way to make a check-in. Automatic fine calculation for late returns. Different criteria for searching a book. Different kind of reports like; total no. of books, no. of issued books, no. of journals, etc. Easy way to know how many books are issued to a particular student. Easy way to know the status of a book. Event calendar for librarian to remember their dates. My Notes section for librarian to write any note. Online access for registered user to see the status of their books.

3.2 Proposed Design

Tagging: Tag is the most important link in any RFID system. It has the ability to store information relating to the specific item to which they are attached, rewrite again without any requirement for contact or line of sight. Data within a tag may provide identification for an item, proof of ownership, original storage location, loan status and history. RFID

tags have been specifically designed to be affixed into library media, including books, CDs, DVDs and tapes. The role of the librarian is to classify the books into groups and paste the RFID tags on them. These paper-like tags helps in tracking the books within the range of the reader.

Figure 3.1: Proposed Library Layout

Check in/out service: The counter station is a staff assisted station on services such as loan, return, tagging, sorting and etc. The patron approaches the counter to borrow or return the book. First the patron is supposed to identify themselves using the tags provided to them. The staff at the counter then uses a reader to read the tags to make an entry in the central database. In case of book return, the staff collects the book and reads the tag. If the book is returned beyond the due date, fine is collected from the patron.

Self check in/out service: The system basically consists of a computer interfaced with a RFID reader, plus special software for personal identification, book and other media handling and circulation. After identifying the patron with a library ID card, a RFID card- containing the patron details and their ID, the patron is asked to choose the next action (check-out or check in of one or more books). After choosing check-out , the patron puts the book(s) in front of the RFID reader and the display will show the book title, author name and its ID number (other optional information can be shown if desired) which have been checked out .It displays the date before which the book is to be returned. Where as in check in, the patron shows the book(s) in front of the RFID reader and the same will be displayed as in check out. Besides, if there are delays in the return of book(s), the fine amount will be displayed.

Shelf Management: Shelf management includes locating and identifying items on

the shelves as an easy task for librarians. It comprises basically of a scanner and a base station. The system is designed to cover three main requirements: Search for individual books requested, Inventory check of the whole library stock, Search for books which are miss-helved.

Chapter 4

Technical Details

4.1 Methodology

The Initial Setup
Whenever a book is acquired by the library, an RFID tags are placed into the books with the relevant information like, call number, author name, and book number, etc. The detailed information regarding the book is also stored in the computer database. The computer database also stores all information for individual users (users) of the library. Each user is supplied with registered RFID cards. These cards carry identification data and details like: address, roll number, and telephone no. etc. for each user.

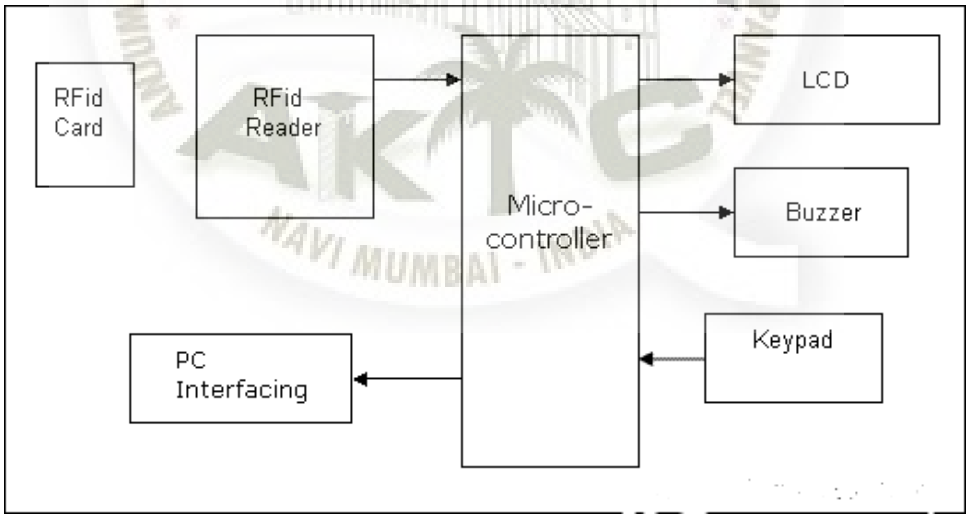

Figure 4.1: Cloud Based Intelligent IoT Framework

There is an administrator with special privileges who has a unique master password controlling the GUI of the RFID SLMS system. As soon as he powers on the system, the first screen displays the LOGIN dialogue box. First he will need to scan his ID card in front of the RFID reader and then entering the corresponding password to enable the system for further usage. When a user wants to return books, he simply places the books again in front of the RFID connected with the controller.

4.2 Project Requirements

4.2.1 Software Requirements

4.2.1.1 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows superfamily of operating systems, as well as web sites, web applications and web services. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code. Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every levelincluding adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer). Visual Studio supports different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++and C++/CLI (via Visual C++), VB.NET (via Visual Basic .NET), C sharp (via Visual C sharp), and F sharp (as of Visual Studio 2010). Support for other languages such as M, Python, and Ruby among others is available via language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Individual

language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J sharp, Visual C sharp, and Visual C++. Microsoft provides "Express" editions of its Visual Studio at no cost. Commercial versions of Visual Studio along with select past versions are available for free to students via Microsoft's DreamSpark program.

4.2.1.2 Visual Studio 2017

Unparalleled productivity for any dev, any app, and any platform. Use Visual Studio 2017 to develop apps for Android, iOS, Windows, Linux, web, and cloud. Code fast, debug and diagnose with ease, test often, and release with confidence. You can also extend and customize Visual Studio by building your own extensions

4.2.1.3 Architecture

Visual Studio does not support any programming language, solution or tool intrinsically, instead it allows the plugging of functionality coded as a VSPackage. When installed, the functionality is available as a Service. The IDE provides three services: SVsSolution, which provides the ability to enumerate projects and solutions; SVsUIShell, which provides windowing and UI functionality (including tabs, toolbars and tool windows); and SVsShell, which deals with registration of VSPackages. In addition, the IDE is also responsible for coordinating and enabling communication between services All editors, designers, project types and other tools are implemented as VSPackages. Visual Studio uses COM to access the VSPackages. The Visual Studio SDK also includes the Managed Package Framework (MPF), which is a set of managed wrappers around the COM-interfaces that allow the Packages to be written in any CLI compliant language. However, MPF does not provide all the functionality exposed by the Visual Studio COM interfaces. The

services can then be consumed for creation of other packages, which add functionality to the Visual Studio IDE. Support for programming languages is added by using a specific VSPackage called a Language Service. A language service defines various interfaces which the VSPackage implementation can implement to add support for various functionalities. Functionalities that can be added this way include syntax coloring, statement completion, brace matching, parameter information tooltips, member lists and error markers for background compilation. If the interface is implemented, the functionality will be available for the language. Language services are to be implemented on a per-language basis. The implementations can reuse code from the parser or the compiler for the language. Language services can be implemented either in native code or managed code. For native code, either the native COM interfaces or the Babel Framework (part of Visual Studio SDK) can be used. For managed code, the MPF includes wrappers for writing managed language services. Visual Studio does not include any source control support built in but it defines two alternative ways for source control systems to integrate with the IDE. A Source Control VSPackage can provide its own customised user interface. In contrast, a source control plugin using the MSSCCI (Microsoft Source Code Control Interface) provides a set of functions that are used to implement various source control functionality, with a standard Visual Studio user interface. MSSCCI was first used to integrate Visual SourceSafewith Visual Studio 6.0 but was later opened up via the Visual Studio SDK. Visual Studio .NET 2002 used MSSCCI 1.1, and Visual Studio .NET 2003 used MSSCCI 1.2. Visual Studio 2005, 2008 and 2010 use MSSCCI Version 1.3, which adds support for rename and delete propagation as well as asynchronous opening. Visual Studio supports running multiple instances of the environment (each with its own set of VSPackages). The instances use different registry hives (see MSDN's definition of the term "registry hive"

in the sense used here) to store their configuration state and are differentiated by their AppId (Application ID). The instances are launched by an AppId-specific .exe that selects the AppId, sets the root hive and launches the IDE. VSPackages registered for one AppId are integrated with other VSPackages for that AppId. The various product editions of Visual Studio are created using the different AppIds. The Visual Studio Express edition products are installed with their own AppIds, but the Standard, Professional and Team Suite products share the same AppId. Consequently, one can install the Express editions side-by-side with other editions, unlike the other editions which update the same installation. The professional edition includes a superset of the VSPackages in the standard edition and the team suite includes a superset of the VSPackages in both other editions. The AppId system is leveraged by the Visual Studio Shell in Visual Studio 2008.

4.2.1.4 Features

Code editor

Like any other IDE, it includes a code editor that supports syntax highlighting and code completion using IntelliSense for not only variables, functions and methods but also language constructs like loops and queries. IntelliSense is supported for the included languages, as well as for XML and for Cascading Style Sheets and JavaScript when developing web sites and web applications. Autocomplete suggestions are popped up in a modeless list box, overlaid on top of the code editor. In Visual Studio 2008 onwards, it can be made temporarily semi-transparent to see the code obstructed by it. The code editor is used for all supported languages. The Visual Studio code editor also supports setting bookmarks in code for quick navigation. Other navigational aids include collapsing code blocks and incremental search, in addition to normal text search and regex search. The code edi-

tor also includes a multi-item clipboard and a task list. The code editor supports code snippets, which are saved templates for repetitive code and can be inserted into code and customized for the project being worked on. A management tool for code snippets is built in as well. These tools are surfaced as floating windows which can be set to automatically hide when unused or docked to the side of the screen. The Visual Studio code editor also supports code refactoring including parameter reordering, variable and method renaming, interface extraction and encapsulation of class members inside properties, among others. Visual Studio features background compilation (also called incremental compilation). As code is being written, Visual Studio compiles it in the background in order to provide feedback about syntax and compilation errors, which are flagged with a red wavy underline. Warnings are marked with a green underline. Background compilation does not generate executable code, since it requires a different compiler than the one used to generate executable code. Background compilation was initially introduced with Microsoft Visual Basic but has now been expanded for all included languages.

Debugger

Visual Studio includes a debugger that works both as a source-level debugger and as a machine-level debugger. It works with both managed code as well as native code and can be used for debugging applications written in any language supported by Visual Studio. In addition, it can also attach to running processes and monitor and debug those processes. If source code for the running process is available, it displays the code as it is being run. If source code is not available, it can show the disassembly. The Visual Studio debugger can also create memory dumps as well as load them later for debugging. Multi-threaded programs are also supported. The debugger can be configured to be launched when an application running outside the Visual Studio environment crashes. The debugger allows

setting breakpoints (which allow execution to be stopped temporarily at a certain position) and watches (which monitor the values of variables as the execution progresses). Breakpoints can be conditional, meaning they get triggered when the condition is met. Code can be stepped over, i.e., run one line (of source code) at a time. It can either step into functions to debug inside it, or step over it, i.e., the execution of the function body isn't available for manual inspection. The debugger supportsEdit and Continue, i.e., it allows code to be edited as it is being debugged (32 bit only; not supported in 64 bit). When debugging, if the mouse pointer hovers over any variable, its current value is displayed in a tooltip ("data tooltips"), where it can also be modified if desired. During coding, the Visual Studio debugger lets certain functions be invoked manually from the Immediate tool window. The parameters to the method are supplied at the Immediate window.

Designer

Visual Studio includes a host of visual designers to aid in the development of applications. These tools include: Windows Forms Designer The Windows Forms designer is used to build GUI applications using Windows Forms. Layout can be controlled by housing the controls inside other containers or locking them to the side of the form. Controls that display data (like textbox, list box, grid view, etc.) can be bound to data sources like databases or queries. Data-bound controls can be created by dragging items from the Data Sources window onto a design surface. The UI is linked with code using an event-driven programming model. The designer generates either C sharp orVB.NET code for the application.

WPF Designer

The WPF designer, codenamed Cider, was introduced with Visual Studio 2008. Like

the Windows Forms designer it supports the drag and drop metaphor. It is used to author user interfaces targeting Windows Presentation Foundation. It supports all WPF functionality including data binding and automatic layout management. It generates XAML code for the UI. The generated XAML file is compatible with Microsoft Expression Design, the designer-oriented product. The XAML code is linked with code using a code-behindmodel. Web designer/development Visual Studio also includes a web-site editor and designer that allows web pages to be authored by dragging and dropping widgets. It is used for developing ASP.NET applications and supports HTML, CSS and JavaScript. It uses a code-behind model to link with ASP.NET code. From Visual Studio 2008 onwards, the layout engine used by the web designer is shared with Microsoft Expression Web. There is also ASP.NET MVC support for MVC technology as a separate download and ASP.NET Dynamic Data project available from Microsoft.

Class designer

The Class Designer is used to author and edit the classes (including its members and their access) using UML modeling. The Class Designer can generate C and VB.NET code outlines for the classes and methods. It can also generate class diagrams from hand-written classes.

Data designer

The data designer can be used to graphically edit database schemas, including typed tables, primary and foreign keys and constraints. It can also be used to design queries from the graphical view.

Mapping designer

From Visual Studio 2008 onwards, the mapping designer is used by LINQ to SQL to design the mapping between database schemas and the classes that encapsulate the data. The

new solution from ORM approach, ADO.NET Entity Framework, replaces and improves the old technology.

4.2.2 Microsoft SQL server

MS SQL Server is a relational database management system (RDBMS) developed by Microsoft. This product is built for the basic function of storing retrieving data as required by other applications. It can be run either on the same computer or on another across a network. This tutorial explains some basic and advanced concepts of SQL Server such as how to create and restore data, create login and backup, assign permissions, etc. Each topic is explained using examples for easy understanding.

Audience

This tutorial is designed for all those readers who want to learn the fundamentals of SQL Server and put it into practice.

Usage of SQL Server

To create databases.

To maintain databases.

To analyze the data through SQL Server Analysis Services (SSAS).

To generate reports through SQL Server Reporting Services (SSRS).

To carry out ETL operations through SQL Server Integration Services (SSIS).

Table Script

CREATE TABLE (dbo).(Rfid_ BookDetails) (

(BookID) INT IDENTITY (1, 1) NOT NULL,

(Title) VARCHAR (MAX) NULL,

(Author) VARCHAR (MAX) NULL,

(Domain) VARCHAR (MAX) NULL,

(TotalQuantity) INT NULL,

(AvailableQuantity) INT NULL,

(OccupiedQuantity) INT NULL,

PRIMARY KEY CLUSTERED ((BookID) ASC));

CREATE TABLE (dbo).(Rfid_ StudentDetails)

((Id) INT IDENTITY (1, 1) NOT NULL,

(StudentRfid) VARCHAR (50) NULL,

(Name) VARCHAR (MAX) NULL,

(Roll) VARCHAR (50) NULL,

(Year) VARCHAR (50) NULL,

(Branch) VARCHAR (MAX) NULL,

PRIMARY KEY CLUSTERED ((Id) ASC));

CREATE TABLE (dbo).(Rfid_ Temp)

((Id) INT NOT NULL,

(Type) VARCHAR (500) NULL,

(Value) VARCHAR (MAX) NULL,

PRIMARY KEY CLUSTERED ((Id) ASC));

4.2.3 Hardware Requirements

RFID Reader (EM-18)

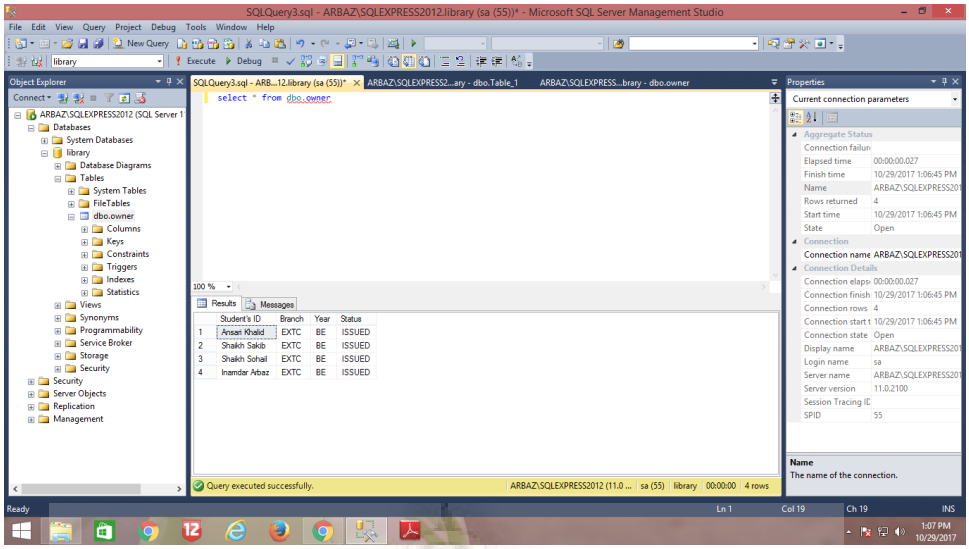

Figure 4.2: Database

This module directly connects to any microcontroller UART or through a RS232 converter to PC.It gives UART/Wiegand26 output. This RFID Reader Module works with any 125 KHz RFID tags

Specifications

5VDC through USB (External 5V supply will boost range of the module)

Current: ¡50mA

Operating Frequency: 125Khz

Read Distance: 10cm

Size of RFID reader module: 32mm(length) * 32mm(width) * 8mm(height)

4.2.3.1 RFID tag

RFID tagging is an ID system that uses small radio frequency identification devices for identification and tracking purposes. An RFID tagging system includes the tag itself, a read/write device, and a host system application for data collection, processing, and transmission.

Figure 4.3: Reader module

RS232 interface format:

10 ASCII DATA (card no.)+ 2 ASCII DATA (XOR result)

E.g. Card number is 4500C5D1E9B8 read from reader then the card number on card will be as below.

45 - Preamble

00C5D1E9 value in Hex = 12964329. / B8 is XOR value for (45 XOR 00 XOR C5 XOR D1 XOR E9)

Hence number on the card is 0012964329.

1. Data baud rate: 9600 bps

2. Data bit 8 bits

3. Parity check: None

4. Stop bit

USB TTL

DESCRIPTION

The cable is easiest way ever to connect to your microcontroller/Raspberry Pi/WiFi router serial console port. Inside the big USB plug is a USB¡-¿Serial conversion chip and at the end of the 36" cable are four wire - red power, black ground, white RX into USB port, and green TX out of the USB port. The power pin provides the 5V @ 500mA direct from the USB port and the RX/TX pins are 3.3V level for interfacing with the most common 3.3V logic level chipsets.

Because of the separated pin plugs, this cable is ideal for powering and connecting up to the debug/login console on the Raspberry Pi or BeagleBone Black. Connect the pins as shown to power the Pi or BBB and establish the RX/TX link.

If you are running Windows 7/8/10 etc, check this tutorial page with links to drivers for both PL2303 and CP2102 If you are running Mac OS X, check this tutorial page with links to drivers for both PL2303 and CP2012 If you are running Linux, drivers are already included in the kernel, no need to install anything! Also handy for hacking WiFi routers to install alternate OS's, or nearly any other 3.3V logic serial port. This is easier to use than an FTDI cable in many cases because the wires are separated. Note that we call this a "TTL cable" (since that's what they're called) but technically it's CMOS logic.

This cable is not good for Arduino re-programming such as a Boarduino, MENTA, Monochron, etc. because it does not have the DTR/RTS wire necessary for initiating

the bootloader reboot sequence. For that we suggest an FTDI cable or FTDI friend.

4.2.3.2 DipTrace

DipTrace is an EDA/CAD software for creating schematic diagrams and printed circuit boards. The developers provide a multi-lingual interface and tutorials (currently available in English and 21 other languages). DipTrace has 4 modules: schematic capture editor, PCB layout editor with built-in shape-based autorouter and 3D-preview and export, component editor, and pattern editor.

Basic features

Simple user interface Multi-sheet and hierarchical schematics High-speed and Differential signal routing Smart manual routing modes Wide import/export capabilities High-speed shape-based autorouter Advanced verifications with real-time DRC Real-time 3D PCB preview Export of PCB to STEP 3D file format ODB++ and Gerber (including Gerber X2) manufacturing outputs

Advanced circuit design tool with support of multi-sheet and multi-level hierarchical schematics that delivers a number of features for visual and logical pin connections. Cross-module management ensures that principal circuits can be easily converted into a PCB, back-annotated, or imported/exported from/to other EDA software, CAD formats and net-lists. DipTrace Schematic has ERC verification and Spice export for external simulation.

4.2.3.3 PCB layout

Engineering tool for board design with smart manual routing, differential pairs, length-matching tools, shape-based autorouter, advanced verification, layer stackup manager, and wide import/export capabilities. Design requirements are defined by net

classes, class-to-class rules, and detailed settings by object types for each class or layer. When routing with real-time DRC, the program reports errors on the fly before actually making them. DRC also checks length and phase tolerances for differential pairs and controls signal synchronization for nets and buses (including layer stackup and bonding wire induced signal delays). The board can be previewed in 3D and exported to STEP format for mechanical CAD modeling. Design rule check with in-depth detailing and net connectivity verification procedures are available.

3D-preview and export

This module includes real-time 3D preview and export feature. It shows the model of the manufactured printed circuit board with all components installed. Rotate board in three axes, zoom in and out in real time, change colors of the board, copper areas, solder mask, silkscreen, and background. 3D preview works on all stages of the design. Board can be exported to STEP or VRML 2.0 formats for mechanical CAD modeling. More than 7500 3D models of PCB packages are supplied for free. Externally designed 3D models in *.wrl, *.step, *.iges, and *.3ds formats can be uploaded and attached to patterns in Pattern Editor or PCB Layout.

4.2.3.4 Component editor

Manage component libraries and create single- or multi-part components by selecting a template and its dimensions, defining visual and electrical pin parameters, setting up a Spice model, and attaching pattern with a 3D model to finalize component creation. BSDL import, bulk pin naming, and pin manager tools for pins and buses. Importing libraries from different EDA formats. More than 140000 components in standard libraries.

4.2.3.5 Pattern editor

Draw patterns with various types of shapes, pads, holes, and dimensions. Circle, lines (headers, DIP), square (QFP), matrix (BGA), rectangle (RQFP), and zig-zag standard templates. Creation of pattern is basically selecting a template, entering a couple of vital parameters, drawing the silkscreen, and launching automatic pad renumbering. Custom templates can be created for non-standard patterns. DXF import makes creating complex layouts easier.

Chapter 5

Expected Outcome

We can ensure that using the concept of Smart Library Management system, is made well manage to access and map the status of book racks reducing the human labour. This reduces the human involvement in the libray,and making that outdated library a history.

5.0.1 Main Form Designer

namespace Library_ Management
{
partial class Form1
{
/// ¡summary¿
/// Required designer variable.
/// ¡/summary¿
private System.ComponentModel.IContainer components = null;

/// ¡summary¿
/// Clean up any resources being used.
/// ¡/summary¿
/// ¡param name=”disposing”¿true if managed resources should be disposed; otherwise, false.¡/param¿ protected override void Dispose(bool disposing)
{
if (disposing && (components != null))
{
components.Dispose();
}
base.Dispose(disposing);
}

region Windows Form Designer generated code

```
        /// ¡summary¿
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// ¡/summary¿
private void InitializeComponent()
{
this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
this.label1 = new System.Windows.Forms.Label();
this.tableLayoutPanel2 = new System.Windows.Forms.TableLayoutPanel();
this.register = new System.Windows.Forms.Button();
this.addBook = new System.Windows.Forms.Button();
this.button3 = new System.Windows.Forms.Button();
this.tableLayoutPanel3 = new System.Windows.Forms.TableLayoutPanel();
this.tableLayoutPanel4 = new System.Windows.Forms.TableLayoutPanel();
this.label2 = new System.Windows.Forms.Label();
this.label3 = new System.Windows.Forms.Label();
this.label4 = new System.Windows.Forms.Label();
this.label5 = new System.Windows.Forms.Label();
this.label6 = new System.Windows.Forms.Label();
this.label7 = new System.Windows.Forms.Label();
this.label8 = new System.Windows.Forms.Label();
this.textBox1 = new System.Windows.Forms.TextBox();
this.dataGridView1 = new System.Windows.Forms.DataGridView();
this.tableLayoutPanel1.SuspendLayout();
this.tableLayoutPanel2.SuspendLayout();
this.tableLayoutPanel3.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();
this.SuspendLayout();
//
// tableLayoutPanel1
//
this.tableLayoutPanel1.Anchor = ((System.Windows.Forms.AnchorStyles)
(((System.Windows.Forms.AnchorStyles.Top — System.Windows.Forms.AnchorStyles.Left)
— System.Windows.Forms.AnchorStyles.Right)));
this.tableLayoutPanel1.ColumnCount = 1;
this.tableLayoutPanel1.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
50F));
this.tableLayoutPanel1.Controls.Add(this.label1, 0, 0);
this.tableLayoutPanel1.Location = new System.Drawing.Point(2, 2);
this.tableLayoutPanel1.Name = "tableLayoutPanel1";
this.tableLayoutPanel1.RowCount = 1;
this.tableLayoutPanel1.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.For
50F));
this.tableLayoutPanel1.Size = new System.Drawing.Size(923, 53);
this.tableLayoutPanel1.TabIndex = 0;
//
// label1
//
```

```
this.label1.AutoSize = true;
this.label1.BackColor = System.Drawing.Color.Teal;
this.label1.Dock = System.Windows.Forms.DockStyle.Fill;
this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 15.75F, System.Drawing.FontSty
System.Drawing.GraphicsUnit.Point, ((byte)(0))); this.label1.ForeColor = System.Drawing.Color.Wh
this.label1.Location = new System.Drawing.Point(3, 0);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(917, 53);
this.label1.TabIndex = 0;
this.label1.Text = "RFID based Library Management System";
this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
//
// tableLayoutPanel2
//
this.tableLayoutPanel2.ColumnCount = 1;
this.tableLayoutPanel2.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
50F));
this.tableLayoutPanel2.Controls.Add(this.register, 0, 0);
this.tableLayoutPanel2.Controls.Add(this.addBook, 0, 1);
this.tableLayoutPanel2.Controls.Add(this.button3, 0, 2);
this.tableLayoutPanel2.Location = new System.Drawing.Point(2, 281);
this.tableLayoutPanel2.Name = "tableLayoutPanel2";
this.tableLayoutPanel2.RowCount = 3;
this.tableLayoutPanel2.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.For
50F));
this.tableLayoutPanel2.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.For
50F));
this.tableLayoutPanel2.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.For
42F));
this.tableLayoutPanel2.Size = new System.Drawing.Size(200, 124);
this.tableLayoutPanel2.TabIndex = 1;
//
// register
//
this.register.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(64))))
((int)(((byte)(64)))));
this.register.Dock = System.Windows.Forms.DockStyle.Fill;
this.register.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.register.ForeColor = System.Drawing.Color.White;
this.register.Location = new System.Drawing.Point(3, 3);
this.register.Name = "register";
this.register.Size = new System.Drawing.Size(194, 35);
this.register.TabIndex = 0;
this.register.Text = "Register";
this.register.UseVisualStyleBackColor = false;
this.register.Click += new System.EventHandler(this.register_ Click);
//
// addBook
```

```
//
this.addBook.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(64))))
((int)(((byte)(64)))));
this.addBook.Dock = System.Windows.Forms.DockStyle.Fill;
this.addBook.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.addBook.ForeColor = System.Drawing.Color.White;
this.addBook.Location = new System.Drawing.Point(3, 44);
this.addBook.Name = "addBook";
this.addBook.Size = new System.Drawing.Size(194, 35);
this.addBook.TabIndex = 1;
this.addBook.Text = "Add Book";
this.addBook.UseVisualStyleBackColor = false;
this.addBook.Click += new System.EventHandler(this.addBook_ Click);
//
// button3
//
this.button3.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(64))))
((int)(((byte)(64)))));
this.button3.Dock = System.Windows.Forms.DockStyle.Fill;
this.button3.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.button3.ForeColor = System.Drawing.Color.White;
this.button3.Location = new System.Drawing.Point(3, 85);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(194, 36);
this.button3.TabIndex = 2;
this.button3.Text = "Settings";
this.button3.UseVisualStyleBackColor = false;
//
// tableLayoutPanel3
//
this.tableLayoutPanel3.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms
— System.Windows.Forms.AnchorStyles.Left)
— System.Windows.Forms.AnchorStyles.Right)));
this.tableLayoutPanel3.CellBorderStyle = System.Windows.Forms.TableLayoutPanelCellBorderStyle
this.tableLayoutPanel3.ColumnCount = 10;
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
7.658643F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
14.55142F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
7.986871F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
22.21007F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
8.533916F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
8.205689F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
```

```
7.549234F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
7.877462F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
7.330416F));
this.tableLayoutPanel3.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
10.17505F));
this.tableLayoutPanel3.Controls.Add(this.label4, 0, 0);
this.tableLayoutPanel3.Controls.Add(this.label5, 2, 0);
this.tableLayoutPanel3.Controls.Add(this.label6, 4, 0);
this.tableLayoutPanel3.Controls.Add(this.label7, 6, 0);
this.tableLayoutPanel3.Controls.Add(this.label8, 8, 0);
this.tableLayoutPanel3.Location = new System.Drawing.Point(5, 79);
this.tableLayoutPanel3.Name = "tableLayoutPanel3";
this.tableLayoutPanel3.RowCount = 1;
this.tableLayoutPanel3.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.For
100F));
this.tableLayoutPanel3.Size = new System.Drawing.Size(917, 55);
this.tableLayoutPanel3.TabIndex = 2;
//
// tableLayoutPanel4
//
this.tableLayoutPanel4.Anchor = ((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms
— System.Windows.Forms.AnchorStyles.Left)
— System.Windows.Forms.AnchorStyles.Right)));
this.tableLayoutPanel4.CellBorderStyle = System.Windows.Forms.TableLayoutPanelCellBorderStyle
this.tableLayoutPanel4.ColumnCount = 5;
this.tableLayoutPanel4.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
20F));
this.tableLayoutPanel4.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
20F));
this.tableLayoutPanel4.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
20F));
this.tableLayoutPanel4.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
20F));
this.tableLayoutPanel4.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windo
20F));
this.tableLayoutPanel4.Location = new System.Drawing.Point(6, 151);
this.tableLayoutPanel4.Name = "tableLayoutPanel4";
this.tableLayoutPanel4.RowCount = 1;
this.tableLayoutPanel4.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.For
100F));
this.tableLayoutPanel4.Size = new System.Drawing.Size(917, 55);
this.tableLayoutPanel4.TabIndex = 3;
//
// label2
//
this.label2.AutoSize = true;
```

```
this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 14.25F,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label2.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(64)))),
((int)(((byte)(64)))));
this.label2.Location = new System.Drawing.Point(5, 66);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(149, 24);
this.label2.TabIndex = 4;
this.label2.Text = "Student Details";
//
// label3
//
this.label3.AutoSize = true;
this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 14.25F,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label3.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(64)))),
((int)(((byte)(64)))));
this.label3.Location = new System.Drawing.Point(8, 139);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(125, 24);
this.label3.TabIndex = 5;
this.label3.Text = "Book Details";
//
// label4
//
this.label4.AutoSize = true;
this.label4.Dock = System.Windows.Forms.DockStyle.Right;
this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label4.Location = new System.Drawing.Point(6, 3);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(60, 49);
this.label4.TabIndex = 6;
this.label4.Text = "RFID No";
this.label4.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
//
// label5
//
this.label5.AutoSize = true;
this.label5.Dock = System.Windows.Forms.DockStyle.Right;
this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label5.Location = new System.Drawing.Point(216, 3);
this.label5.Name = "label5";
```

```
this.label5.Size = new System.Drawing.Size(51, 49);
this.label5.TabIndex = 7;
this.label5.Text = "Name :";
this.label5.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
//
// label6
//
this.label6.AutoSize = true;
this.label6.Dock = System.Windows.Forms.DockStyle.Right;
this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label6.Location = new System.Drawing.Point(479, 3);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(59, 49);
this.label6.TabIndex = 8;
this.label6.Text = "Roll No :";
this.label6.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
//
// label7
//
this.label7.AutoSize = true;
this.label7.Dock = System.Windows.Forms.DockStyle.Right;
this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label7.Location = new System.Drawing.Point(637, 3);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(43, 49);
this.label7.TabIndex = 9;
this.label7.Text = "Year :";
this.label7.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
//
// label8
//
this.label8.AutoSize = true;
this.label8.Dock = System.Windows.Forms.DockStyle.Right;
this.label8.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label8.Location = new System.Drawing.Point(761, 3);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(56, 49);
this.label8.TabIndex = 10;
this.label8.Text = "Branch :";
this.label8.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
//
// textBox1
```

```
//
this.textBox1.Location = new System.Drawing.Point(6, 214);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(193, 20);
this.textBox1.TabIndex = 6;
this.textBox1.Text = "Search Book...";
//
// dataGridView1
//
this.dataGridView1.ColumnHeadersHeightSizeMode
=
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize; this.dataGridView
= new System.Drawing.Point(206, 213);
this.dataGridView1.Name = "dataGridView1";
this.dataGridView1.Size = new System.Drawing.Size(716, 189);
this.dataGridView1.TabIndex = 7;
//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.BackColor = System.Drawing.Color.White;
this.ClientSize = new System.Drawing.Size(927, 407);
this.Controls.Add(this.dataGridView1);
this.Controls.Add(this.textBox1);
this.Controls.Add(this.label3);
this.Controls.Add(this.label2);
this.Controls.Add(this.tableLayoutPanel4);
this.Controls.Add(this.tableLayoutPanel3);
this.Controls.Add(this.tableLayoutPanel2);
this.Controls.Add(this.tableLayoutPanel1);
this.Name = "Form1";
this.Text = "RFID based Library Management System";
this.Load += new System.EventHandler(this.Form1_ Load);
this.tableLayoutPanel1.ResumeLayout(false);
this.tableLayoutPanel1.PerformLayout();
this.tableLayoutPanel2.ResumeLayout(false);
this.tableLayoutPanel3.ResumeLayout(false);
this.tableLayoutPanel3.PerformLayout();
((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();

    }

    # endregion

    private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
```

```
private System.Windows.Forms.Label label1;
private System.Windows.Forms.TableLayoutPanel tableLayoutPanel2;
private System.Windows.Forms.Button register;
private System.Windows.Forms.Button addBook;
private System.Windows.Forms.Button button3;
private System.Windows.Forms.TableLayoutPanel tableLayoutPanel3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.TableLayoutPanel tableLayoutPanel4;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.DataGridView dataGridView1;
}
}
```

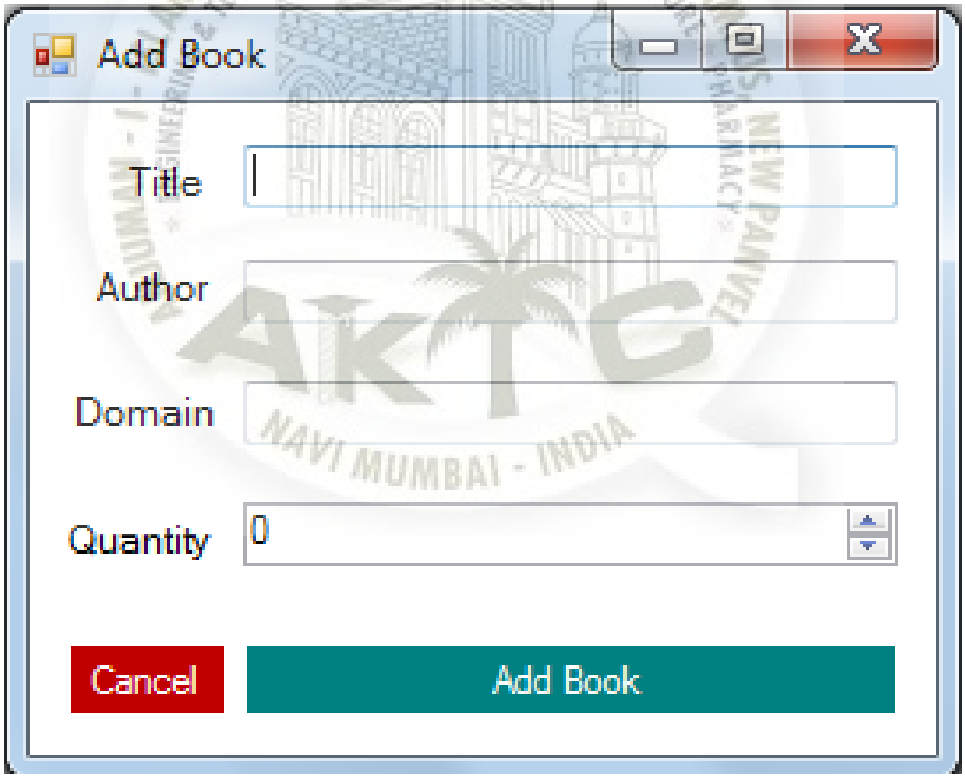

Figure 5.1: Add Book Form

5.0.2 Main Form

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

    namespace Library_ Management
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
CenterToScreen();
}

    private void Form1_ Load(object sender, EventArgs e)
{

    }

    private void register_ Click(object sender, EventArgs e)
{
RegisterForm rf = new RegisterForm();
rf.Show();
}

    private void addBook_ Click(object sender, EventArgs e)
{
AddBookForm abf = new AddBookForm();

    abf.Show();

    }
}
}
```

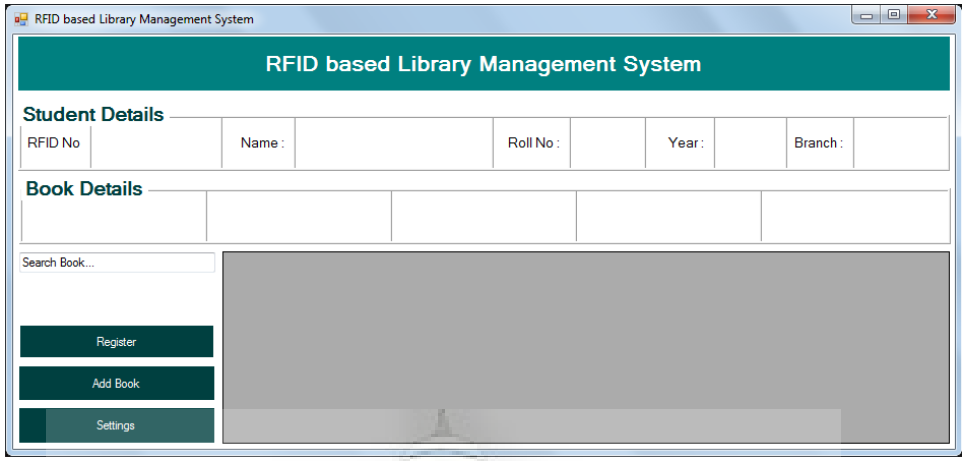

Figure 5.2: Main Form

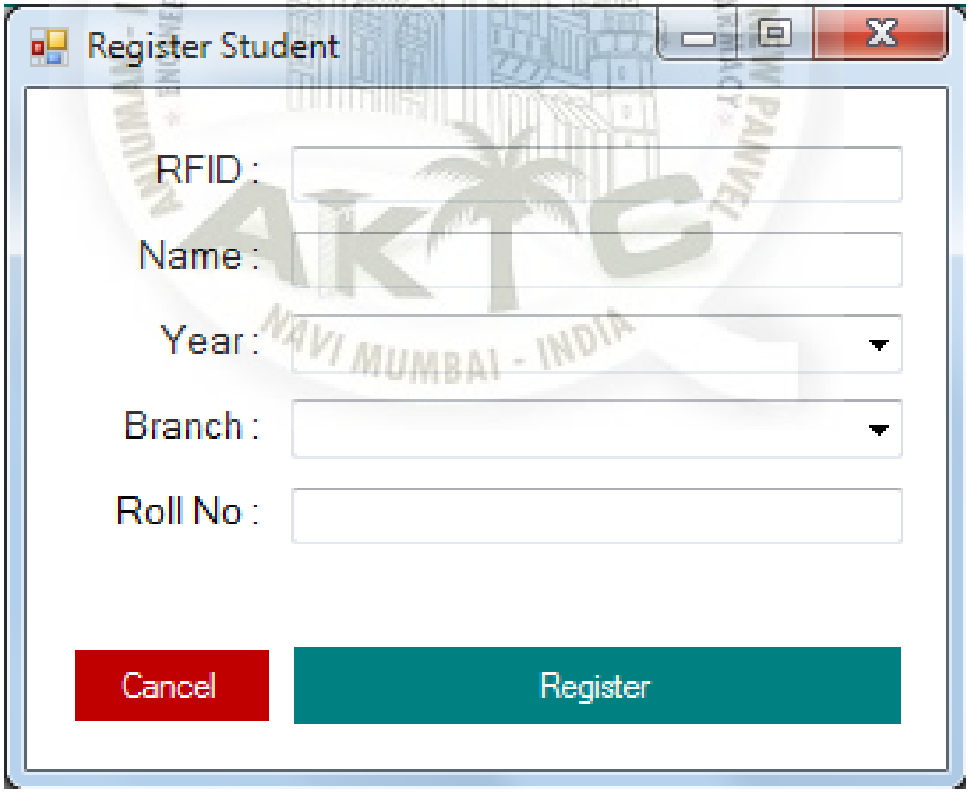

Figure 5.3: Register Form

References

- http://www.ijareeie.com/upload/2015/april/17A_ 087_ Annaraman.pdf.
- https://pdfs.semanticscholar.org/de2b/5111888f2b6f51c756d2d2c6744afae1de87.pdf
- http://www.nskelectronics.com/em-18_ rfid_ reader.html
- T. Taleb and A. Kunz, "Machine Type Communications in 3GPP Networks: Potential Challenges, and Solutions", to appear, IEEE Commun.