# Improving the EMS Library

Luke Bowden & Callum O'Brien
Exeter Mathematics School

October 25, 2015

# Contents

# Chapter 1

# Details and Context

## 1.1 Description of the Current System

Exeter Mathematics School's library contains, as one might expect, books. Some of the books are available to check out of the library, whereas others are not; currently, this is communicated by stickers on the books with a grey sticker on the spine of a book meaning one is not to take that book out of the library. Other colour stickers of books' spines mean one can check them out.

When a student checks out a book, they write their name in a log book, along with the current date. Upon returning the book, they find the entry in the log book corresponding to its checking out and add the date they returned it. If one wishes to find out who has a book or, indeed, whether a book is currently in the library (without visually searching the shelves), one must read through this log book, find the lastest entry regarding the book one wishes to know about and see who checked it out and/or whether the book has been returned.

## 1.2 Inefficiencies Arising from the Current System

## 1.3 Potential Solutions to Current Inefficiencies

## 1.4 Appraisal of Potential Solutions

# Chapter 2

# Design

## 2.1 Overall System Design

The proposed system involves (as well as the obvious) a daemon, a database and an NFC reader. Upon checking out a book, a student will touch both the book and their identification card to an NFC reader device. This will invoke the daemon which will query the Book and Student tables in the database for the UIDs of the book and the student. After these data have been retrieved, the daemon will query the database again, this time concerning the BookStudent table; the daemon requests the UIDs of any BookStudent records whose BookID and StudentID attributes are the recently retrieved UIDs and whose pending attribute is True. This query will return either nothing or a single UID. If no UID is received, the daemon will create a new record in the BookStudent table. This record will have;

- BookID of the Book's UID
- StudentID of the Student's UID
- OutTime of the current time
- InTime of NULL
- Pending of True

If a UID is received, the record with that UID is updated to have;

- InTime of the current time
- Pending of False

## 2.2 Entity Relationships & Database Structure

The database must encapsulate two entities, Book and Student, the Book entity being an abstraction of a book in the EMS library and the Student entity being an abstraction of a student attending EMS. The Book entity must contain all information one might wish to identify a book by; its

| Table | Feild | Type |
|---|---|---|
| Book | Author | Int |
| | UID | Int |
| | ISBN | String |
| | Subject | String |
| | Title | String |
| BookStudent | BookID | Int |
| | InTime | Date |
| | OutTime | Date |
| | Pending | Bool |
| | StudentID | Int |
| | UID | Int |
| Student | Name | String |
| | UID | Int |

Table 2.1: Tables and the fields they contain

author, isbn, subject and title. However, each instance of this entity must be unique and so a fifth attribute, a UID, is required. The Student entity has attributes name and UID.

A single Student may take out many Books and a single Book may be taken out by many Students, thus these entities have a many-to-many relationship. In order to fully encapsulate these entities in a relational database we must normalise this relationship, creating a third entity; BookStudent. A BookStudent record will be created for every time a Student takes out a Book. This effects a one-to-many relationship between Book and BookStudent and a many-to-one relationship between BookStudent and Student (see figure 2.3).

## 2.3 User Interface

## 2.4 Security Measures


Figure 2.1: The table structure of the database before normalisation.


Figure 2.2: The table structure of the database after normalisation.

# Chapter 3

# Implementation

## 3.1 Discussion of Language

## 3.2 Discussion of Platform

## 3.3 Issues and Resolutions

# Chapter 4

# Testing

# Chapter 5

# Evaluation

## 5.1 Assessment of Features

## 5.2 Known Bugs

# Appendix A

# Source Code

# List of Figures

# List of Tables

# Bibliography