|  |
| --- |
| **INDEPENDENT UNIVERSITY, BANGLADESH** |
| School of [Engineering and Computer Science](http://www.secs.iub.edu.bd/)    **PROJECT: Online Management System**  **By Team: Byte.M3**  Syeda Sutapa Rahman Meem-1530232  Sadek Hossen Leon -1521818  Kazi Imdadul Hoque -1522248  Syed Shams Elahi -1621176  Instructor: Radiah Rose Haque  Date: 23/06/2018 |

**Declaration of Originality**

This project is all our group own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines ,internet ,etc ) has been acknowledged within the main report to an entry in the References list.

Abstract

A management system is a project which goals in growing an automated System’s to keep all the everyday work. Our university management system will do this for libraries and notice boards. This task has many functions which are usually not available in regular library management systems and online notice boards like facility of consumer login and a facility of instructors login .It also has a facility of admin login through which the admin can screen the complete system .It additionally has facility of an online observe board in which instructors can student can placed up data approximately workshops or seminars being held in our university or close by universities and librarian after right verification from the concerned organization organizing the seminar can upload it to the notice board . it has also a facility in which student after logging of their accounts can see listing of books issued and its issue date and return date and additionally the students can request the librarian to add some new book by using filling book request form. The librarian logs into his account, which is the admin account, then they can generate several reports like student report, issue report, teacher report and book report. All in all, this project of ours is being developed to aid the students and staff of library to manage the library in the most efficient way viable and also minimize the human tasks. It will also provide a complete fully functional notice board online for the university.

Table of Contents

[Abstract 2](#_Toc515726839)

[Table of Contents 3](#_Toc515726840)

[Additional Materials on the Accompanying CD 4](#_Toc515726841)

[Acknowledgements 5](#_Toc515726842)

[1 Introduction 6](#_Toc515726843)

[1.1 Background to the Project 6](#_Toc515726844)

[1.2.1 (](#_Toc515726846)*[eg](#_Toc515726846)*[) Gather client and user requirements](#_Toc515726846) 7

[1.2.2 (](#_Toc515726847)*[eg](#_Toc515726847)*[) Analyse and model the requirements](#_Toc515726847) 8

[1.2.3 (](#_Toc515726848)*[eg](#_Toc515726848)*[) Investigate possible solutions, etc, etc](#_Toc515726848) 9-11

[1.3 Overview of This Report](#_Toc515726849) 12

[2 Literature Review](#_Toc515726850) 13

[3 Methodology](#_Toc515726851) 14-16

[4 Requirements 19](#_Toc515726852)

[5 Analysis](#_Toc515726853) 20

[6 Design](#_Toc515726854) 21

[7 Implementation](#_Toc515726855) 22

[8 Testing](#_Toc515726856) 23

[9 Project Management](#_Toc515726857) 24

[9.1 Project Schedule](#_Toc515726858) 24

[9.2 Social, Legal, Ethical and Professional Considerations](#_Toc515726859) 24

[10 Conclusions](#_Toc515726860) 25

[10.1 Achievements](#_Toc515726861) 25

[10.2 Critical Appraisal](#_Toc515726862) 25

[10.3 Limitations](#_Toc515726863) 25

[10.4 Future Work](#_Toc515726864) 25

[11 Group Reflections](#_Toc515726865) 26

12 [Bibliography and References](#_Toc515726866) 27

13 [Appendix A – Project Specification](#_Toc515726867) 28

14 [Appendix B User Manual](#_Toc515726868) 29

15 [Appendix E – Project Presentation](#_Toc515726869) 30

16 [Appendix X – As required](#_Toc515726870) 31

Acknowledgements

The project we submitted herewith is a result of our own effort in totality and in every aspects of the project works. Materials, which we used in the project found by web site, are mentioned by reference. All the information that has been obtained from other sources have been fully acknowledged. We understand that any plagiarism, cheating or collusion of any sorts constitutes a breach of University rules and regulations and would be subjected to disciplinary actions.

The satisfaction that accompanies that the successful completion of any task would be incomplete without them ention of people whose ceaseless cooperation made it possible, whose constant guidance and encourage men to crown all efforts with success.

We are grateful to our course instructor ,Miss Radiah Rose Haque, for the guidance, inspiration and constructive suggestions that helpful us in the preparation of this project.

We also thank our Student on Duty (SOD), Rabbi Iqbal, who has helped in successful completion of the project.

**Syeda Sutapa Rahman Meem -1530232**

**Sadek Hossen Leon-1521818**

**Kazi Imdadul Hoque -1522248**

**Syed Shams Elahi-1621176**

# 1. Introduction

This chapter provides an outlook into the motives, objectives, backgrounds and operation environment of the system.

# 1.1 Background

A University Management System has 2 sub components.

They are:

1. Notice Management System
2. Library Management System

A Notice Management System is an application which enables the staff members of a University (Registrar’s office, Faculties) or organization to make notices and important dates or updates available to everyone involved with ease. Notices regarding makeup class, holidays and exams can be posted and seen only by those in the organization or University.

A Library Management System is an application which refers to library systems which are generally small or medium in size. Librarians use it to maintain the library using a computerized system where they can record several transactions like issue of books, return of books, addition of new books, addition of new students etc. Books and student maintenance modules are also incorporated in this system which would track the students who use the library and also an informative description relating to the books a library contains. With this computerized system there will be no loss of book record or member record which generally happens when a non-computerized system is used. Moreover, a report module is also added in this Library Management System. If user’s position is admin, the user is able to generate different kinds of reports like lists of students registered, list of books, issue and return reports. All these modules are able to aid the librarian to maintain the library with more ease and in a more sustainable way compared to manual non-computerized library systems.

The Integrated library management will keep all records of a library operation such as items, bill, paid, and also patrons’ record. Using this software we can operate all the library operations with ease. Integrated library system there is two interfaces one is patron and another one is for library staff. In library system the operation of a user/member/patron and staff of library is different. A user can search a book, view book list which are available in library, can issue book, renew book, can hold book, can print issue list, can edit his/her information (patron information). Online public excess catalogue, circulation, some function under serial control can operate by patron. But a staff can store bibliographic (book, CD, DVD, journal etc.) record actually library materials record in database, can create patron in database, order a book, purchase a book. Acquisition, cataloguing, some function under serial control, management all are operate by staff of a library.

Modern Generation is going to dependable on computer and through computer we use Software. By using this, we can store a student’s information, issue a book and store a student’s information are used by library administration.

Libraries are essential in a process of giving citizens access to knowledge. In digital times they are needed more than before. In our country, library systems are written that’s why we make great use of paper. By making paper we cut trees. It is very harmful for our environment. For our environment, we take this step to reduce this. This increases environment friendliness and company profits.

The Management System will cut down the amount of human resources required by the company as much of the Tasks of putting up notices and maintaining documents, printing, and managing forms can be done online by an individual. Thus increasing the profit.

All these combined will help make the organization take a step into modernization, be more competitive and updated as other companies using the latest technologies, improve the outlook of the company, overall making the current existing model a new transformed and much better oiled machine.

# 1.2 Objectives

The aim of this project is to develop a system that can handle & manage the activities involved in a library management and university notices in an efficient & reliable way. Less managing personnel & easy searching availability and user profile managing are major goals in this project.

The project goals and objectives that will be gained after completing this project are discussed in this subchapter. The aims and objectives are as follows:

* Develop a system that can replace the manual library managing system.
* Develop a database which stores user details and book details.
* Give reliable search facility for the user.
* Administrator or librarian should have logins.
* Create an easy to understand user friendly environment.
* Attractive user interfaces to navigate through the system for the users.
* Develop the system documentation with detailed UML specification.
* The option for students to search online for books in the library
* See if books are reserved, available or borrowed
* Advanced search features to find books
* A student interface with the options for account creation, login, and forgot your password
* Student accounts can see books they have borrowed, due date, and can borrow new books
* Student accounts can also see books which students with similar tastes or departments have borrowed
* A library staff interface with the options for account creation, login, and forgot your password
* Library staff accounts can see total number of books in library, which are available, which are borrowed, which are reserved
* Library staff can add new books, edit their details, or remove existing books
* Library staff accounts can issue books to students, and verify student accounts
* A notice board web site showing all upcoming seminars, workshops and events or new facilities in university
* University staffs can update notice board and modify it as required on timely basis
* Students can give their reviews and comment on books
* An all access super-user account which has permission to access and privileges to everything
* Special permissions to access and privileges for each type of accounts
* Search filtering
* Special filter sidebar
* Email notification on withdrawing books, and reminder of return date, and if overdue, with fined amount
* Secure password
* Login using Facebook
* Responsive stable web application for all platforms
* A university site page with notices, and information about events, and date and venue
* Departments or register office or staff can add, update, delete events
* Students can see event information
* Has a Powerful look for engine
* Maintaining any media
* Help and User's Guide
* Book issuing books Online
* Requesting column for librarian for providing new books
* a separate digital library repository
* Student login page where student can find books issued by him/her and date of return.
* A search column to search for books
* A teacher login page where teacher can add any events being organized in the college and important suggestions regarding books.
* Online notice board about the workshop.
* A librarian login page.

**Elaborated:**

* The Management System will can be signed up by students, faculties, library and Registrar’s office staff members. The students will need their student ID and IUB email address to be verified. The rest will be verified by their Faculty or Staff ID and email address.
* Using our Management System, the staff from the Registrar’s office or faculties can login online and post notices in this online notice board.
* The Library staff can login and add new books, see which books (and which copies) are available and withdrawn and overdue, and request for new copies of lost books.
* The students will receive notifications of new notices, and book withdrawn, due back period, and if book is overdue, and fined amount by email. They can also log in and comment and rate books and get suggestions of similar books which students with similar courses have taken.
* There will be certain set of permissions, the Library staff account can add, remove, delete and view all books available and on loan, and also the fined amount, and can verify student’s accounts if required and adjust fines.
* The Registrar’s office staff or faculties accounts’ can add notice, delete notice, schedule workshops, and exam dates online.
* The students can view books, and notices and see which books they borrowed and due date, and fined amount if any. They can also see what similar books students took and comment on those.
* There is also email verification and confirmation where appropriate (loaning books, creating account, due date, overdue).
* It can remain complete records of a file; it can be a documents, registers, books of any type etc.
* It can be appended while got any new arrivals.
* All of the user can find the way through the database and locate a document of their choice.
* A non-member or member can search a book without consulting library staff in a network environment, where in several PCs of different department which are connected.
* It has huge search facilities along with point search facility where, an exacting item/string in a given field of a specified period may be search at finger tips.
* Presented records can be edited only by an allowed person.
* Authorized person/administrator they can change the password at any time to prevent people who want to hack the password.
* New members can only be registered by the library administrator or certified person.
* Member's photograph and signature can be integrated and may be confirmed at the time of issuing any paper.
* The software can give the subsequent information at finger tips:
* The software will store the Issue or return records of any file.
* Then it also store the Issue or return records of a member.
* Details of all papers issued from the library on time.
* These will make the activities for our major stakeholders – faculties, Registrar’s office staff, Library staff, and students – quite streamlined, efficient, rapid, flexible and smooth , automating the entire process, requiring less manual work.



# 1.3 Overview of This Report

This report is the first part of multiple submissions of a complete report. It also includes with it a working functional prototype.

In the initial (previous) sections we have provided the background and information on our project, and our objectives and goals, which will make it clearer of the parts that are to come. It will make it more evident on why we are choosing to do the task as follows.

In the subsequent sections we provide a literature review, analysing what a basic management system does, and why our proposed system is more viable and desirable than the currently existing ones.

We go on to explain the development methodology we have used in our implementation and the basis for using it.

# 2. Literature Review

In this project we are working on Management System. A Management System includes a Library Management System (LMS) and an Online Notice Board System. A Library Management System (LMS) is a device to help any libraries and institution which are still using the old system to manage their library. The old way like searching for a book using manual work is scrap, fast report generation is not possible, information about issue/return of the books are not properly maintained, no central database can be created as information is not available in database. But by using the system, user can overcome all the problems mentioned above. This system can manage all the process of the library. Book connections including book searching, personal book borrowing history details, availability of the book and appearance of the book and etc. can be very easily handled by this system. This system is suitable for small to big libraries including legal libraries and medical, university, schools, colleges, academic resource centers and other corporate houses .However, we have focused on LMS for universities and colleges.

LMS has three modules that’s are Administration Module, Staff Module and Students Module. First of all, Students Module allows students to search material by title, subject, publication, name/author, ISBN/ISSN, series and etc. it also allow students to find recently arrived material.

In Staff Module, staff can issue book and receive book with bar code. Staff also can fine for books received after due date.

Lastly, the administration module help administrator to register/convey shelf and category, issue library card, change book category , configuration to register users, and database backup/restore.

The Online Notice Board will enable staff members to post and update notices online with ease. This will make information regarding the organization readily accessible to everyone.

Overall this project of management system can be very helpful and it can make things easier.

## Study on existing MS

## 2.1Overview

Nearly all of the users are like to use a system which are already long exists and already have a positive status. Those people whom towards ICT; they may like to compare with some existing systems before they select one, as well as Management System (LMS). In this report, quite a lot of features, functions and comparison among the both systems will be discussed to understand better how a MS works and its main principle to help out universities, businesses and companies.

**Advantages of our MS:**

* Simple and easy to use
* Online interface easy to reach
* Supports all platforms as web interface
* Mobile responsive
* Can create, read, update, delete notices, book information and notices with ease and instantly
* A modern approach to automation and improving existing systems to stay up to date with recent technologies
* Helps build the library students’ community and allows them to interact
* Improve efficiency and productivity
* Increased library commitment
* Highly secure
* With MS, it helps most businesses to increase efficiency on their daily work such as reducing man power to manage books or to post notices in the university notice board, and manual activities subsequently improve productivity.
* Save time
* Dynamic reports
* MS reduces manual activities, man power and paperwork. With this programmed system, people which have multiple skills can be used more efficiently.
* Error free
* Mobile access
* Innovation

# 3.Methodology

* In this project we have used the agile methodology for our development. We have adapted this for several reasons.
* It has a lower cost, as there is always a scope for correction and changing the implementation and carry out something better as the project grows.
* It enables our clients to take part in the development process and voice what they require from our program and taking their feedback into consideration to make the end product better and making them happier.
* It enables us to openly communicate with the clients, take their suggestions and visions, and also show them what other alternatives we have and how we can make it better by discussing amongst ourselves.
* Debugging and testing becomes easier as we are constantly checking for defects and errors and fixing them as we go, and not at the complete end of the project. Thus making us more competitive in the market as well.
* Evaluating and preparation time is sped up greatly as each iteration consists of a small section of the entire project, so we can finish it and then focus on the next rather than the whole project.
* Assessments are made quickly and the product is evaluated at every stage of the process so as to make sure we maintain the best quality standards required.
* It is easier to make sure that our product is meeting the requirements requested at every stage.
* Communication with clients and systems makes the process transparent.

### High Product Quality

* High product quality is ensured through regular testing to make sure the product is working during the development phase.
* Requirements can be defined and broadened upon in the appropriate time.
* It enables continuous integration and regular testing into the development process.
* The software gets developed in incremental, quick cycles.

### Higher Customer Satisfaction

* High customer satisfaction is ensured by showing the working functionalities to the clients.
* It also makes delivering to the market faster and more frequently with every release of the products.
* Customers are involved and engaged in the process making them more confident on our product and enabling us to understand them better.
* If anything is not favourable or accepted positively, it can be quickly changed in the next iteration.

### Increased Project control

* Agile process enables more control over the project through daily sprint meetings and transparency.

### Reduced Risks

* Risks are reduced as small incremental developments ensures a small period between feature development.
* Agile gives the luxury when newer changes need to be incorporated.
* The client’s needs and preferences can be adapted to in the development process.

### Faster ROI

* The Return Of Investment (ROI) is quicker as it is more business focused, allowing the client to focus on the major features first.
* A functional ready to use product is made after a few iterations thus more beneficial.
* Customer reaction can be gauged and built upon as agile provides fast product releases.

## Early and Predictable Delivery

* Delivery dates are more predictable and sooner as a fixed schedule of sprints or iterations are used. This also enables the opportunity to release a beta or test a prealpha version if a sufficient model is completed before the due date.

## Predictable Costs and Schedule

* As the iterations are meant to be completed in a fixed duration, the cost of each iteration is predictable and fixed to the amount of work that can be performed by the team in that time frame. This makes it easier to estimate a cost for the client prior to each iteration and how the costs are arising with respect to the features. Additional features can also be thought of here.

## Allows for Change

Every iteration allows changes to be made to the product easily making it grow and develop.

## Focuses on Business Value

Focusing on business value is a priority as the features are developed in every iteration based on priority, step by step going towards a minimum usable output, then making better.

## Focuses on Users

Users are given top priority and this ensures the product is useful and tries to meet their demands every stage of the way.

## Improves Quality

* Small iterations of a large development process makes it easier for the development team to focus on high-quality development, testing, and collaboration. Frequent builds and tests and reviews through each iteration ensures better quality and identifies and corrects defects earlier.
* Small teams work better using agile development process as on developer can possess multiple skills and can utilize them at several stages of the iteration without conflicts.
* Agile welcomes changing requirements, even late in development process. Agile processes harness change for the customer's competitive advantage.

# 4 Requirements

To help gather the functional and non-functional requirements for developing our application suite, we used a number of methods. Below is a description of some of the tools used to collect these information.

**1. Brainstorming**

It is used in requirements induction to collect decent number of ideas from a number of people. Typically brainstorming is implemented in finding out all the possible solutions to problems and makes the detail of opportunities simpler. It casts a broad net, finding out many discreet possibilities. Prioritization of these possibilities is important to find needles in haystack.

**2. Document Analysis**

Document Analysis is a vital information-gathering technique. Understanding the documentation of a present system can aid when creating AS-IS process documents and also when driving the gap analysis for scoping of the migration projects. In today’s world, we will also be finding out the requirements that initiated the creation of an existing system- a starting point for documenting all of the present requirements. Pieces of data are mainly buried in current documents that help us in setting questions as a part of validating the requirement completeness.

**3. Focus Group**

A focus group is in reality the gathering of people who are customers or users representatives for a product to gather their feedback. The feedback can be gathered about opportunities, needs, and problems to find out the requirements or it can be collected to clean and validate the already elicited requirements. This type of market research is contrary from brainstorming in which it is a managed process with certain participants.

**4. Interface Analysis**

Interface for any software product will either be human or machine. Integration with external devices and systems is another interface. The user centric design approaches are rather advantageous to make sure that we create usable software. Interface analysis- analyzing the touch points with another external system- is crucial to ensure that we do not overlook the requirements that are not readily visible to the users.

**5. Interview**

Interviews of users and stakeholders are vital in making a wonderful software. With no knowledge of the expectations and motivations of the stakeholders and users, we are highly unlikely to satisfy them. We also need to understand the perspective of every interviewee, in order to accurately address and weigh their inputs. Like a good reporter, listening is a quality that assists an excellent analyst to get better value through an interview as compared to an average analyst.

**6. Observation**

The observation covers the analysis of users in their natural habitat. By watching users, a process flow, pain points, awkward steps and opportunities can be determined by an analyst for betterment. Observation can either be active or passive. Passive observation provides better feedback to refine requirements; on the same hand active observation works best for obtaining an understanding over an existing business process. We use these approaches to uncover the implicit requirements that are often overlooked.

**7. Prototyping**

Prototyping can be very beneficial at collecting feedback. Low fidelity prototypes make a good listening tool. Often times, people are not able to articulate a specific need in the abstract. They can swiftly review whether a design approach would satisfy the need. Prototypes are very efficiently done with fast sketches of storyboards and interfaces. Prototypes in some cases are also used as official requirements.

**8. Requirements Workshop**

Also known as JAD or joint application design, these workshops are beneficial for collecting requirements. The requirements workshops are much more organized and structured than a brainstorming session where the involved parties get together to document requirements. Creation of domain model artifacts like activity programs or static diagrams is one of the many ways to capture the collaboration.

**9. Reverse Engineering**

Reverse Engineering current management system models available to generate our own customized model.

**10. Survey**

When collecting data from many people, it is often time and cost ineffective - too many people, too little time, too much expense taking interview. Thus a questionnaire survey can be used. The survey insists the users to choose from the given options (agree / disagree or rate something).

### 11. Request for proposals (RFPs)

We request for requirements through an RFP. This is a list of requirements that is there for us to compare against our own capabilities to determine how close a match we are to the client's needs.

### 12.Use cases

Use cases are simply stories that describe how individual processes function. The stories include people (actors) and describe how the solution works from a user perspective. Use cases may be easier for the users to articulate, although the use cases may need to be distilled later into the more specific detailed requirement

# 5 Analysis

**Management System Processes:**

In this section we discuss in depth about the Processes our Management System has.

In our model:

1. *The Staff consists of - the Library Staff, the Registrar’s Office Staff and the Faculty.*
2. *Types of Notice consists of - Holiday, Middle of Term Examination, Final Examination, Make-Up Class, Seminar, Workshop, Event, Competition*

*C) Authenticated Accounts consist of - Staff Account and Student Account.*

Our Online Management System can be accessed by going to https://www.domainname.org.

**Setting Up Account:**

I) **Sign Up** for Account Process:

1. *The Student or Staff goes to Site.*
2. *They navigate to the Sign Up page.*
3. *They put their personal details and University ID.*
4. *Verification email and SMS sent.*
5. *Confirmation of Staff account by Registrar’s Office Staff.*
6. *Signed Up.*

II) **Sign In** to Account Process:

1. *The Student or Staff goes to Site.*
2. *They navigate to the Sign In page.*
3. *They put their University ID and Password.*
4. *Signed In.*

**Permissions and Use Cases:**

I) All **Signed In User** can:

1. *View Notices on the Notice Board.*
2. *View Books available in the Library.*
3. *View Books they have Withdrawn.*
4. *View Due Date of returning Books.*
5. *View Fined Amount for Late Returning.*
6. *View Similar Books (people from similar Department also read)*
7. *View Ratings on Books.*
8. *View Comments on Books.*
9. *Add Comments on Books.*
10. *Give Ratings to Book.*

II) All **Staff Accounts** can:

1. *Add Notice to Message Board.*
2. *Delete Notice from Message Board.*
3. *Update Notice from Message Board.*

III) **Library Staff Accounts** can:

1. *Add Books to Library.*
2. *Remove Books from Library.*
3. *Update editions of Books in Library.*
4. *See list of Books Withdrawn by all* *Library Users.*
5. *See list of Books Withdrawn by a specific Library User.*
6. *See list of Overdue Books.*
7. *See list of Overdue Books by a specific Library User.*
8. *See Fined Amount on Books Overdue.*
9. *Verify Withdrawal and Return of Books and Fined Amount.*
10. *Verify Library Account.*
11. *Delete or Block Library Account.*

*IV)* **Registrar’s Office Staff Account** can:

*A) Verify* *Online Account.*

*B) Delete or Block Online Account.*

V) **All User Accounts**:

1. *Will receive E-mail Notifications for Notices or Library Updates.*
2. *Can change their Personal Details, Account Information and Password.*

VII) **Non Authenticated Users**:

*Cannot get access to any of these facilities.*

**Rich Picture:**

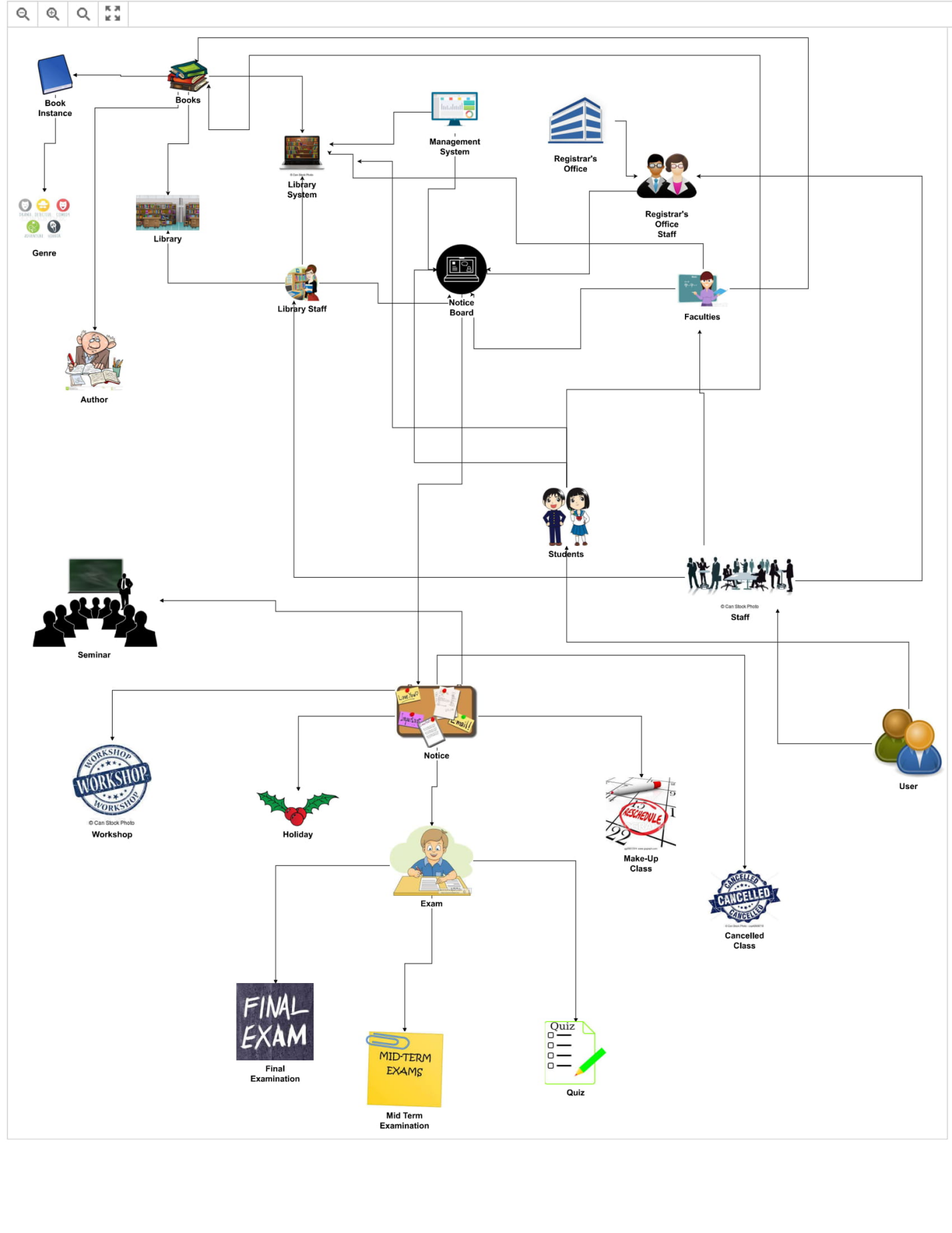
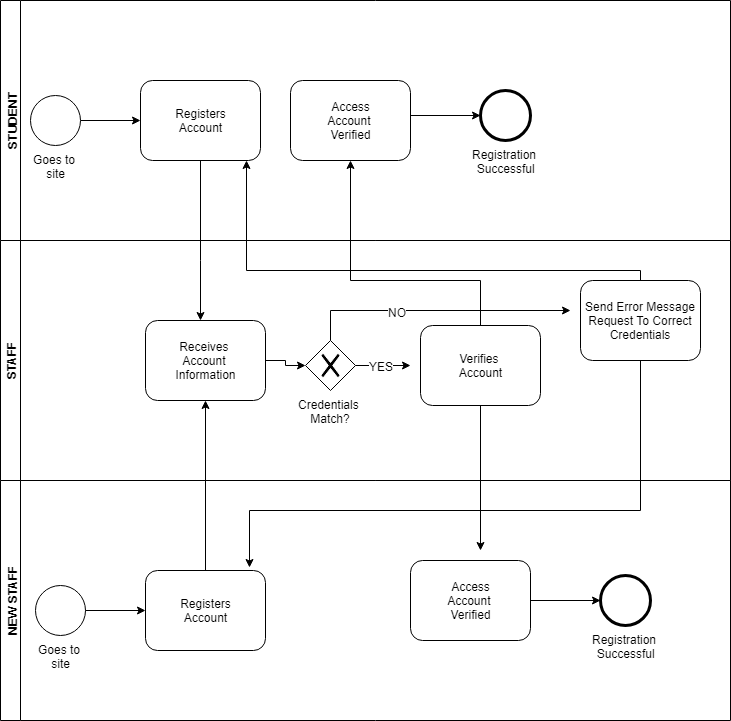


Table 1 Rich Picture of Management System

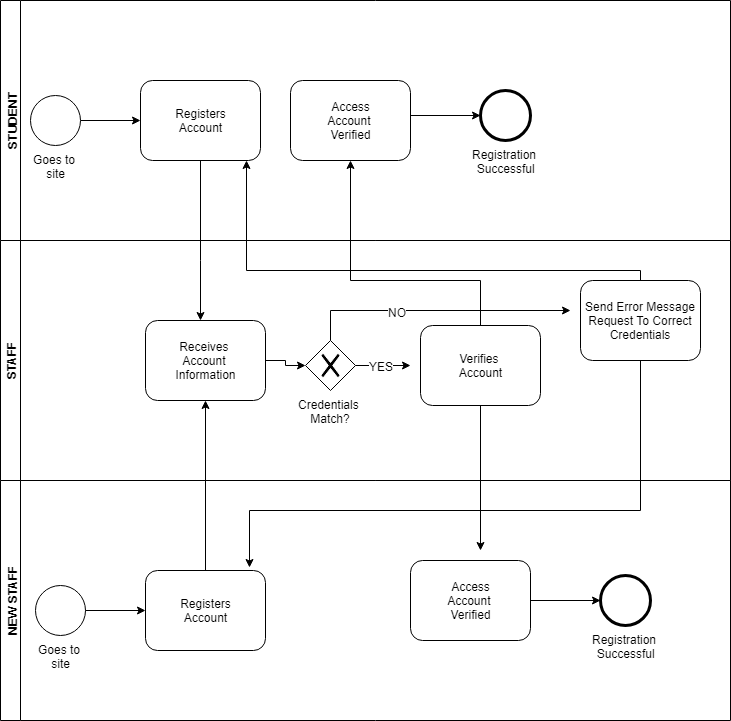
**Business Process Model Notation Diagrams:**

Due to the size and complexity of our project, and the number of possible processes involved in its usage and for the sake of time constraints, listed below are some of the BPMN diagrams for the major processes in our System.

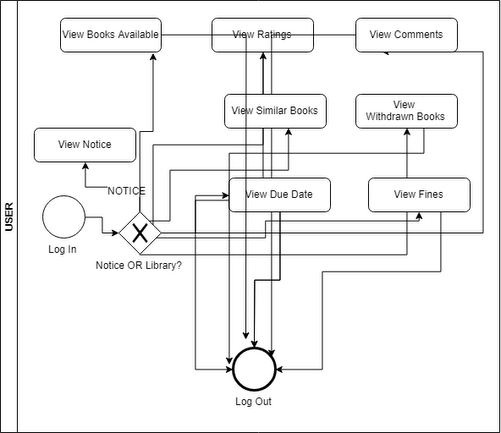
*Account Registration:*



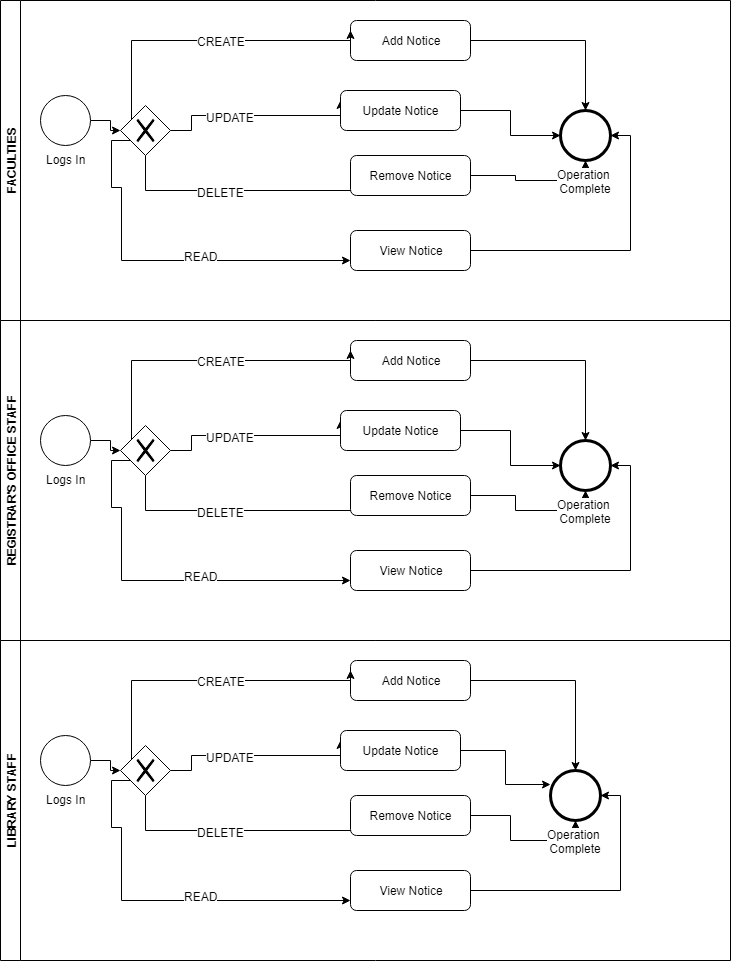
*Log In:*



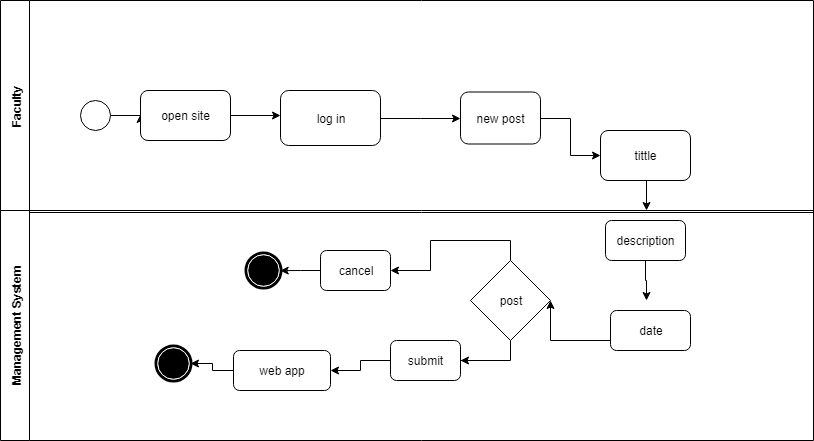
*View Books:*

**

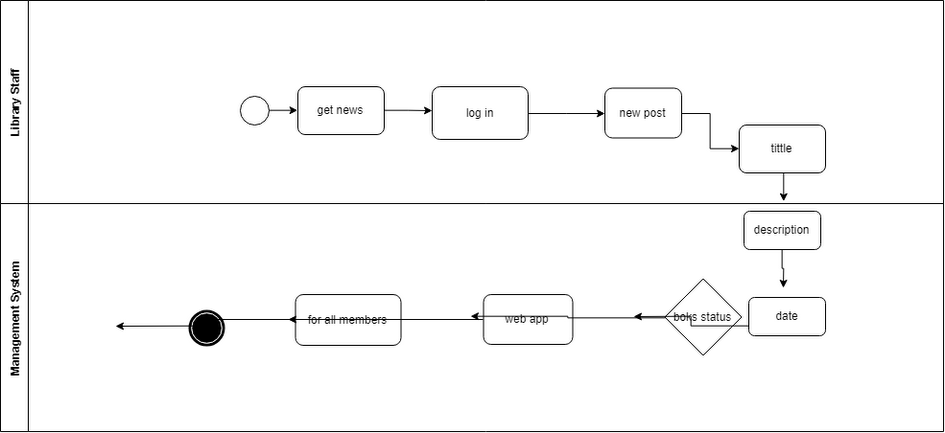
*Notice (Create - Read - Update - Destroy):*

**

*Faculty Post Notice:*

**

*Library Staff Post Notice:*

**

*Registrar’s Office Staff Post Notice:*

**

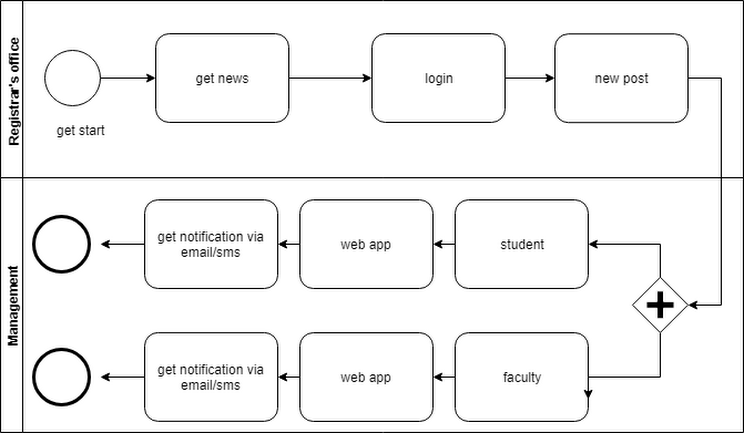
*Logout:*

**

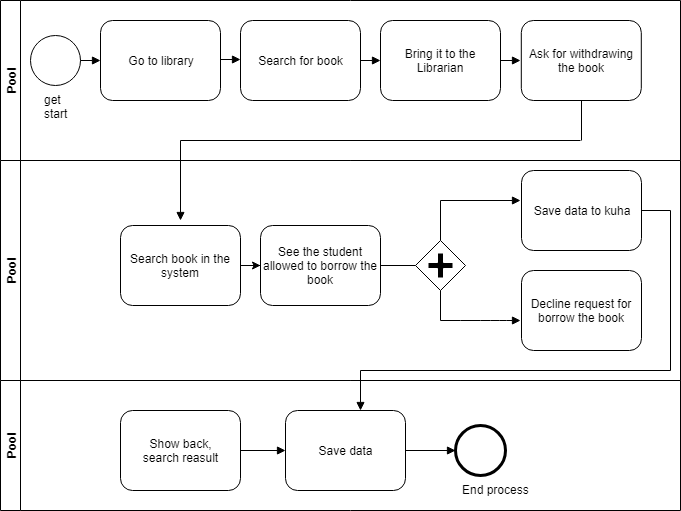
*Search Book:*

**

*Notifications:*

**

*Book Withdraw:*

**

# 6 Design

# Our design model stands out from the existing systems in the following ways:

# One stop Management Solution:

# Most Management System suits come separately for the individual components. But our model brings all the different components and packages them (Library Management System, Notice Management System) as a bundled solution.

# B) 2-3 Factor Authentication and Verification:

# Authentication is done in a 2-3 step process:

# Through E-mail verification

# Through SMS verification

# Through Staff verification

# This makes the System very secure.

# C) Passwords Hash Encrypted:

# The password is handled through several encryption and hashing algorithms so as it will not go astray if any flaws in the system were to occur, nor will it be easily penetrable by any hacker.

# D) Fast, responsive, web-based technology:

# The entire System is web-based, requiring no additional hardware to set up and can be set up from a remote location from the developers’ office.

# The front-end is User friendly and made to enhance the user experience, and also make the application readily accessible through virtually minimal loading time.

# It is also responsive and supports all platforms and devices (mobile, tablet, PC, etc)

# E) Notifications all the way:

# Our solution notifies the End-Users of any changes or activities that is being done under their name as they occur. This is done through E-mail and SMS notifications. This ensures greater security and more user-friendly experience.

# F) Go paper free:

# Having automatized most of the manual tasks of recording and submitting details by the staffs and students and much of the task being conducted over the web, it cuts down the use of paper.

# G) Advanced Search:

# We incorporated an advanced searching feature to make searching for books efficient and easy.

# Users can locate books with no trouble without the help of any librarian.

**Enhanced Entity Relation Diagram (EERD):**

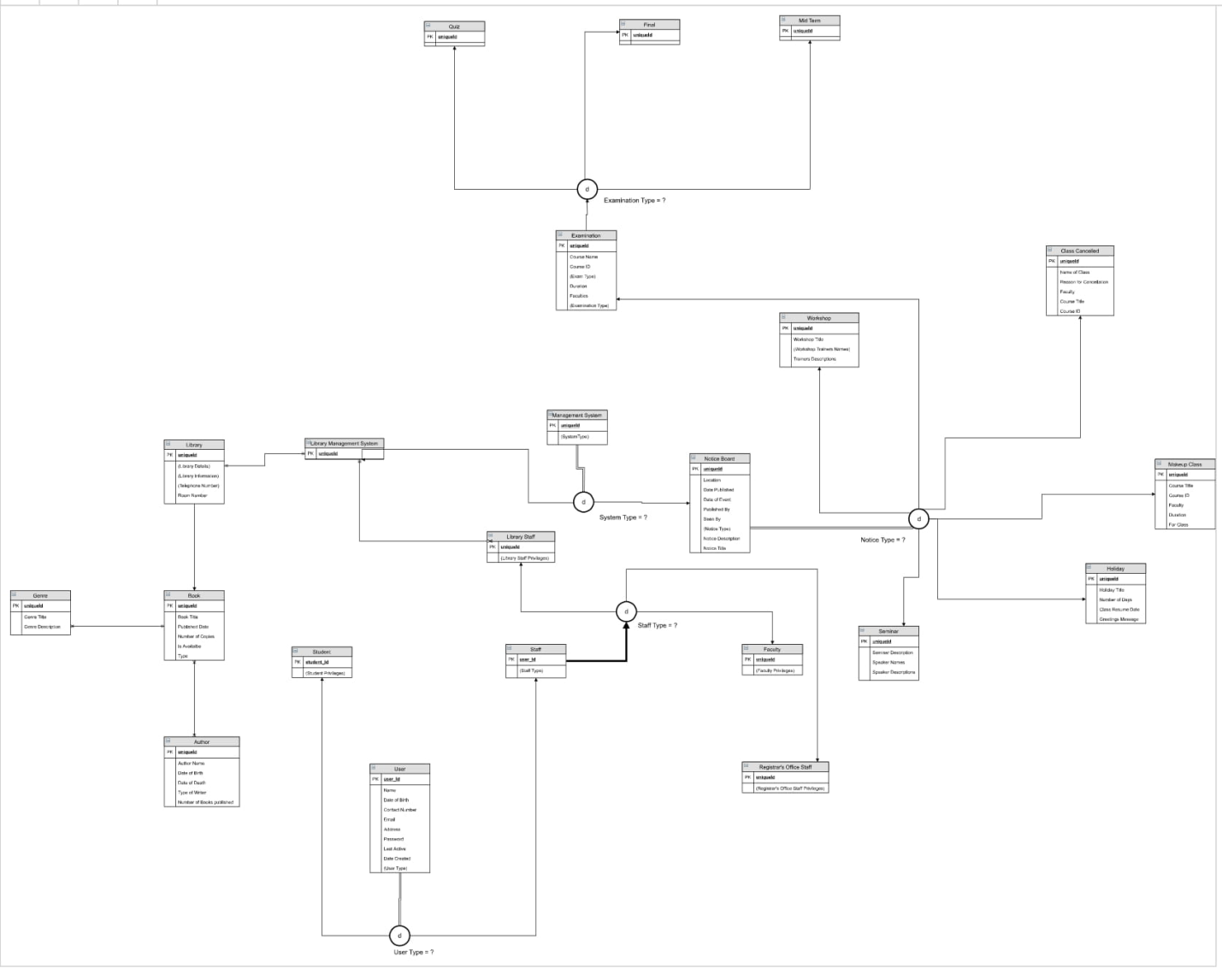


Table 2 Enhanced Entity Relation Diagram for Management System

**Data Dictionaries:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id(1,2,3,,4,5) | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Authentication Group. Ex: 1 |
| Name(students,library staff,faculties) | Text | 300 | Name of the Authentication Group.  Ex: Students |
| Message\_board\_notice |  |  | Contains a notice as a Foreign Key. |
| Auth\_user\_groups |  |  | Contains a Auth User Group as a Foreign Key. |
| Auth\_group\_permissions |  |  | This contains the Auth Group Permissions as a Foreign Key. |

AUTH\_GROUP

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Authentication Group Permissions.  Ex: 1 |
| group\_id | Number | Any positive number | Group ID from the Group Table as a Foreign Key.  Ex: 1 |
| permission\_id | Number | Any positive number | Permission ID from the Permission Table as a Foreign Key.  Ex: 1 |

AUTH\_GROUP\_PERMISSIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Content\_type\_id | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Authentication Permissions. Ex: 1 |
| Codename |  |  | Ex: add\_logentry,  Change\_logentry,add\_book |
| Name |  |  | Ex: can add log entry,can change log entry,can add group |
| Auth\_group\_permissions |  |  | This contains the Auth Group Permissions as a Foreign Key. |

AUTH\_PERMISSION

AUTH\_USER

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Authenticated User. Ex: 1 |
| Last\_login | Date-Time |  | This contains the Date and Time of the Last Login.  Ex: 2018-06-21 13:29:33.005402 |
| Is\_superuser | Boolean | Yes/No | This contains the Boolean information if the User is Superuser or not.  Ex: Yes,  No |
| Username | Text | 100 | This contains the Account’s User Name.  Ex: Mdadul Hakqe |
| First\_name | Text | 100 | This contains the User’s First Name.  Ex: Charles |
| Email | Email | 100 | This contains the User’s primary email.  Ex: Ratul\_Shams@yahoo.com |
| Is\_staff | Boolean | Yes/No | This contains the Boolean information if the User is staff or not.  Ex: Yes,  No |
| Is\_active | Boolean | Yes/No | This contains the Boolean information if the User Account is active or not.  Ex: Yes,  No |
| Data\_joined | Date | “YYYY-MM-DD HH:MM:SS, YYYY-MM-DD HH:MM:SS, “ | This contains the Date and Time of when the User Account was created.  Ex: 2018-06-21 13:29:33.005402 |
| Last\_name | Text | 100 | This contains the User’s Last Name.  Ex: Dickens |
| Account\_profile |  |  | This contains a Account Profile as a Foreign Key. |
| Django\_admin\_log |  |  | This is the admin log as a foreign key. |
| Catalogue\_Book\_instance |  |  | This contains the book instance as the Foreign Key. |
| Auth\_user\_groups |  |  | Contains a Auth User Group as a Foreign Key. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Authenticated User group. Ex: 1 |
| User\_id | Number | Any positive number | This contains the ID from the User Table as a Foreign Key.  Ex: 1 |
| Group\_id | Number | Any positive number | This contains the ID from the Group Table as a Foreign Key.  Ex: 1 |

AUTH\_USER\_GROUP

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Sql\_Query |  |  | SELECT \* FROM 'auth\_user\_user\_permissions' LIMIT 0,30 |

AUTH\_USER\_USER\_PERMISSIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id |  | Any positive number | This is the primary key of this relation. This contain the ID of the Author. Ex: 1 |
| First\_name | Text | 100 | This contains the First Name of the Author.  Ex: Charles |
| Last\_name | Text | 100 | This contains the Last Name of the Author.  Ex: Dickens |
| Data\_of\_birth | Date | “YYYY-MM-DD” | This contains the Date of Birth of the Author.  Ex: 2018-06-21 |
| Date\_of\_death | Date | “YYYY-MM-DD” | This contains the Date of Death of the Author.  Ex: 2018-06-21 |
| Catalog\_book |  |  | This contains the Book from the Book Table as a Foreign Key. |

CATALOG\_AUTHOR

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Book.  Ex: 1 |
| Summary | Text | 200 | This contains the text summary of the book.  Ex: ADD SUMMARY HERE |
| Isbn | Number | Standard ISBN number | This contains a standard ISBN number of the Book.  Ex: 4356457567523 |
| Author\_id | Number | Any positive number | This contains the ID from the Author Table as a Foreign Key.  Ex: 1 |
| Catalog\_bookinstance |  |  | This contains the Book Instance as the Foreign Key. |
| Catalog\_book\_genre |  |  | This contains the Book Genre as the Foreign Key. |

CATALOG\_BOOK

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Book Genre. Ex: 1 |
| Book\_id | Number | Any positive number | This contains the ID from the Book Table as a Foreign Key.  Ex: 1 |
| Genre\_id | Number | Any positive number | This contains the ID from the Genre Table as a Foreign Key.  Ex: 1 |
| Catalog\_book\_genre | Number | Any positive number | This contains the Book Genre as the Foreign Key. |

CATALOG\_BOOK\_GENRE

|  |  |  |  |
| --- | --- | --- | --- |
| Name  CATALOG\_BOOKINSTANCE | Data Type | Size | Remark |
| Id | Text | 100 | This is the primary key of this relation. This contain the ID of the Book Instance.  Ex: 235452acaaea4632ba4749f3cb401032 |
| Imprint | Boolean | Yes/No | This is a Boolean data if the book is imprint or not.  Ex: Yes,  No |
| Due\_back | Date | YYY-MM-DD | This contains the Due Back Date Time.  Ex: 2018-06-21 |
| Status | Text | A - Available,  W - Withdrawn,  M - Maintenance,  R - Reserved,  etc | This contains the Availability Status of a Book.  Ex: Available |
| Book\_id | Number | Any positive number | This contains the ID from the Book Table as a Foreign Key.  Ex: 1 |
| Borrower\_id | Number | Any positive number | This contains the ID from the User Table as a Foreign Key.  Ex: 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id(1,2) | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Genre. Ex: 1 |
| Name(horror,comedy) | Text | 50 | This is the Name of the Genre.  Ex: Horror,  Comedy |

CATALOG\_GENRE

|  |  |  |  |
| --- | --- | --- | --- |
| Name  DJANGO\_ADMIN\_LOG | Data Type | Size | Remark |
| Id | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Admin Log. Ex: 1 |
| Object\_id | Number | Any positive number | This contains the ID from the Object Table as a Foreign Key.  Ex: 1 |
| Object\_repr | Text | 1000 | This contains the object string.  Ex: kumar,Anil,  Comedy,horror |
| Action\_flag | Number | Predefined set of numbers. | This contains the action flag.  Ex: 1 |
| Change\_message | Text | 1000 | This contains the change message description.  Ex: [{"added": {}}], [{"added": {}}] |
| Content\_type\_id | Number | Any positive number | This contains the ID from the Content Type Table as a Foreign Key.  Ex: 1 |
| User\_id | Number | Any positive number | This contains the ID from the User Table as a Foreign Key.  Ex: 1 |
| Action\_time | Date-Time | “YYYY-MM-DD HH:MM:SS, YYYY-MM-DD HH:MM:SS, “ | This contains the Action Date Time.  Ex: 2018-06-21 19:07:13.182173,  2018-06-26 13:22:36.450456 |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Content Type.  Ex: 1 |
| App\_label |  |  | This contains the app label.  Ex: admin |
| Model |  |  | This contains the model.  Ex: group |

DJANGO\_CONTENT\_TYPE

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id(1,2,3,4) | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Migration.  Ex: 1 |
| App(contenttypes,auth,admin) |  |  | This contains the apps.  Ex: contenttypes,  auth,  admin |
| Name | Text | “NNNN\_MIGRATION-NAME” | This contains the autogenerated Name of the Migration.  Ex: 0001\_initial,  0001\_initial |
| Applied | Date-Time | “YYYY-MM-DD HH:MM:SS, YYYY-MM-DD HH:MM:SS, “ | This contains the Migration Applied Date.  Ex: 2018-06-21 19:07:13.182173,  2018-06-26 13:22:36.450456 |

DJANGO\_MIGRATIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Session\_key | Text | 300 | This contains the Session Key.  Ex: m69agka8ui8gyl0wrnqus8fa3oz2mhp5,  seawglfxjmaggb7c6v2iel6hzchj3jpp |
| Session\_data |  |  | This contains the Session Data. |
| Expire\_data(2018-06-21 19:07:13.182173,2018-06-26 13:22:36.450456) | Date-Time | “YYYY-MM-DD HH:MM:SS, YYYY-MM-DD HH:MM:SS, “ | This contains the Expire Data.  Ex: 2018-06-21 19:07:13.182173,  2018-06-26 13:22:36.450456 |

DJANGO\_SESSION

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id (1,2,3) | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Notice.  Ex: 1 |
| Notice\_title | Text | 300 | This contains the Title of the Notice.  Ex: Mid Term Examination |
| Notice\_text | Text | 1000 | This contains the Description of the Notice.  Ex: Due to Durga Puga |
| Published\_date | Date | “DD/MM/YYYY” | This contains the Published Date of the Notice.  Ex: 01/01/2001 |
| Event\_date | Date | “DD/MM/YYYY” | This contains the Event Date of the Notice.  Ex: 01/01/2001 |
| Event\_kind | Text | 1 | This contains the Event Type of the Notice.  Ex: n - Notice, m - Month, h - Holiday |
| Posted\_by\_id | Number | Any positive number | This contains the Group ID - from the Group Table as a Foreign Key - that posted the Notice.  Ex: 1 |

MESSAGE\_BOARD\_NOTICE

SQILITE\_SEQUENCE

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Name |  |  | This contains name of sqlite sequence.  Ex: django\_migrations,  Django\_admin\_log |
| Seq |  |  | This contains the sequence number.  Ex: 22 |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Size | Remark |
| Id (1) | Number | Any positive number | This is the primary key of this relation. This contain the ID of the Profile.  Ex: 1 |
| Date\_of\_birth (2018-06-21) | Date | ‘’DD-MM-YYYY” | This contains date of birth of the User.  Ex: 29/01/81 |
| photo | Image | .jpg, or any well known image format | This contains the User Account’s Profile Picture. |
| User\_id (1) | Number | Any positive number | This contains the User ID from Users Table as a Foreign Key.  Ex: 1 |

ACCOUNT\_PROFILE

# 7 Implementation

We have used state of the art production-ready long term supported tools to implement our project.

**To Set Up:**

**Hardware Requirements:**

Note that - Our Web Application can be set up remotely from Team:Byte.M3’s office.

Processor: Pentium Core i 3

RAM: 16 GB

SSD: 16 GB

**Software Requirements:**

OS: Linux Cinnamon Mint (Virtual Machine)

VMWare Player

**Dependencies:**

Back-End Dependencies:

Python 3.7

Django 2.0

PILlow (Python Imaging Library)

Sqlite3

My SQL

PostgreSQL

Django-Debug Toolbar

Django Extensions

**Front-End Dependencies:**

HTML 5

CSS 3

Javascript

JQuery

Bootstrap 4

**Process to reverse engineer our software:**

1. We first set up a Linux Cinnamon Mint Virtual Machine on our SSD with the space allocated.
2. Then we created a directory for all our source codes named “src”.
3. In that directory, we set up a Python Virtual Environment Wrapper (Pipenv) and installed the Python dependencies in our Pipenv environment.
4. We activate our Pipenv environment.
5. We start a clean Django project in src directory named management\_system, using the following command:

Django-admin startproject management\_system

1. Then we create the library and notice board apps in the management\_system directory, using the following command:

Python3 manage.py startapp library

Python3 manage.py startapp notice\_board

1. We configure our Project’s settings.py file to:

I) Handle Static Files (Static files directory)

II) Handle Template Files (Template directory)

III) Handle URL Requests (URLs directory)

IV) Connect to our Database

H) Set up our App level URL files.

I) Create the required Models in both the Apps.

J) Create the required Views in both the Apps.

K) Create front-end HTML Template files for the End-User.

L) Make migrations to our Database.

M) Create superuser account.

N) Run Web Server Gateway Interface (WSGI) Server.

O) Have running software.

SQL Migration gulo dekhaite

**Django Models for Database:**

*User Accounts App:*

from django.db import models

from django.db import models

from django.conf import settings

class Profile(models.Model):

user = models.OneToOneField(settings.AUTH\_USER\_MODEL,

on\_delete=models.CASCADE)

date\_of\_birth = models.DateField(blank=True, null=True)

photo = models.ImageField(upload\_to='users/%Y/%m/%d/',

blank=True)

def \_\_str\_\_(self):

return 'Profile for user {}'.format(self.user.username)

from django.db import models

# Create your models here.

# users/models.py

from django.contrib.auth.models import AbstractUser

from django.db import models

class CustomUser(AbstractUser):

# First/last name is not a global-friendly pattern

# name = models.CharField(blank=True, max\_length=255)

# def \_\_str\_\_(self):

# return self.email

pass

*Library Models App:*

from django.db import models

from django.urls import reverse

import uuid

from django.contrib.auth.models import User

from datetime import date

class Genre(models.Model):

name = models.CharField(max\_length=200, help\_text='Enter a book genre (e.g. Science Fiction, French Poetry etc.)')

def \_\_str\_\_(self):

return self.name

class Book(models.Model):

title = models.CharField(max\_length=200)

author = models.ForeignKey(to='Author', on\_delete=models.SET\_NULL, null=True)

summary = models.TextField(max\_length=1000, help\_text='Enter a brief description of the book')

isbn = models.CharField(verbose\_name='ISBN', max\_length=13, help\_text='13 Character <a href="https://www.isbn-international.org/content/what-isbn">ISBN number</a>')

genre = models.ManyToManyField(to=Genre, help\_text='Select a genre for this book')

def \_\_str\_\_(self):

return self.title

def get\_absolute\_url(self):

return reverse(viewname='book-detail', args=[str(self.id)])

def display\_genre(self):

return ', '.join([ genre.name for genre in self.genre.all()[:3] ])

display\_genre.short\_description = 'Genre'

class BookInstance(models.Model):

id = models.UUIDField(primary\_key=True, default=uuid.uuid4, help\_text="Unique ID for this particular book across whole library")

book = models.ForeignKey(to='Book', on\_delete=models.SET\_NULL, null=True)

imprint = models.CharField(max\_length=200)

due\_back = models.DateField(null=True, blank=True)

LOAN\_STATUS = (

('m', 'Maintainance'),

('o', 'On Loan'),

('a', 'Available'),

('r', 'Reserved'),

)

status = models.CharField(max\_length=1, choices=LOAN\_STATUS, default='m', blank=True, help\_text='Book availability')

class Meta:

ordering = ['due\_back']

permissions = (("can\_mark\_returned", "Set book as returned"),)

def \_\_str\_\_(self):

return '{0} ({1})'.format(self.id, self.book.title)

borrower = models.ForeignKey(User, on\_delete=models.SET\_NULL, null=True, blank=True)

@property

def is\_overdue(self):

if self.due\_back and date.today() > self.due\_back:

return True

return False

class Author(models.Model):

first\_name = models.CharField(max\_length=100)

last\_name = models.CharField(max\_length=100)

date\_of\_birth = models.DateField(null=True, blank=True)

date\_of\_death = models.DateField(verbose\_name='Died', null=True, blank=True)

def get\_absolute\_url(self):

return reverse(viewname='author-detail', args=[str(self.id)])

def \_\_str\_\_(self):

return '%s, %s' % (self.last\_name, self.first\_name)

class Meta:

ordering = ['last\_name']

*Notice Board Models App:*

from django.db import models

# Create your models here.

from django.urls import reverse

# from datetime import date

# from django.contrib.auth.models import User

from django.contrib.auth.models import Group

class Notice(models.Model):

notice\_title = models.CharField(max\_length=200, help\_text='Enter a title for the notice (e.g. Make-Up Class.)')

notice\_text = models.TextField(help\_text='Enter a description for the notice.')

published\_date = models.DateField(verbose\_name='Published on', null=True, blank=True)

event\_date = models.DateField(verbose\_name='Takes place on', null=True, blank=True)

# posted\_by = models.ForeignKey(to=User, on\_delete=models.SET\_NULL, null=True,)Group

posted\_by = models.ForeignKey(to=Group, on\_delete=models.SET\_NULL, null=True,)

EVENT\_OPTIONS = (

('t', 'Tests Week'),

('m', 'Mid Term Examinations'),

('f', 'Final Examinations'),

('s', 'Seminar'),

('w', 'Workshop'),

('h', 'Holiday'),

('m', 'Make-Up Class'),

('n', 'Notice'),

)

event\_kind = models.CharField(max\_length=1, choices=EVENT\_OPTIONS, default='n', blank=True, help\_text='Notice Type')

def \_\_str\_\_(self):

return self.notice\_title

def get\_absolute\_url(self):

return reverse(viewname='notice-detail', args=[str(self.id)])

class Meta:

ordering = ['event\_date']

**Database Connection:**

*settings.py*

*# Database*

*# https://docs.djangoproject.com/en/2.0/ref/settings/#databases*

*DATABASES = {*

*'default': {*

*'ENGINE': 'django.db.backends.sqlite3', # Can use MySql, PosgreSql*

*'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),*

*‘USER’: ‘Scarlet’,*

*‘PASSWORD’: ‘1234ABCD!@#$’,*

*}*

*}*

**SQL Commands:**

**User Accounts SQL:**

BEGIN;

--

-- Create model Profile

--

CREATE TABLE "account\_profile" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "date\_of\_birth" date NULL, "photo" varchar(100) NOT NULL, "user\_id" integer NOT NULL UNIQUE REFERENCES "auth\_user" ("id") DEFERRABLE INITIALLY DEFERRED);

COMMIT;

-------------------------------------------------------------------------------------

**Library Models SQL:**

BEGIN;

--

-- Create model Author

--

CREATE TABLE "catalog\_author" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "first\_name" varchar(100) NOT NULL, "last\_name" varchar(100) NOT NULL, "date\_of\_birth" date NULL, "date\_of\_death" date NULL);

--

-- Create model Book

--

CREATE TABLE "catalog\_book" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "title" varchar(200) NOT NULL, "summary" text NOT NULL, "isbn" varchar(13) NOT NULL, "author\_id" integer NULL REFERENCES "catalog\_author" ("id") DEFERRABLE INITIALLY DEFERRED);

--

-- Create model BookInstance

--

CREATE TABLE "catalog\_bookinstance" ("id" char(32) NOT NULL PRIMARY KEY, "imprint" varchar(200) NOT NULL, "due\_back" date NULL, "status" varchar(1) NOT NULL, "book\_id" integer NULL REFERENCES "catalog\_book" ("id") DEFERRABLE INITIALLY DEFERRED);

--

-- Create model Genre

--

CREATE TABLE "catalog\_genre" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(200) NOT NULL);

--

-- Add field genre to book

--

CREATE TABLE "catalog\_book\_genre" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "book\_id" integer NOT NULL REFERENCES "catalog\_book" ("id") DEFERRABLE INITIALLY DEFERRED, "genre\_id" integer NOT NULL REFERENCES "catalog\_genre" ("id") DEFERRABLE INITIALLY DEFERRED);

CREATE INDEX "catalog\_book\_author\_id\_b0849980" ON "catalog\_book" ("author\_id");

CREATE INDEX "catalog\_bookinstance\_book\_id\_69f93415" ON "catalog\_bookinstance" ("book\_id");

CREATE UNIQUE INDEX catalog\_book\_genre\_book\_id\_genre\_id\_d15f6922\_uniq ON "catalog\_book\_genre" ("book\_id", "genre\_id");

CREATE INDEX "catalog\_book\_genre\_book\_id\_e5a77c43" ON "catalog\_book\_genre" ("book\_id");

CREATE INDEX "catalog\_book\_genre\_genre\_id\_77d7ffde" ON "catalog\_book\_genre" ("genre\_id");

COMMIT;

-------------------------------------------------------------------------------------

BEGIN;

--

-- Change Meta options on author

--

--

-- Add field borrower to bookinstance

--

ALTER TABLE "catalog\_bookinstance" RENAME TO "catalog\_bookinstance\_\_old";

CREATE TABLE "catalog\_bookinstance" ("borrower\_id" integer NULL REFERENCES "auth\_user" ("id") DEFERRABLE INITIALLY DEFERRED, "id" char(32) NOT NULL PRIMARY KEY, "imprint" varchar(200) NOT NULL, "due\_back" date NULL, "status" varchar(1) NOT NULL, "book\_id" integer NULL REFERENCES "catalog\_book" ("id") DEFERRABLE INITIALLY DEFERRED);

INSERT INTO "catalog\_bookinstance" ("book\_id", "status", "imprint", "due\_back", "id", "borrower\_id") SELECT "book\_id", "status", "imprint", "due\_back", "id", NULL FROM "catalog\_bookinstance\_\_old";

DROP TABLE "catalog\_bookinstance\_\_old";

CREATE INDEX "catalog\_bookinstance\_borrower\_id\_0d71c37c" ON "catalog\_bookinstance" ("borrower\_id");

CREATE INDEX "catalog\_bookinstance\_book\_id\_69f93415" ON "catalog\_bookinstance" ("book\_id");

COMMIT;

-------------------------------------------------------------------------------------

BEGIN;

--

-- Change Meta options on bookinstance

--

COMMIT;

-------------------------------------------------------------------------------------

**Notice Board Models SQL:**

BEGIN;

--

-- Create model Notice

--

CREATE TABLE "message\_board\_notice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "notice\_title" varchar(200) NOT NULL, "notice\_text" text NOT NULL, "published\_date" date NULL, "event\_date" date NULL);

COMMIT;

-------------------------------------------------------------------------------------

BEGIN;

--

-- Add field event\_kind to notice

--

ALTER TABLE "message\_board\_notice" RENAME TO "message\_board\_notice\_\_old";

CREATE TABLE "message\_board\_notice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "event\_kind" varchar(1) NOT NULL, "notice\_title" varchar(200) NOT NULL, "notice\_text" text NOT NULL, "published\_date" date NULL, "event\_date" date NULL);

INSERT INTO "message\_board\_notice" ("notice\_title", "notice\_text", "event\_kind", "published\_date", "id", "event\_date") SELECT "notice\_title", "notice\_text", 'n', "published\_date", "id", "event\_date" FROM "message\_board\_notice\_\_old";

DROP TABLE "message\_board\_notice\_\_old";

COMMIT;

-------------------------------------------------------------------------------------

BEGIN;

--

-- Add field posted\_by to notice

--

ALTER TABLE "message\_board\_notice" RENAME TO "message\_board\_notice\_\_old";

CREATE TABLE "message\_board\_notice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "posted\_by\_id" integer NULL REFERENCES "auth\_user" ("id") DEFERRABLE INITIALLY DEFERRED, "notice\_title" varchar(200) NOT NULL, "notice\_text" text NOT NULL, "published\_date" date NULL, "event\_date" date NULL, "event\_kind" varchar(1) NOT NULL);

INSERT INTO "message\_board\_notice" ("published\_date", "event\_date", "event\_kind", "posted\_by\_id", "notice\_text", "notice\_title", "id") SELECT "published\_date", "event\_date", "event\_kind", NULL, "notice\_text", "notice\_title", "id" FROM "message\_board\_notice\_\_old";

DROP TABLE "message\_board\_notice\_\_old";

CREATE INDEX "message\_board\_notice\_posted\_by\_id\_dbe98196" ON "message\_board\_notice" ("posted\_by\_id");

COMMIT;

-------------------------------------------------------------------------------------

BEGIN;

--

-- Remove field posted\_by from notice

--

ALTER TABLE "message\_board\_notice" RENAME TO "message\_board\_notice\_\_old";

CREATE TABLE "message\_board\_notice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "notice\_title" varchar(200) NOT NULL, "notice\_text" text NOT NULL, "published\_date" date NULL, "event\_date" date NULL, "event\_kind" varchar(1) NOT NULL);

INSERT INTO "message\_board\_notice" ("notice\_text", "notice\_title", "event\_date", "event\_kind", "id", "published\_date") SELECT "notice\_text", "notice\_title", "event\_date", "event\_kind", "id", "published\_date" FROM "message\_board\_notice\_\_old";

DROP TABLE "message\_board\_notice\_\_old";

COMMIT;

-------------------------------------------------------------------------------------

BEGIN;

--

-- Add field posted\_by to notice

--

ALTER TABLE "message\_board\_notice" RENAME TO "message\_board\_notice\_\_old";

CREATE TABLE "message\_board\_notice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "posted\_by\_id" integer NULL REFERENCES "auth\_group" ("id") DEFERRABLE INITIALLY DEFERRED, "notice\_title" varchar(200) NOT NULL, "notice\_text" text NOT NULL, "published\_date" date NULL, "event\_date" date NULL, "event\_kind" varchar(1) NOT NULL);

INSERT INTO "message\_board\_notice" ("id", "event\_date", "notice\_title", "notice\_text", "event\_kind", "posted\_by\_id", "published\_date") SELECT "id", "event\_date", "notice\_title", "notice\_text", "event\_kind", NULL, "published\_date" FROM "message\_board\_notice\_\_old";

DROP TABLE "message\_board\_notice\_\_old";

CREATE INDEX "message\_board\_notice\_posted\_by\_id\_dbe98196" ON "message\_board\_notice" ("posted\_by\_id");

COMMIT;

-------------------------------------------------------------------------------------------------------------------------------

We have used an agile development process to complete this project. Each member of the Team had specific activities, which they worked on, an made the individual components of our Web Application. And connecting all the moving parts together gave live to the final output.

We went back and forth, having several iterations, in which we tried fixing and enhancing our Application in increments.

**Front-End:**

At the time of submission, the front-end web pages have placeholder ‘dummy text’ which will need to be changed and filled as per the system in question by the company.

Some of the front-end urls have not been fully routed due to time constraints, although the components to route them are present.

Other than that, the front-end is fully functional, and responsive, catering itself to all types of platforms and screen sizes (mobile, PC, tablet).

All static files have also been partially minified and made to support faster loading.

The animations and load times are also efficient and smooth.

**Back-End:**

The backend of our system is a fully functional prototype, and for a medium to small sized company, it is fully functional.

It has multiple database support (Sqlite3, MySQL, PostgreSQL).

It can route requests and render proper response to the end user.

The application is also quite secure and prone to penetration or malpractice.

Issues might arise if several asynchronous requests are made to the database at the same time (as maybe the case for a large company) but as these rarely occur in a medium to small sized company, it shall not be a issue there. For large companies we will need to set up our database to handle asynchronous requests.

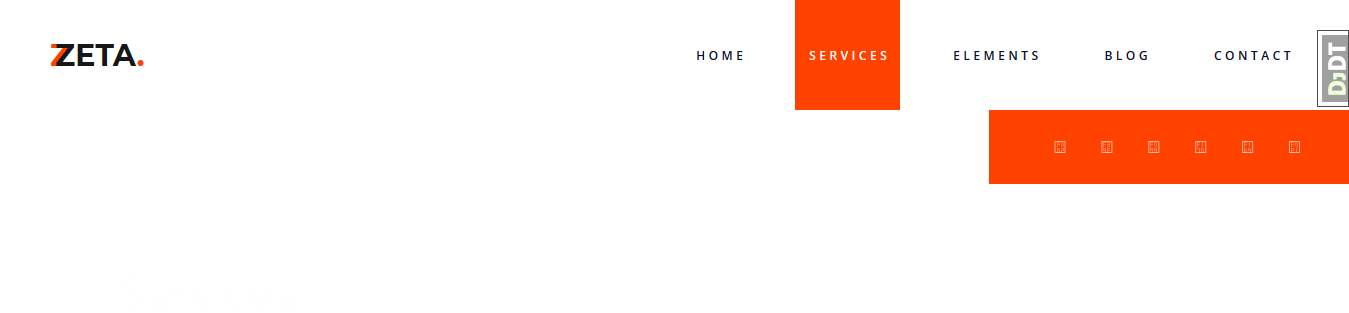
**Deployment:**

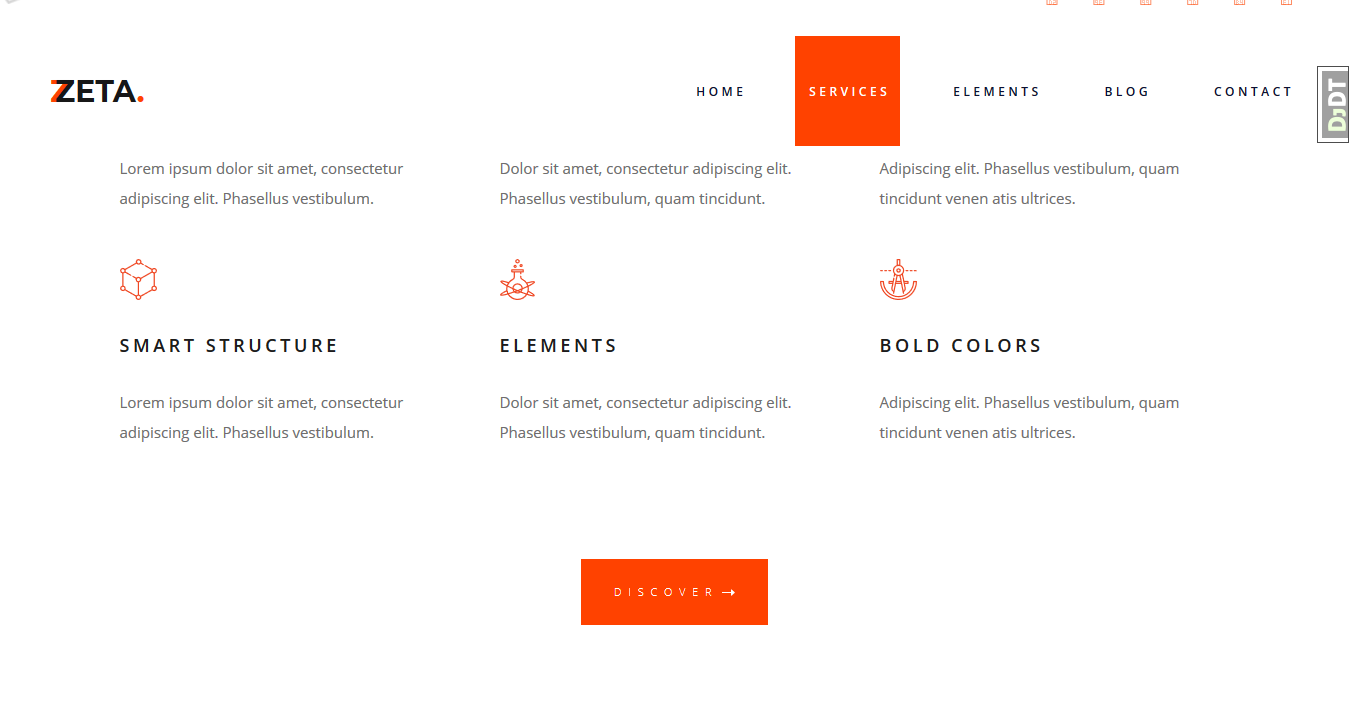
Due to the time and financial constraint, we have not been able to pay for a web hosting and deploy our website live.

To host our site, there are several Python server supporting hosts, one of the cheapest, most popular and efficient one being PythonAnywhere (<https://www.pythonanywhere.com/pricing/>) which costs $12/month.

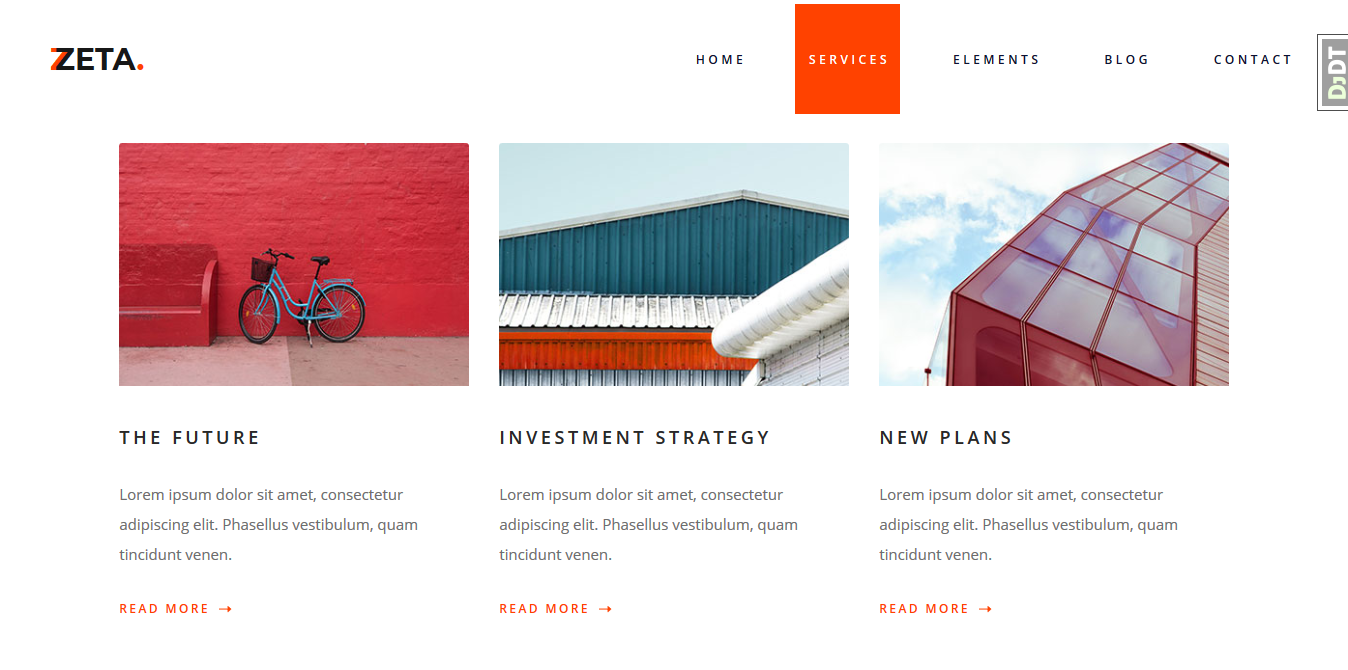
# 8Testing

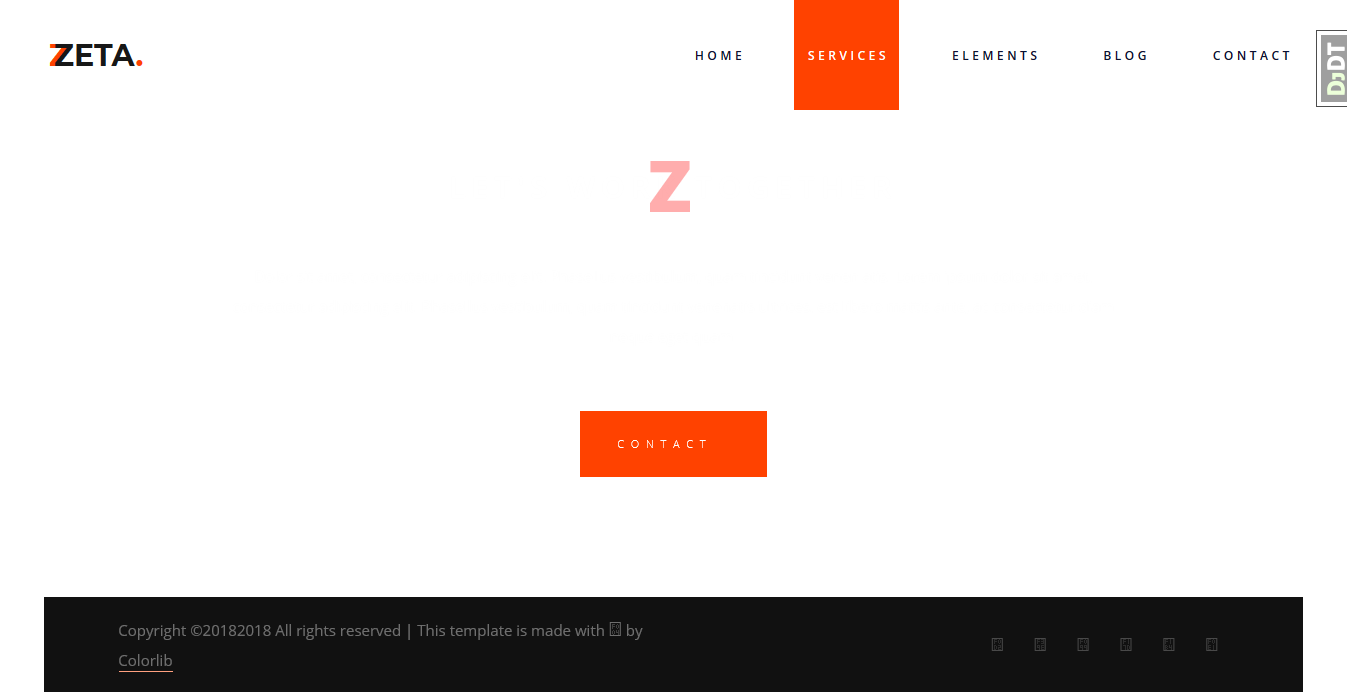
This section shows the main parts of our web application.

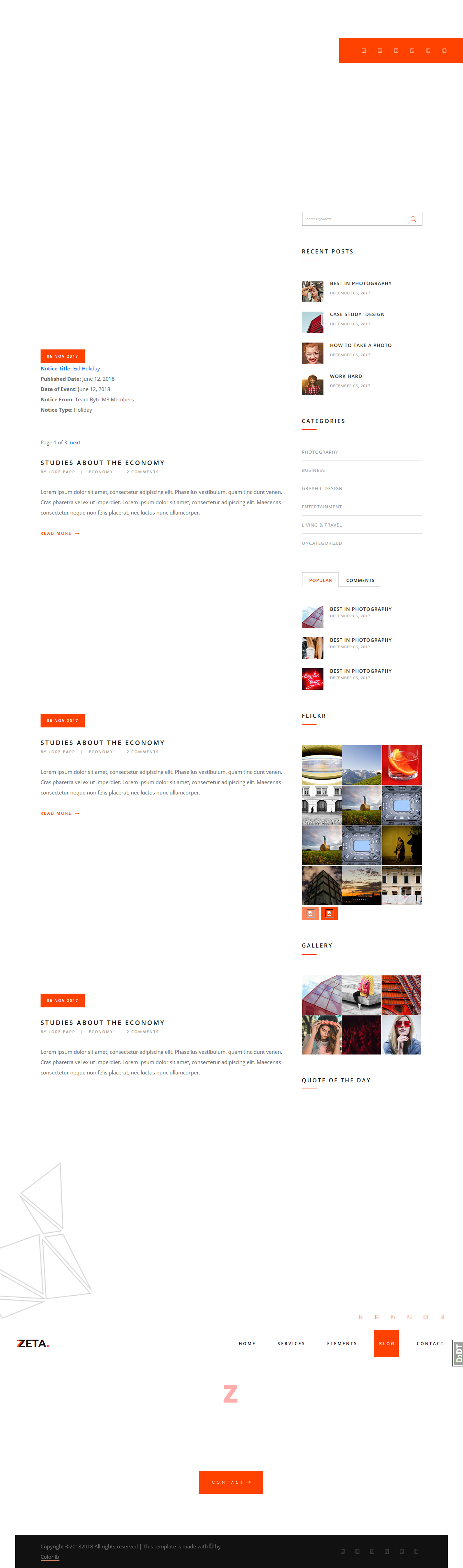


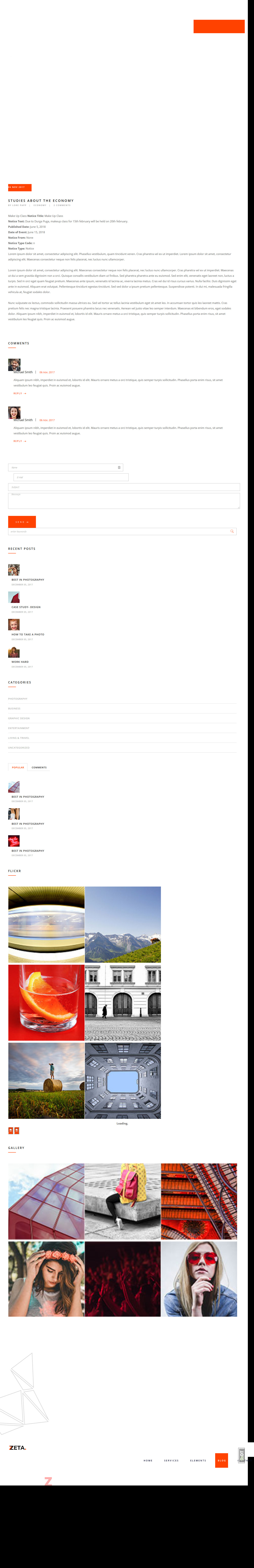


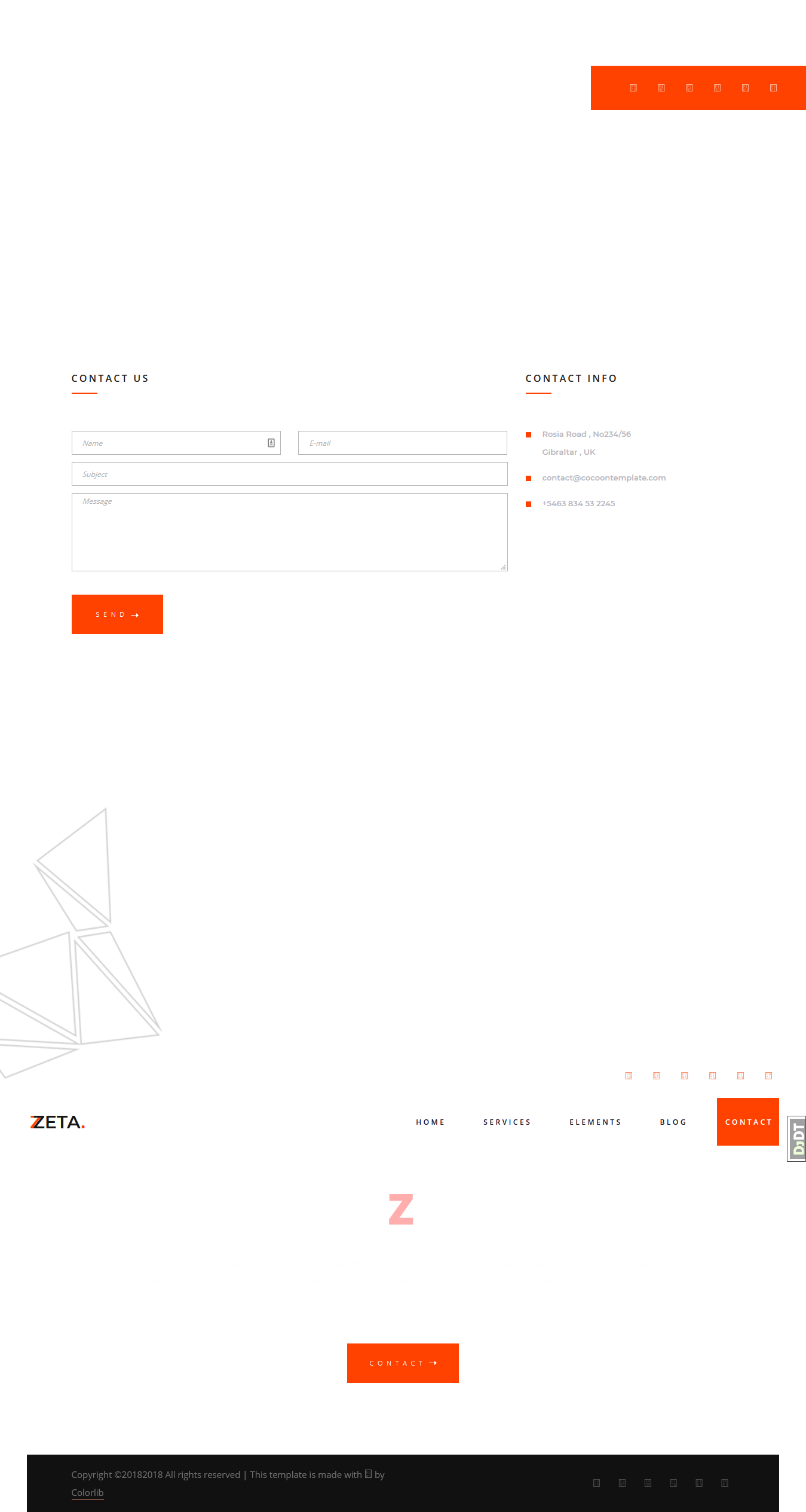


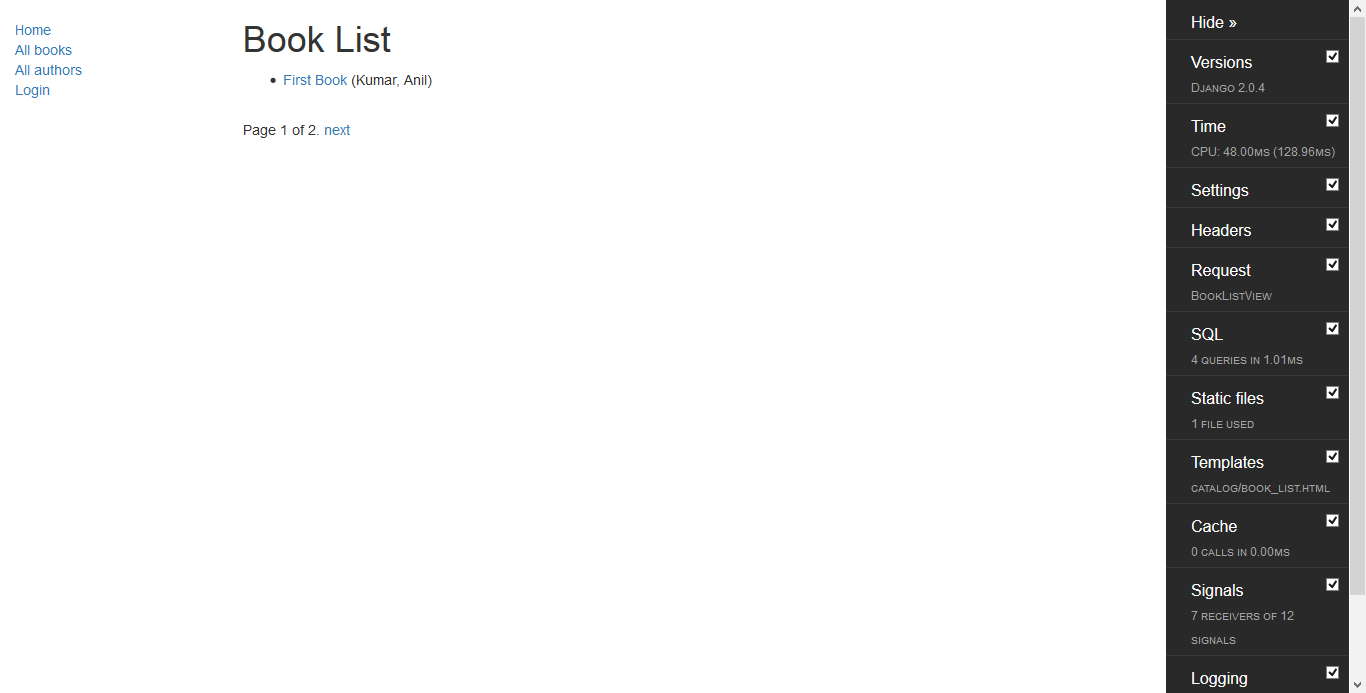


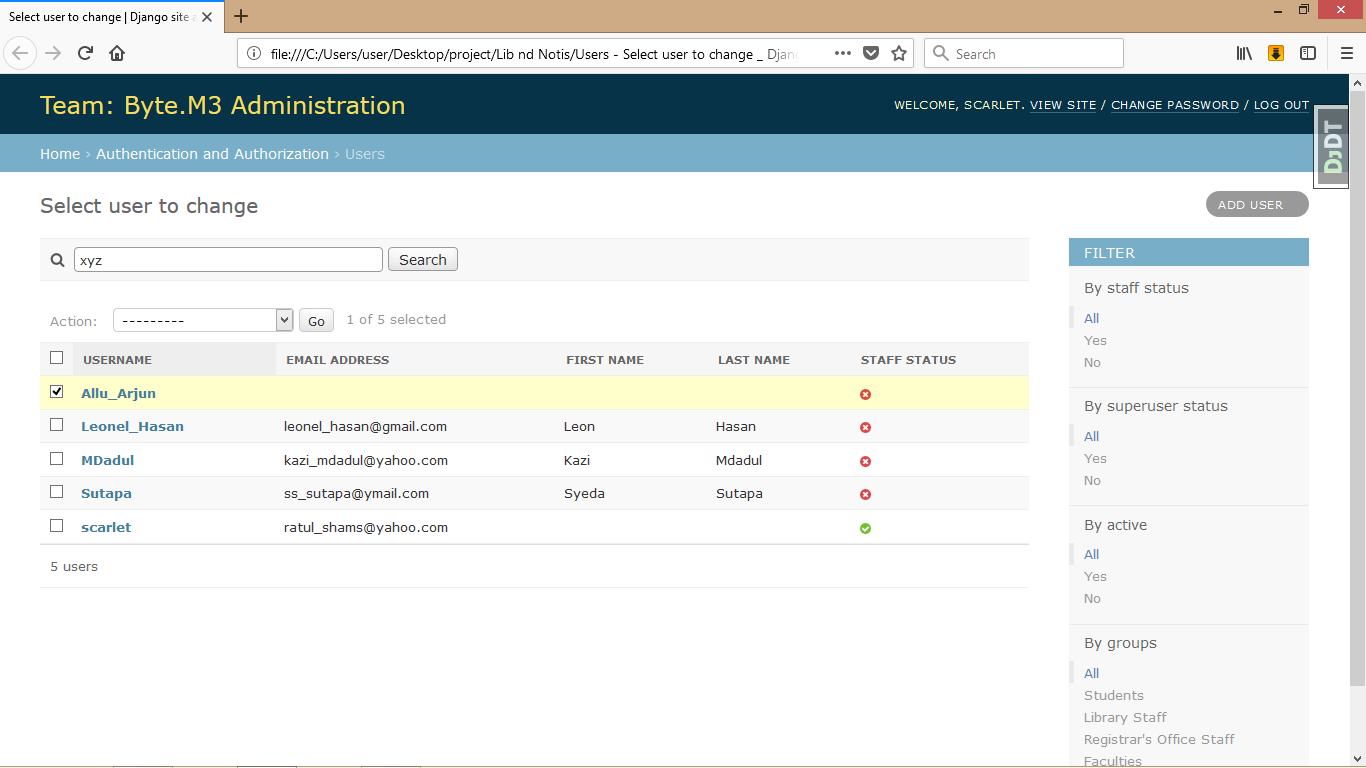


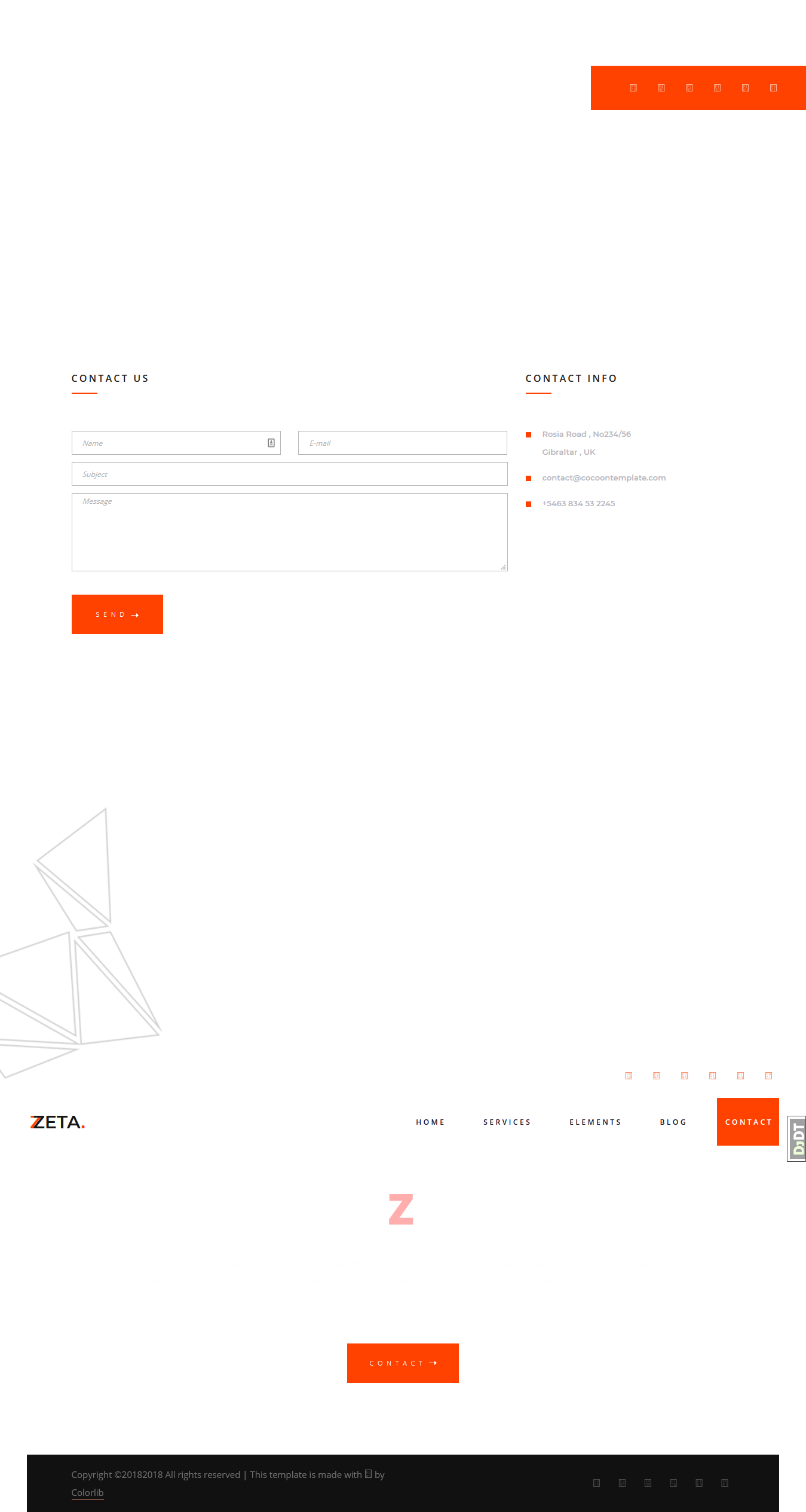


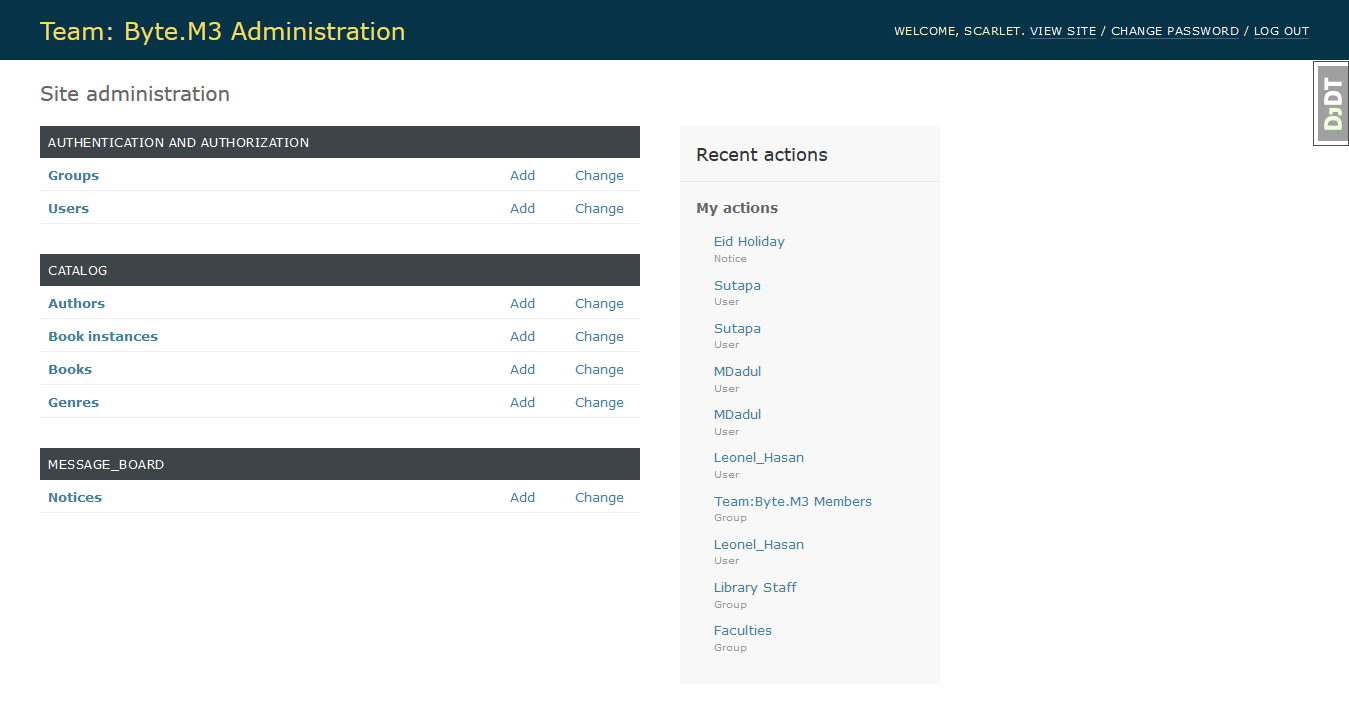


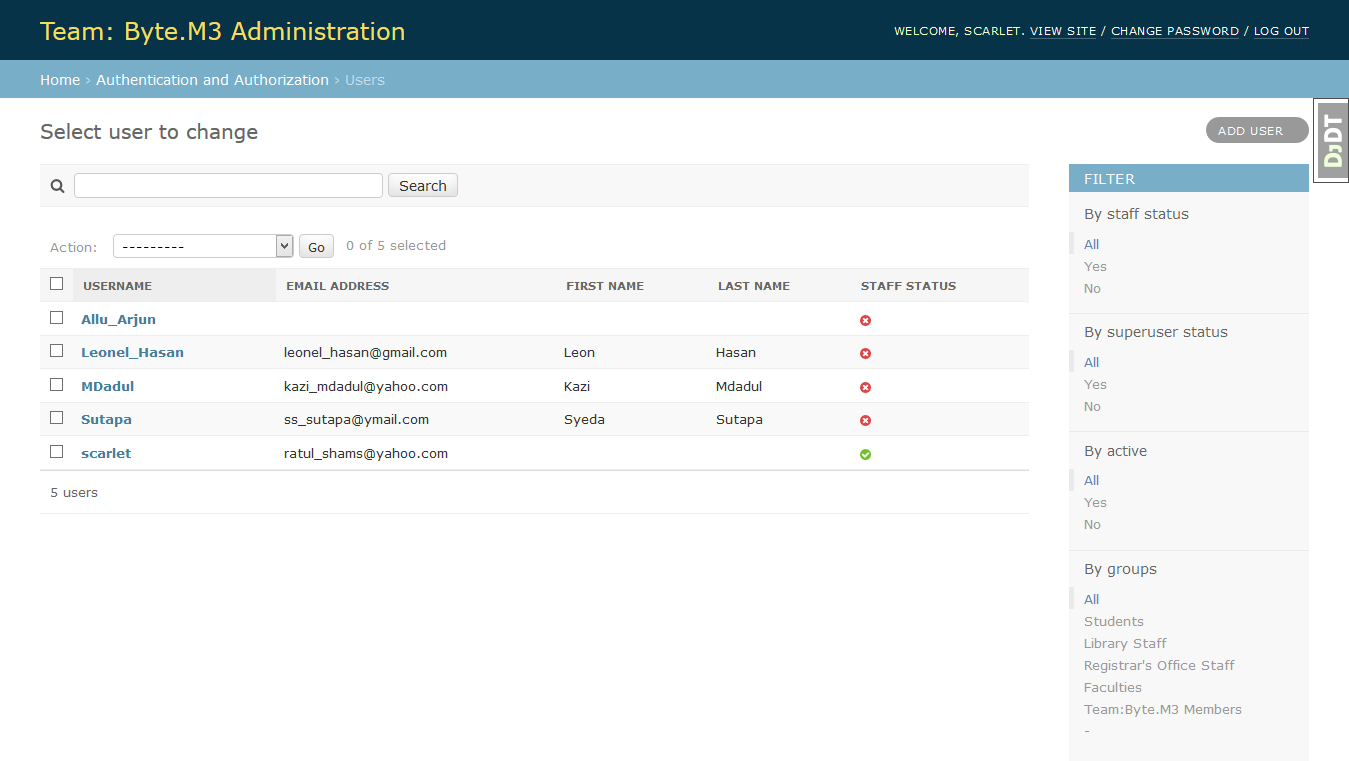


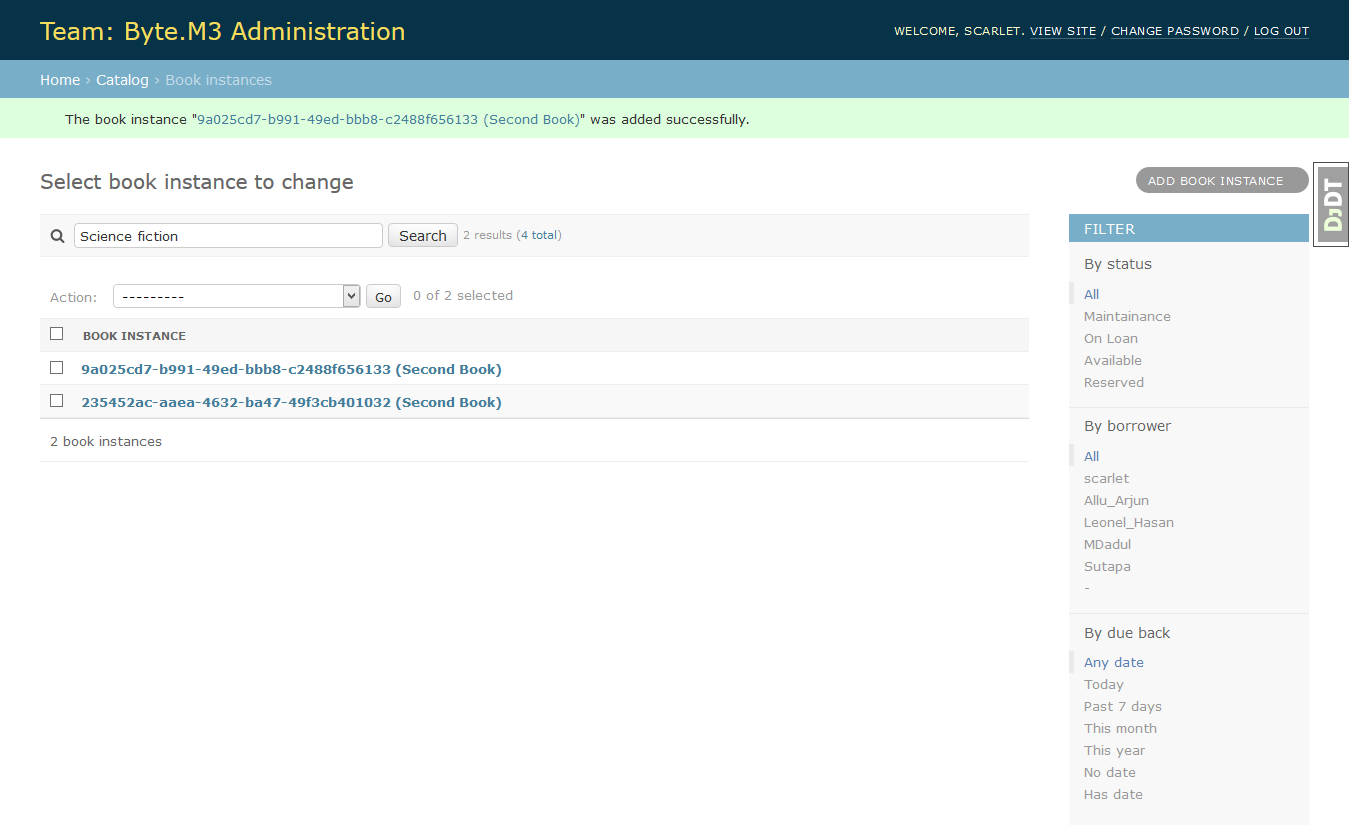


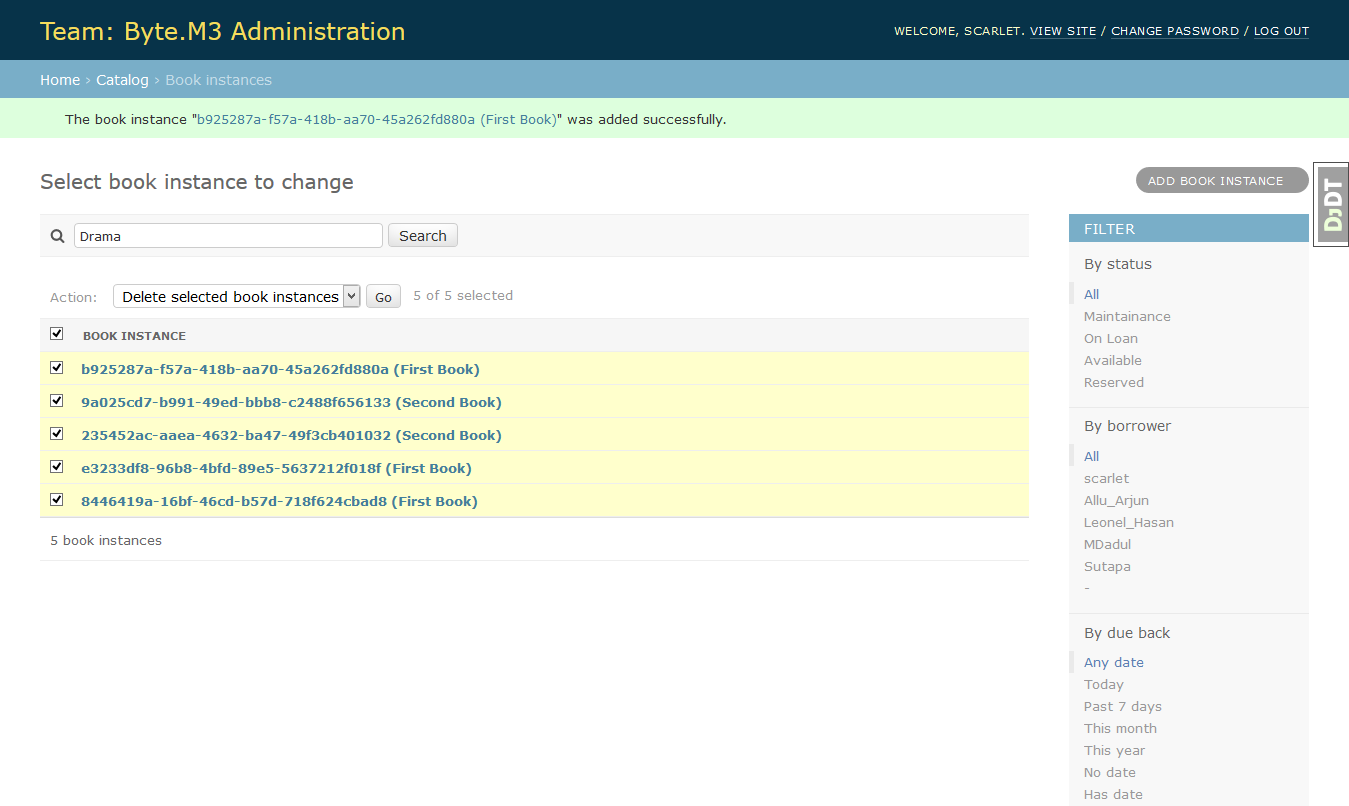


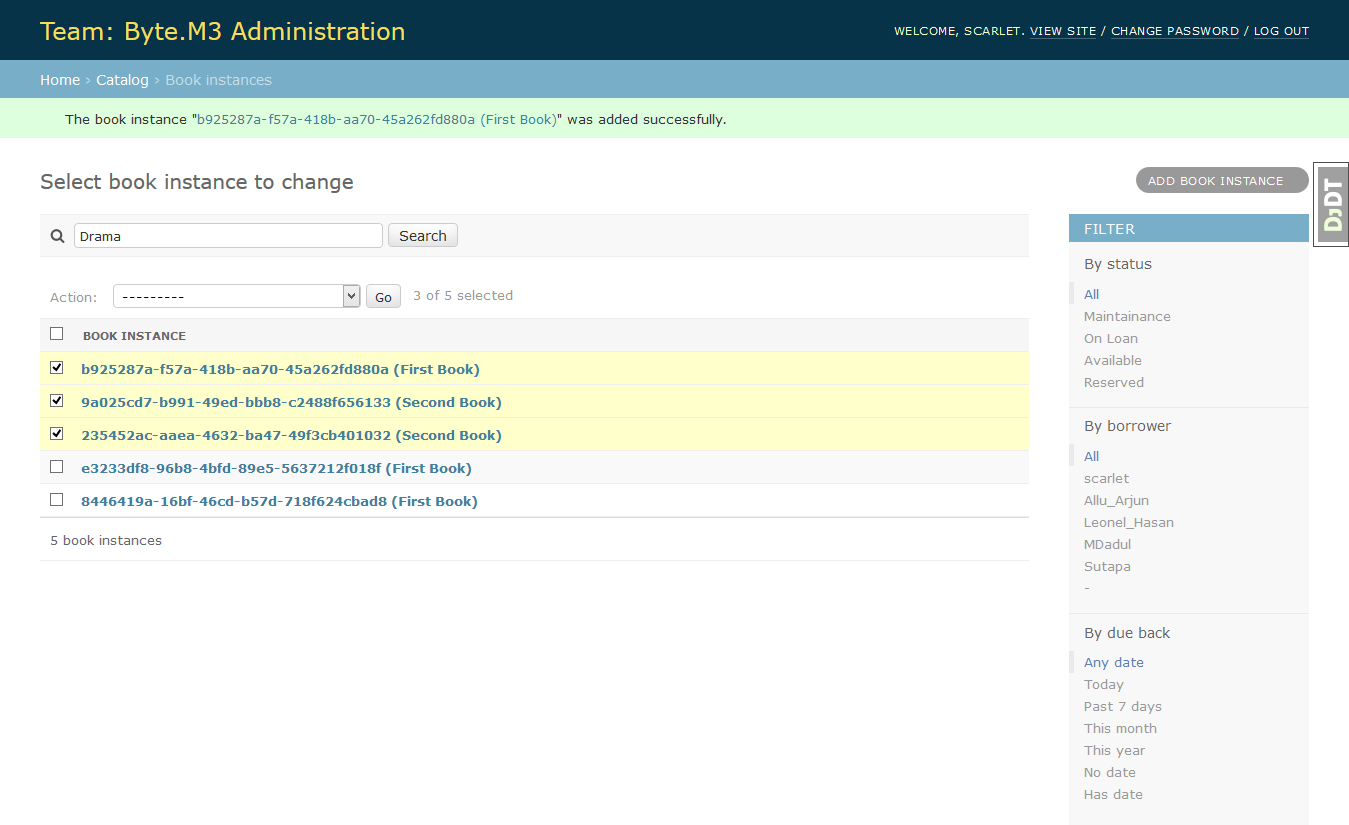


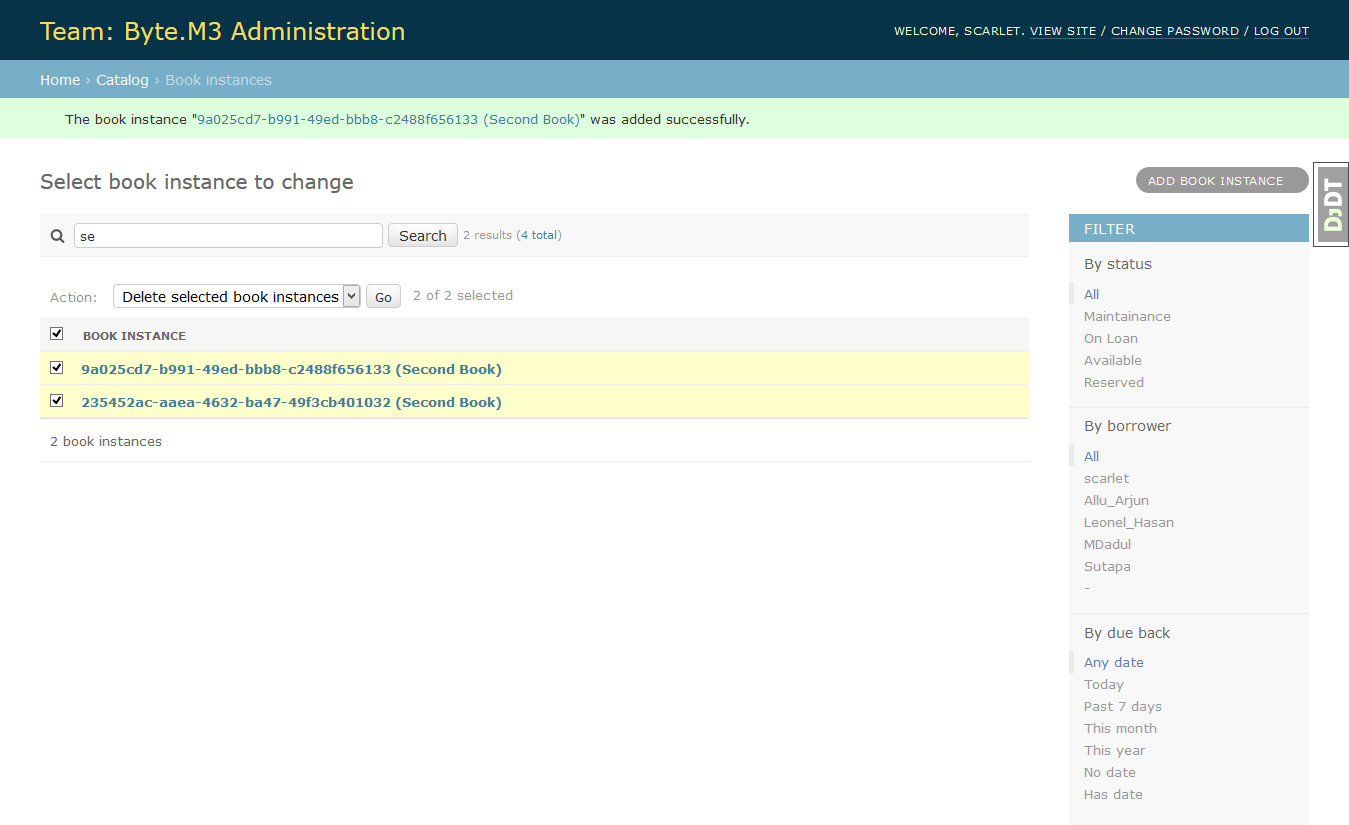


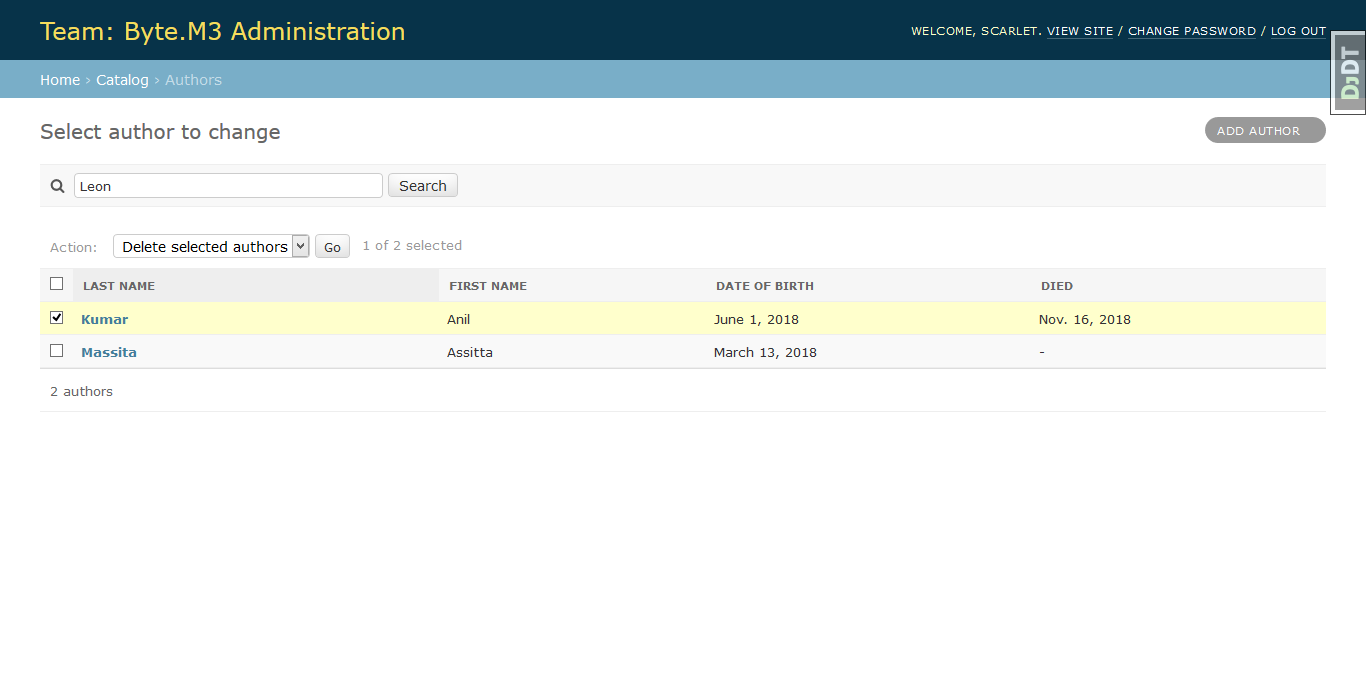


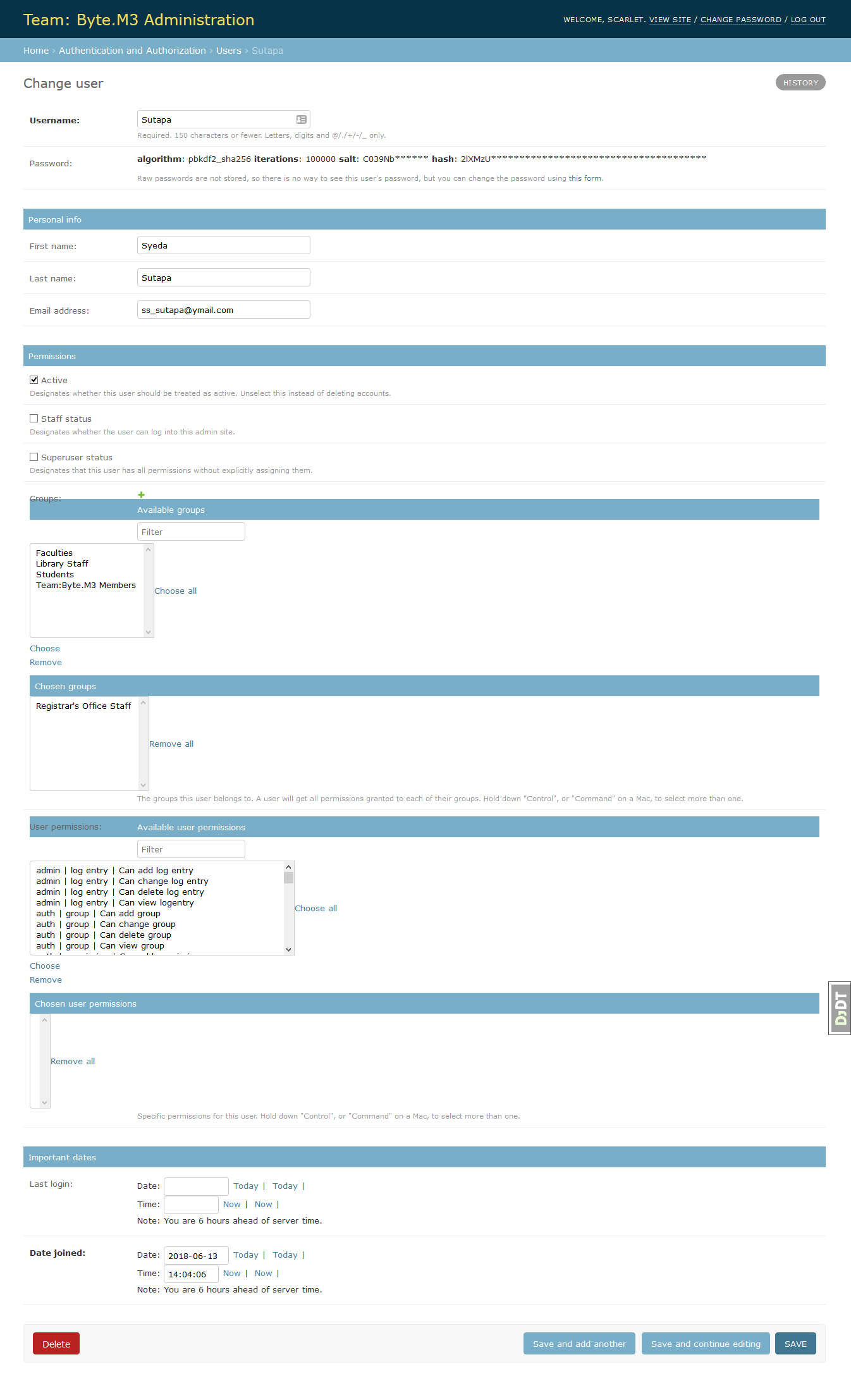


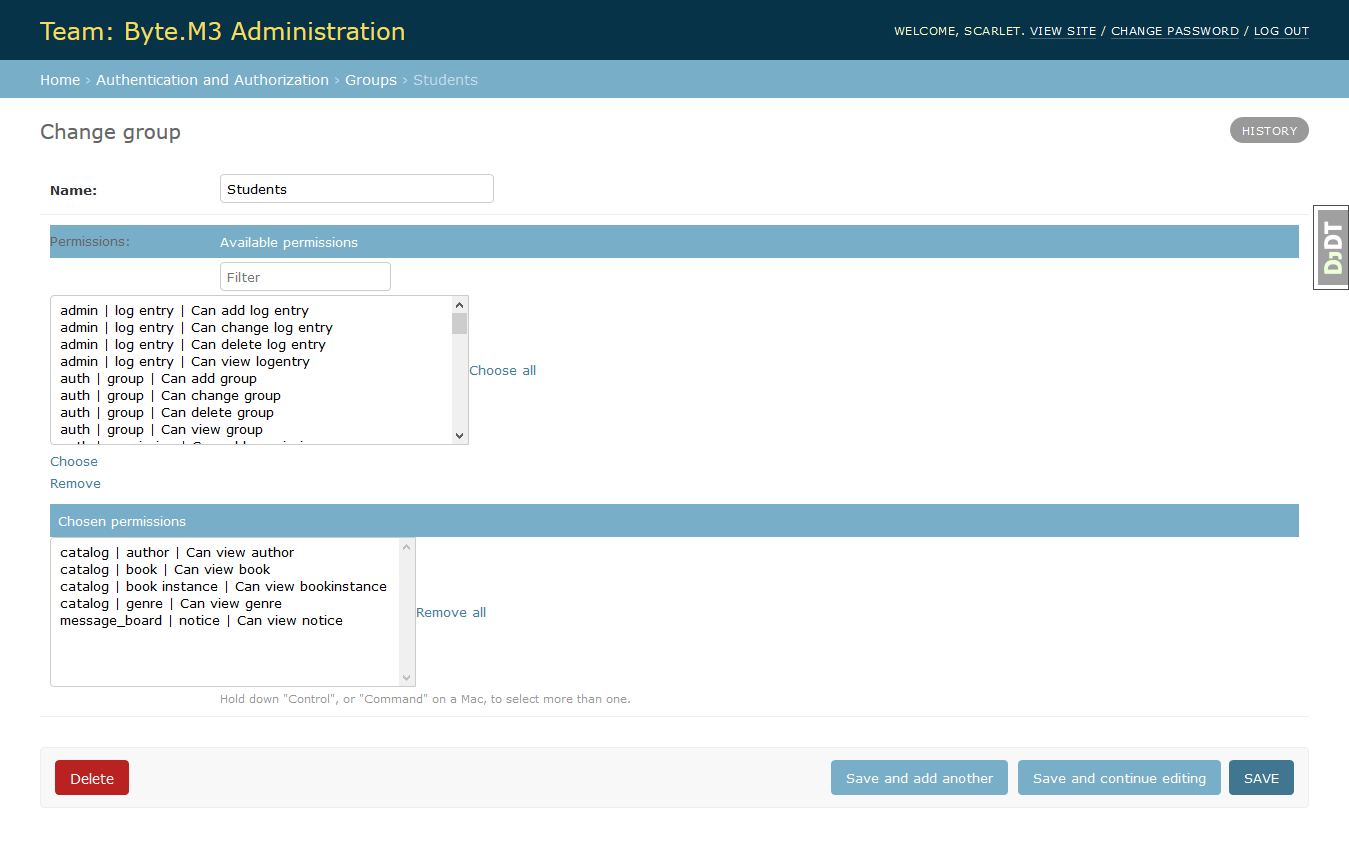


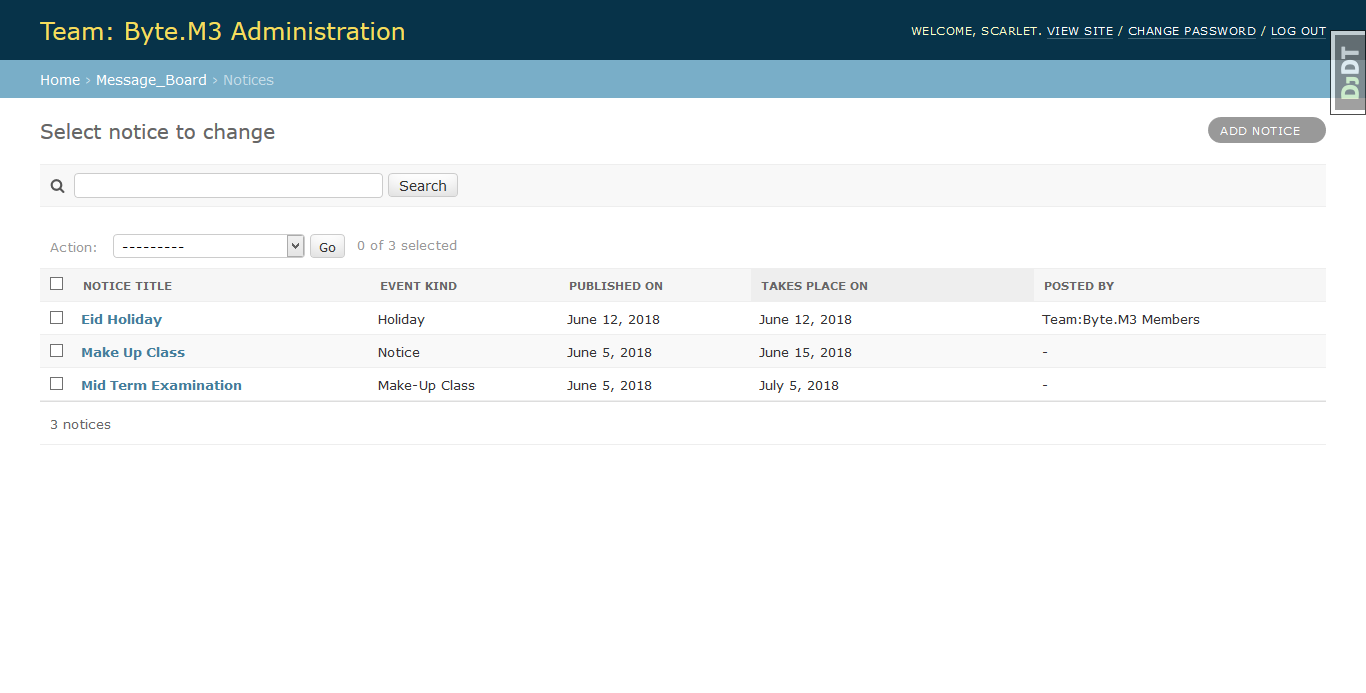


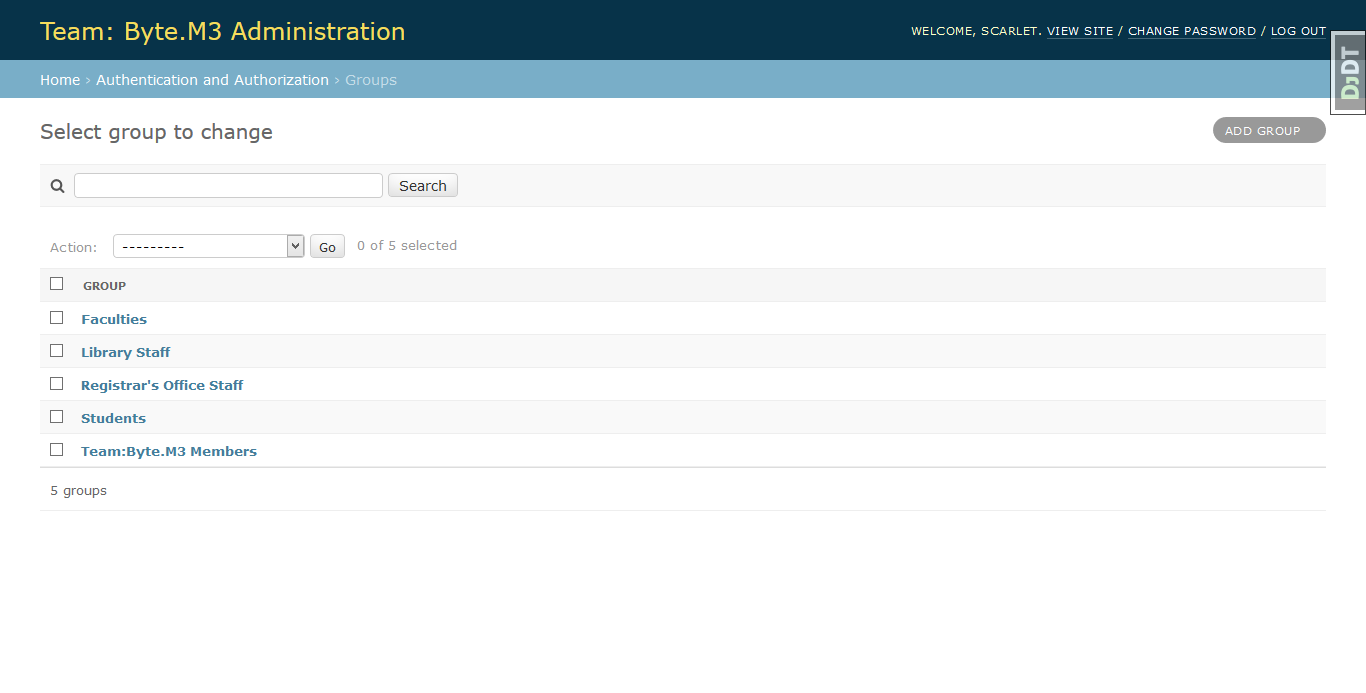


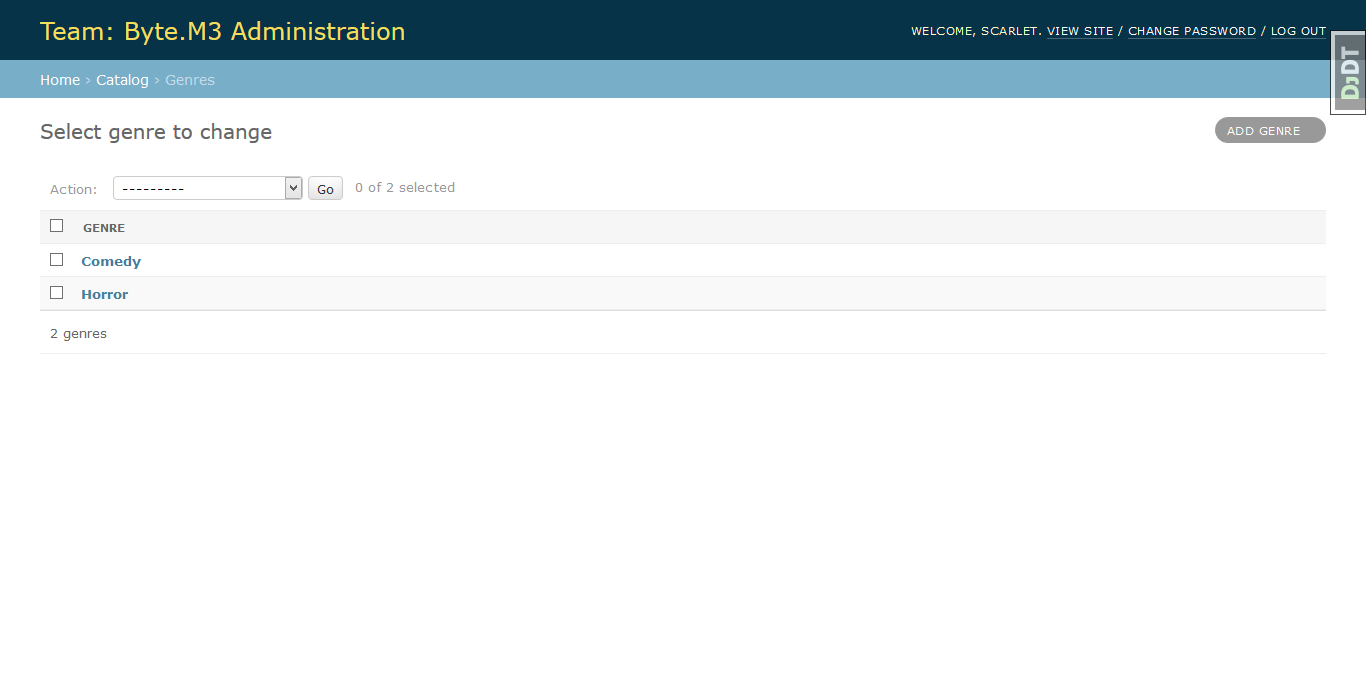


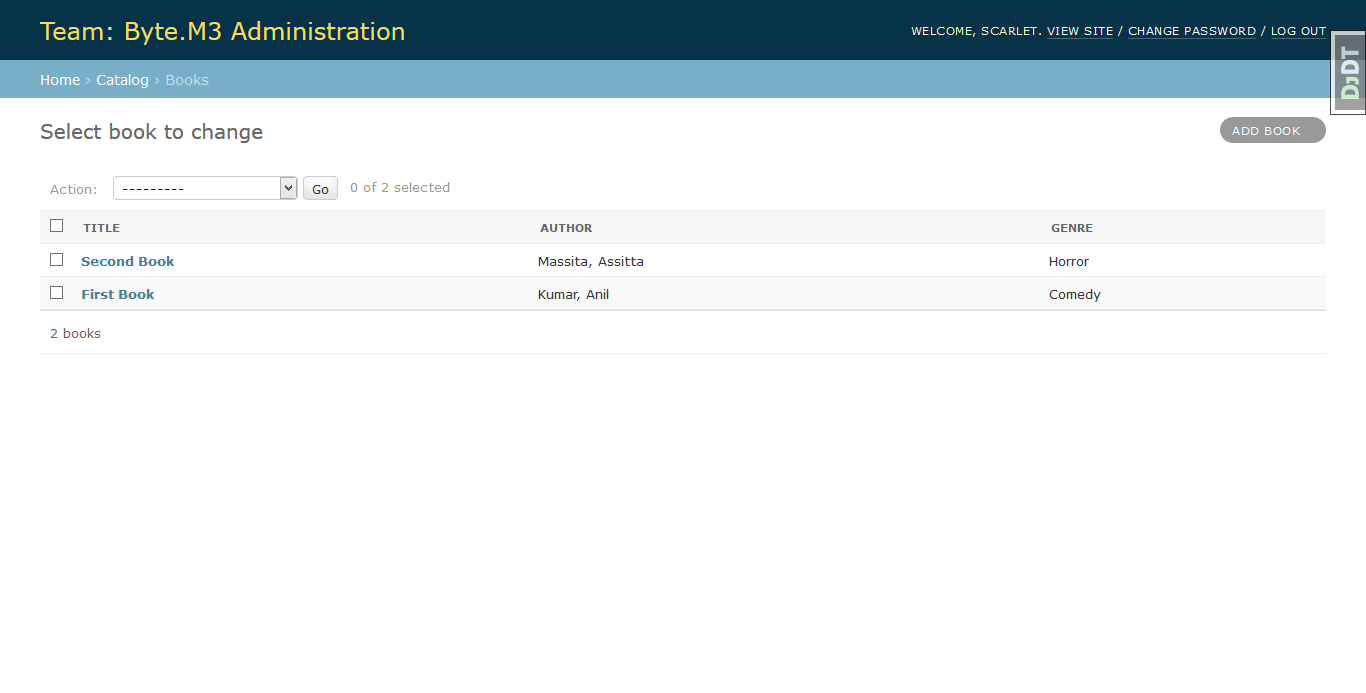


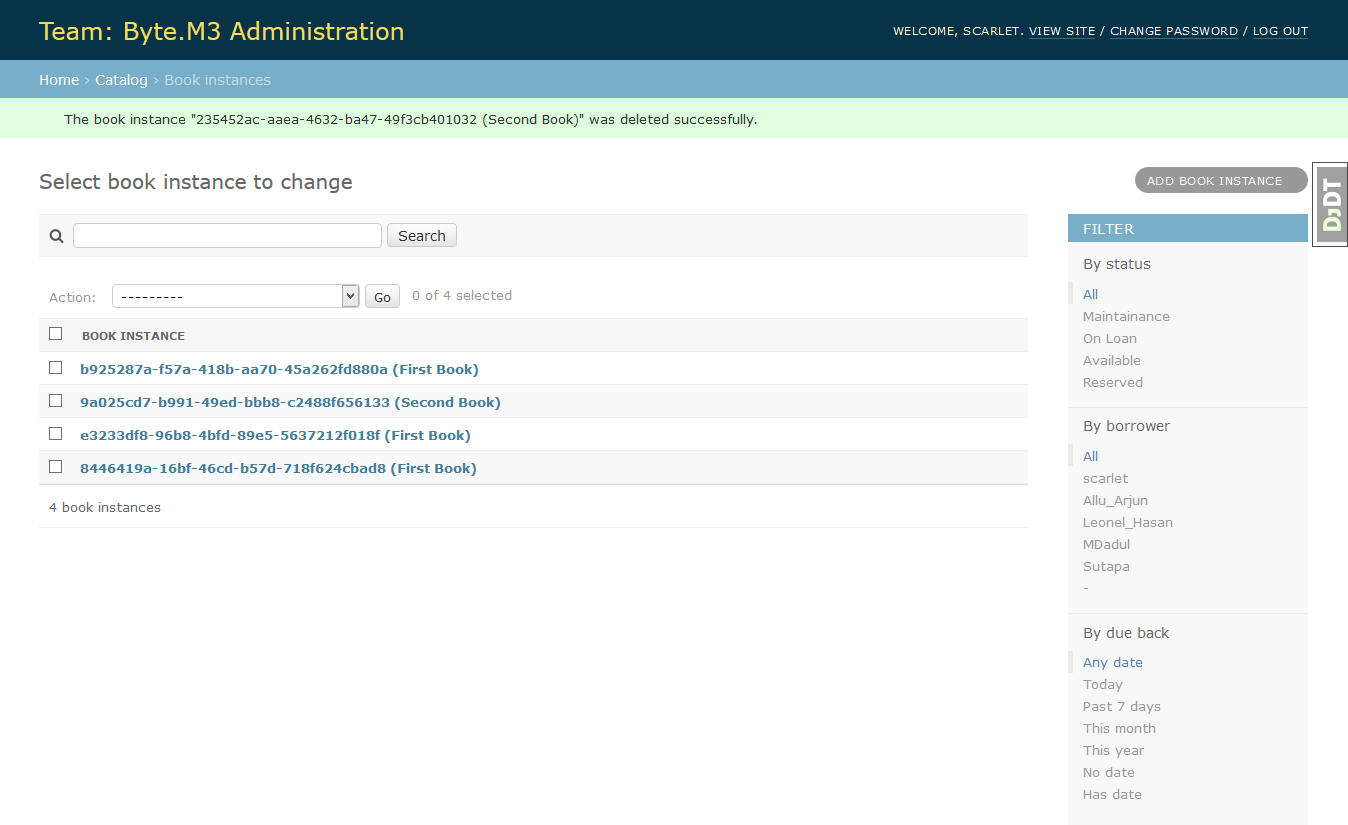


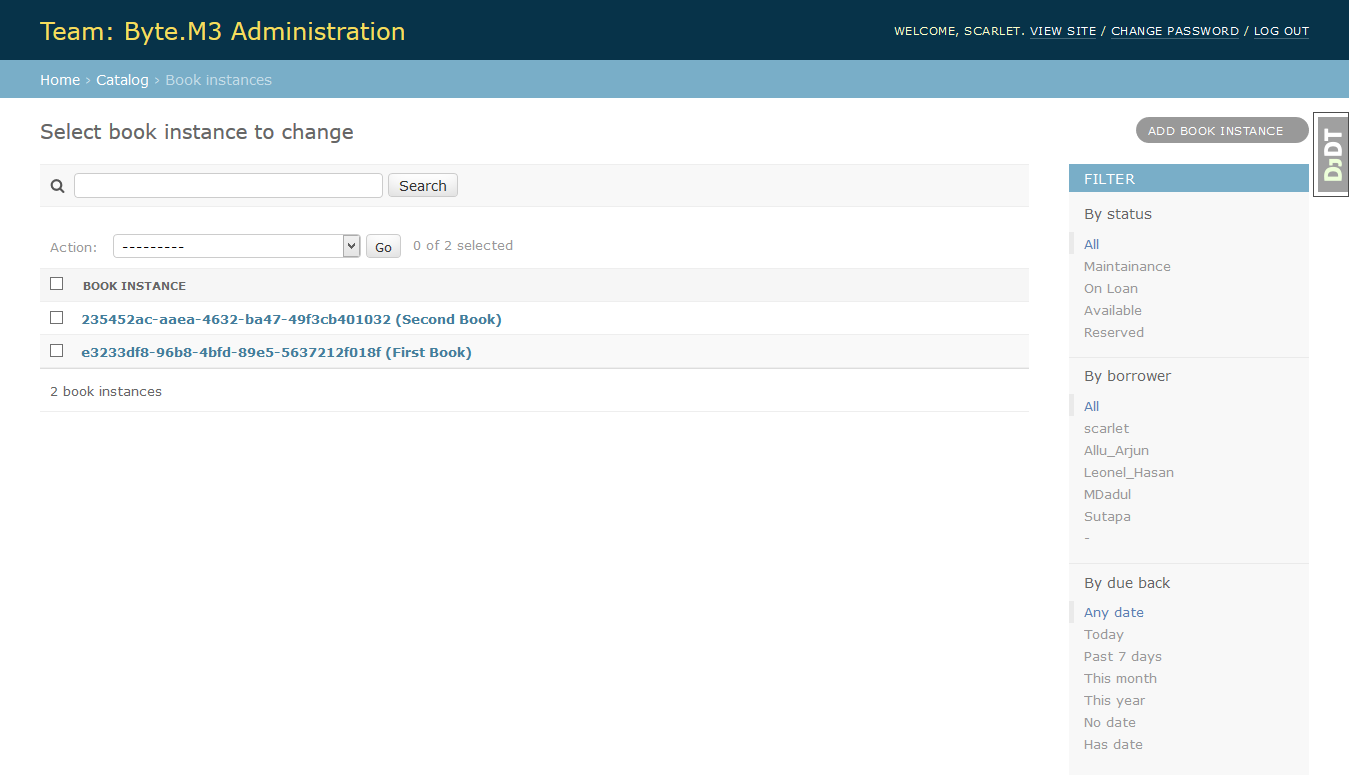


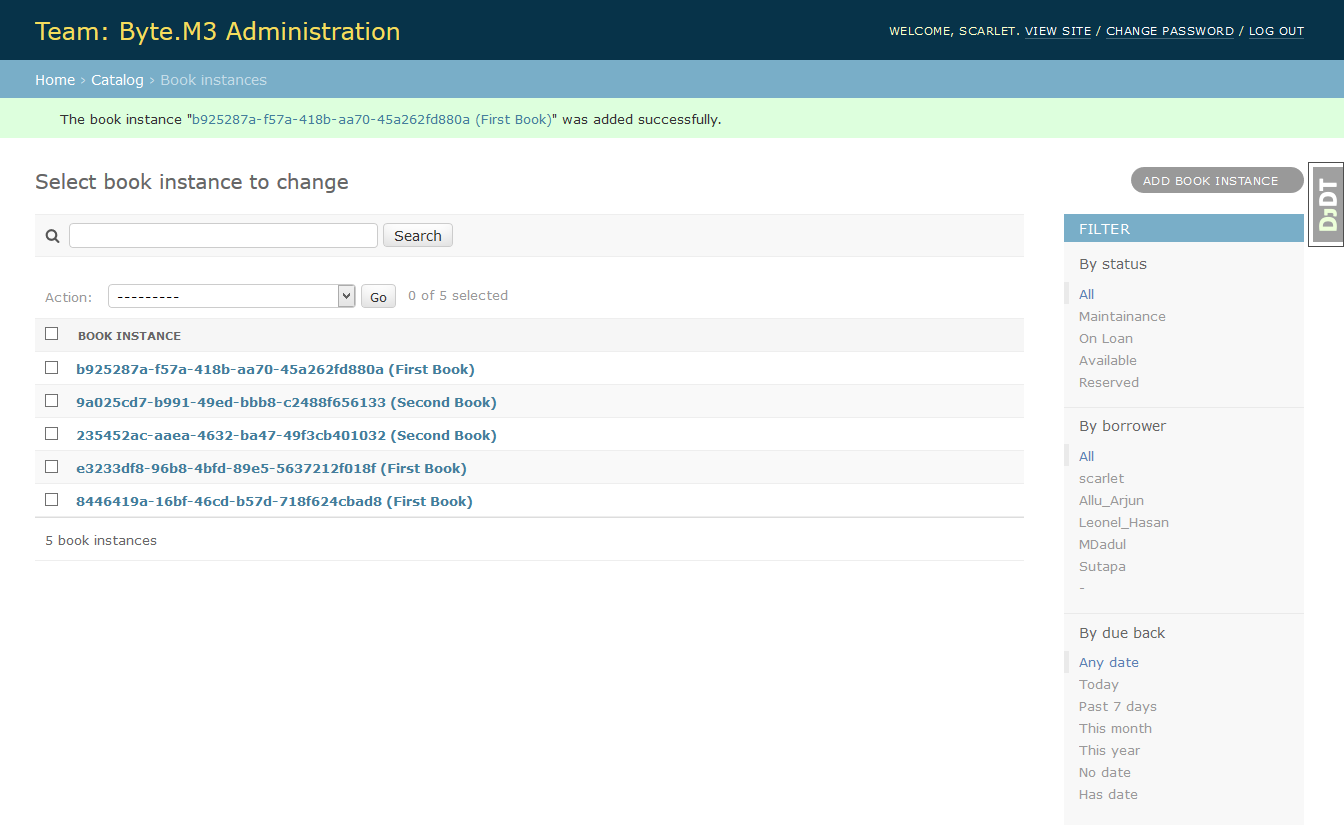


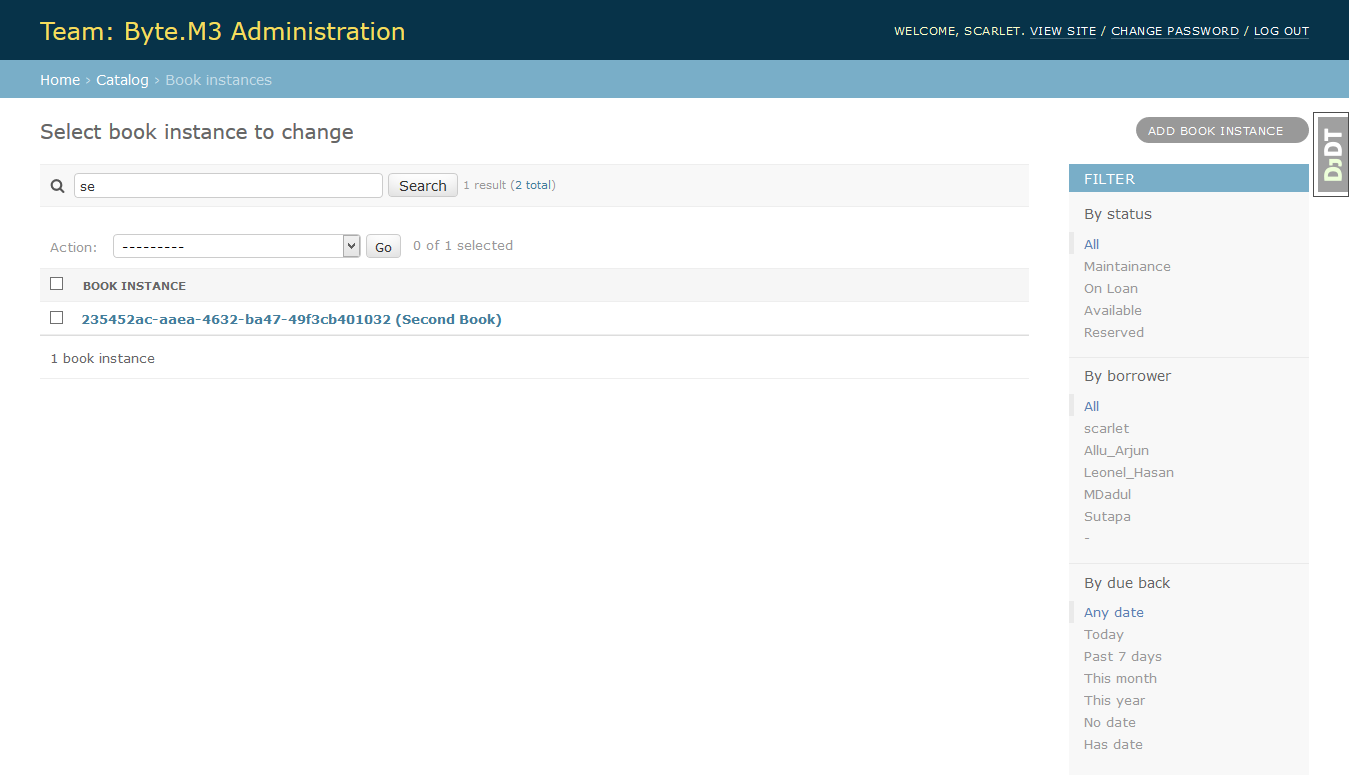


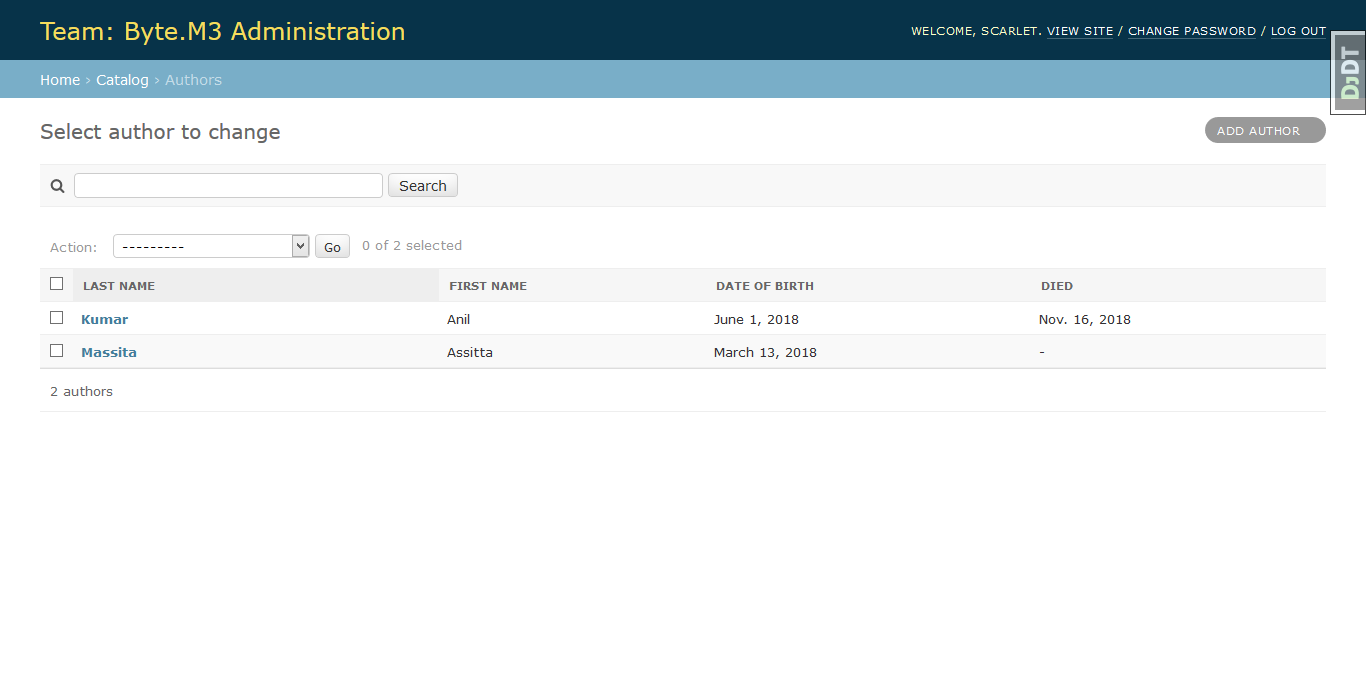


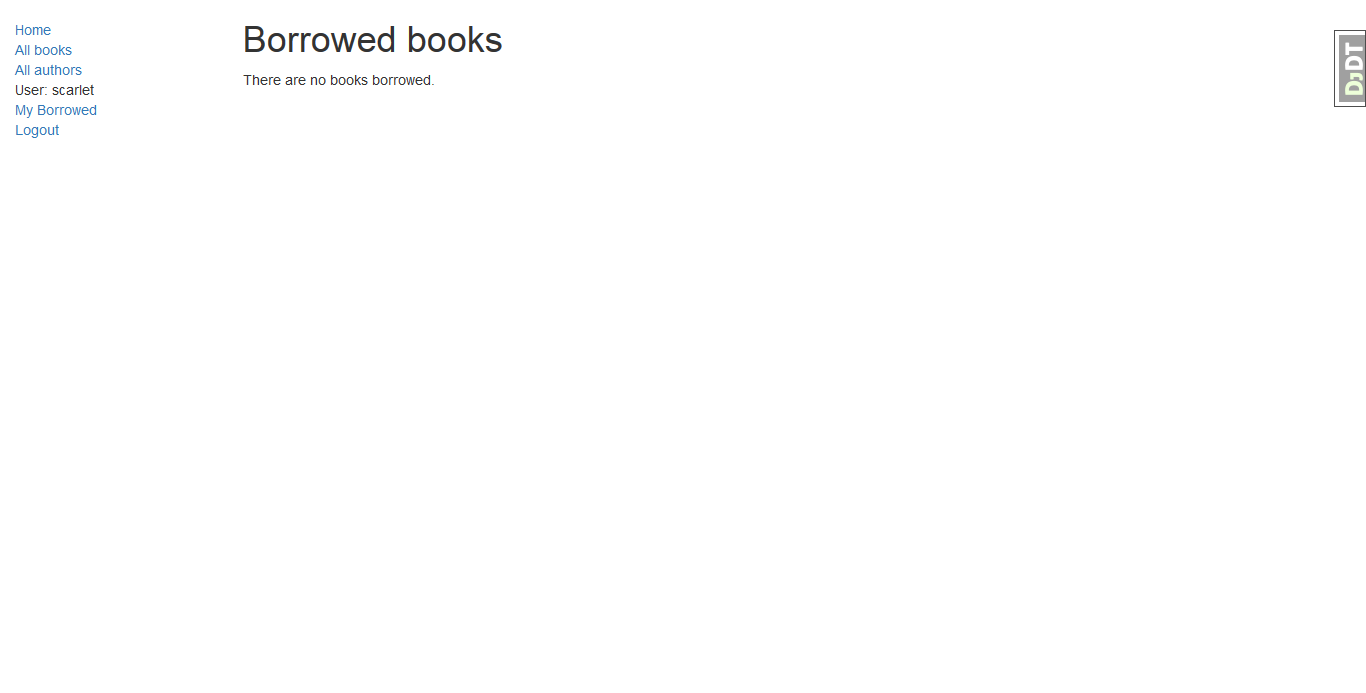


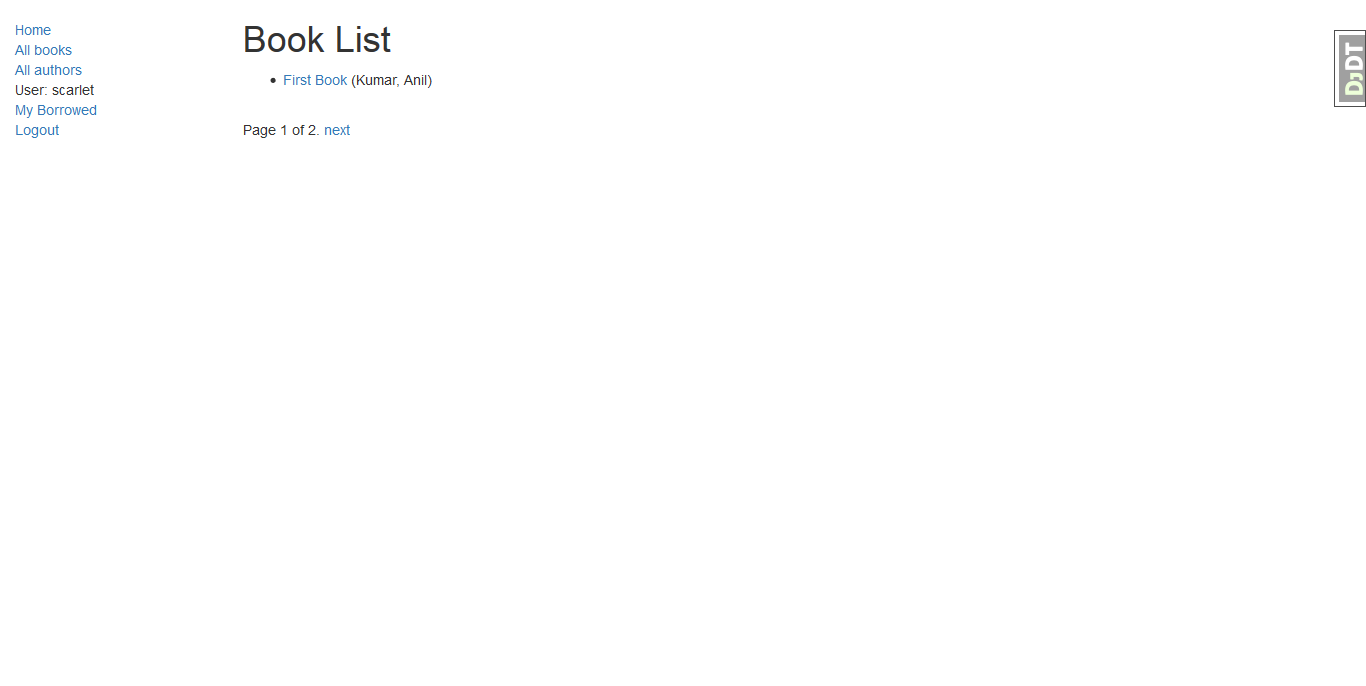


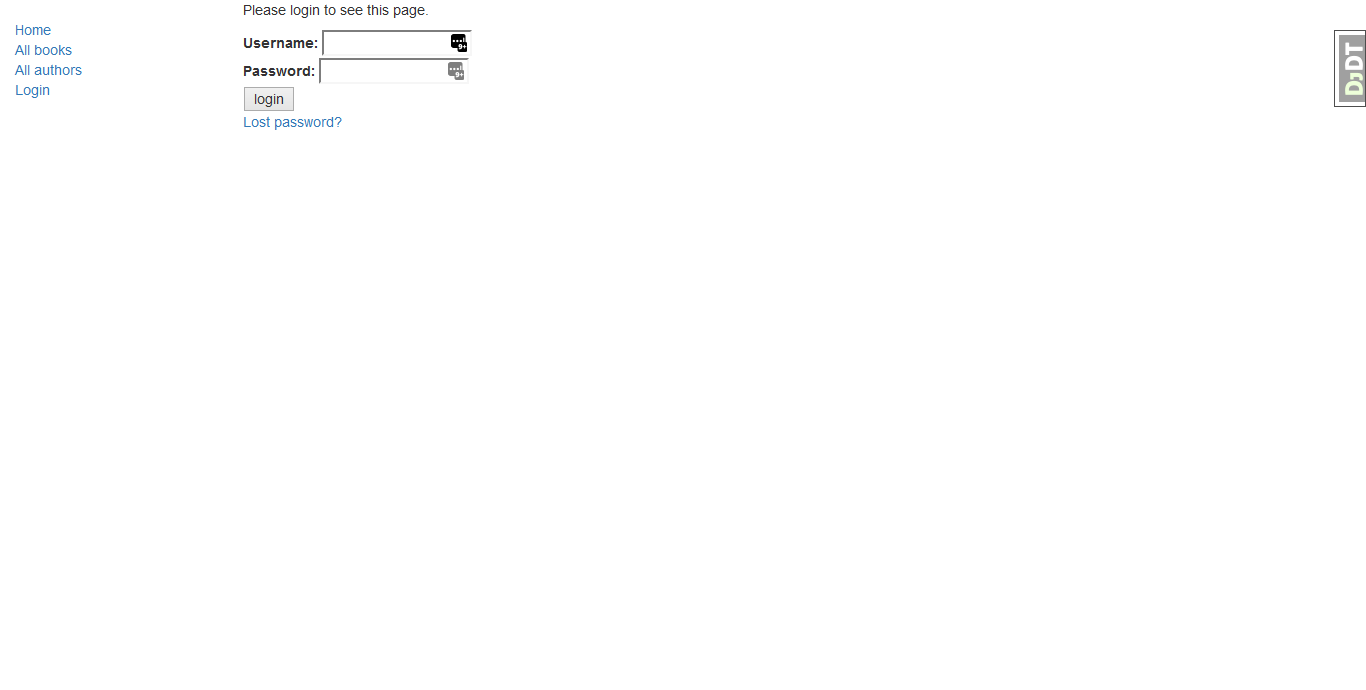


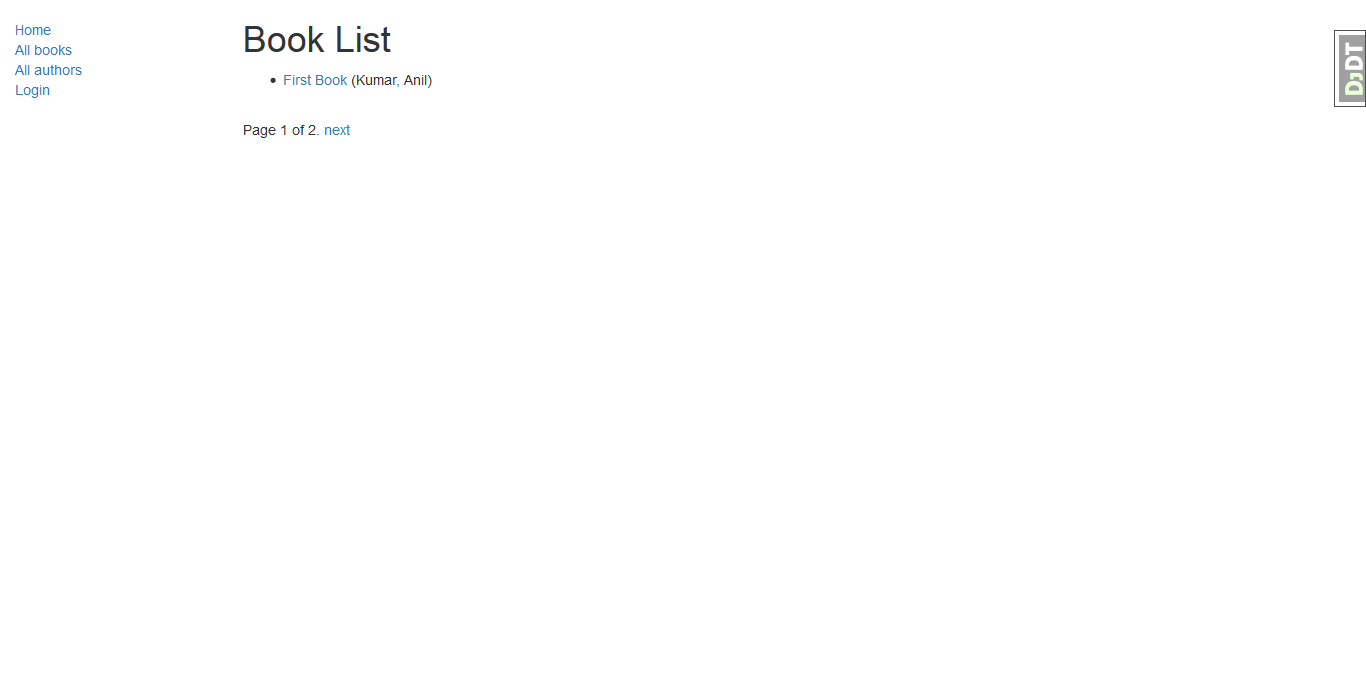


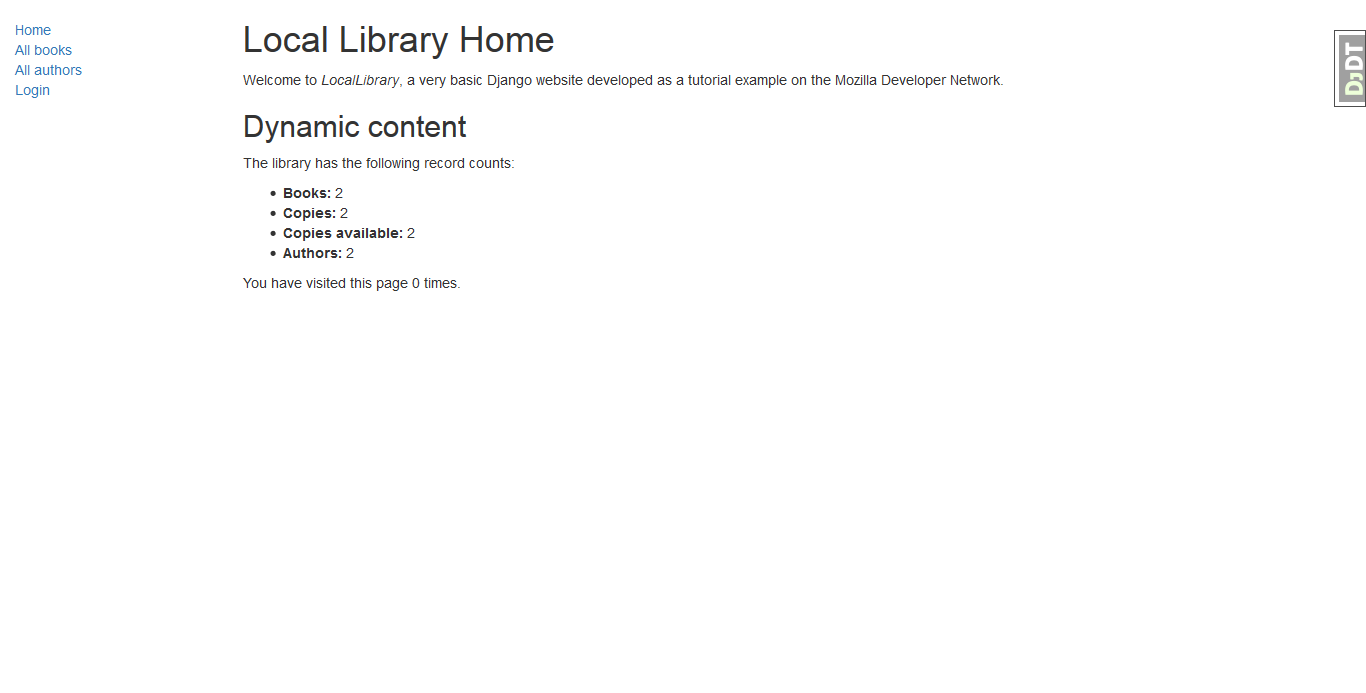


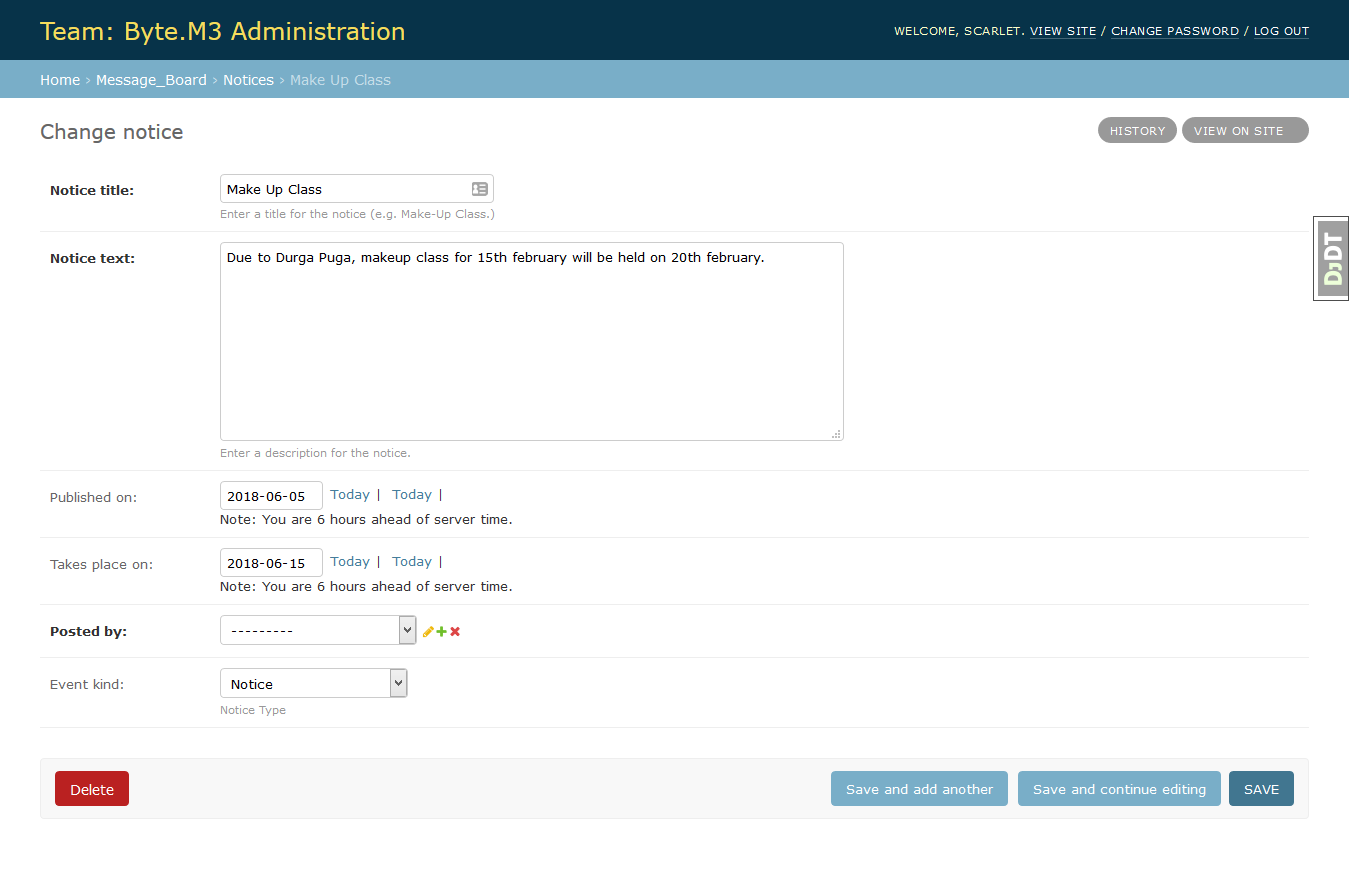


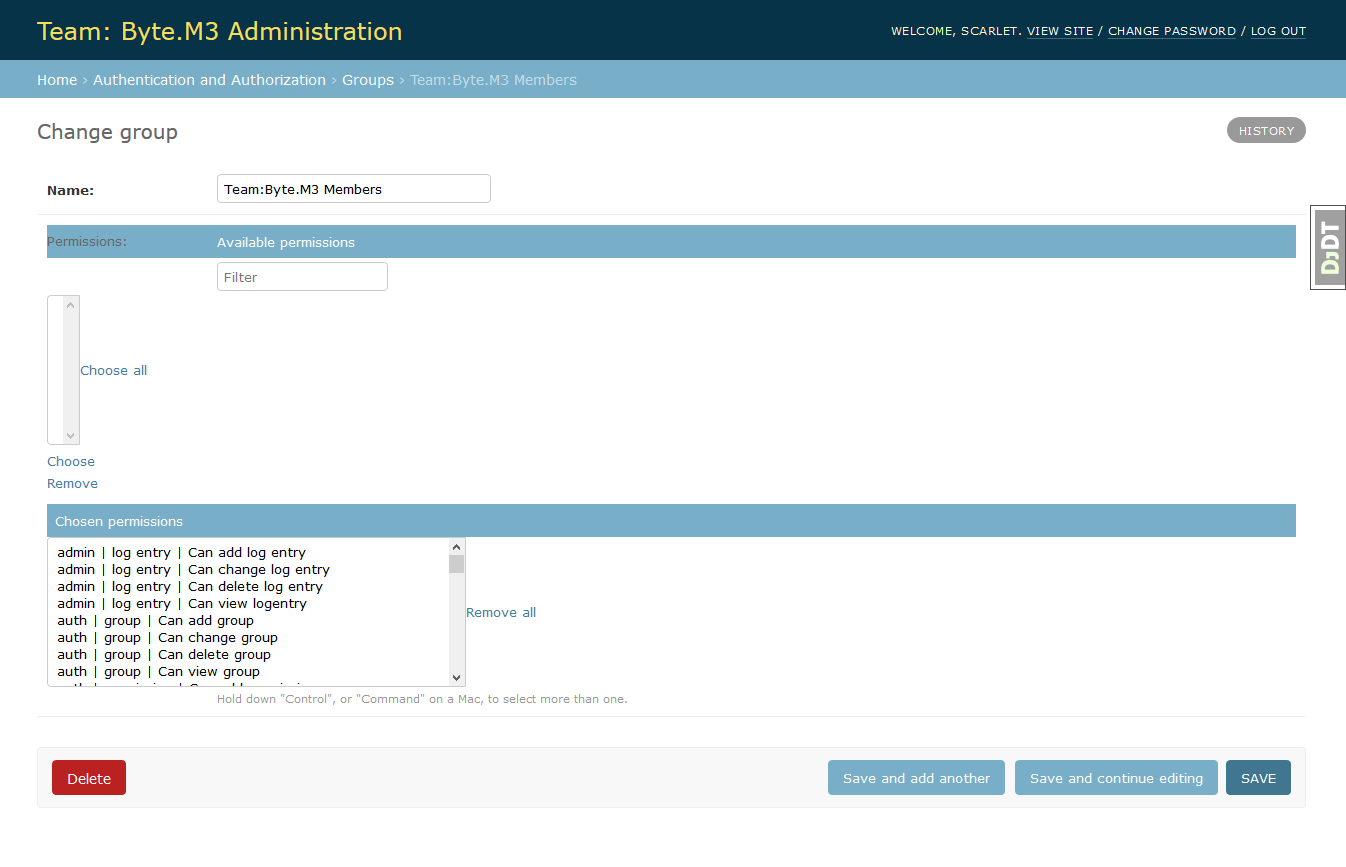


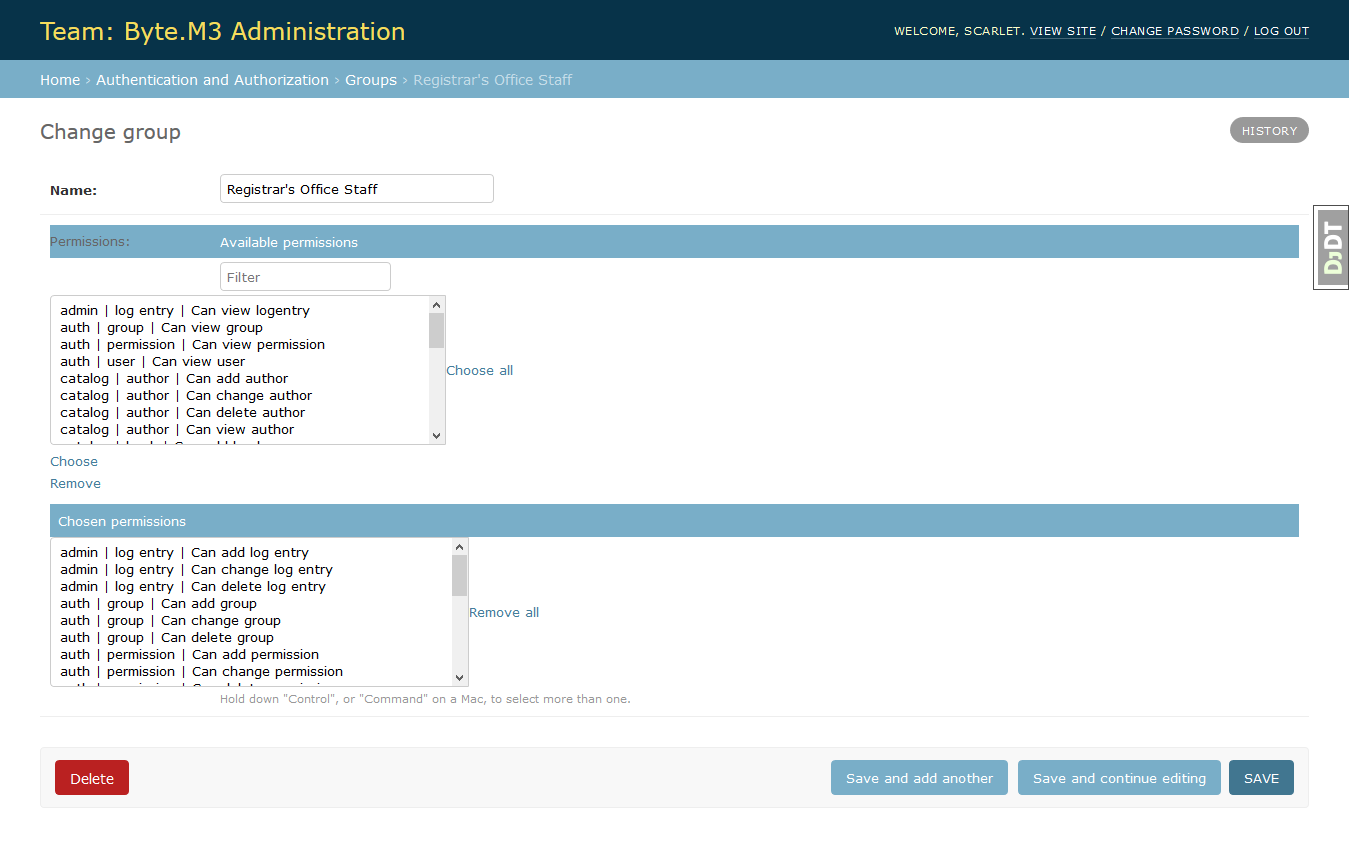




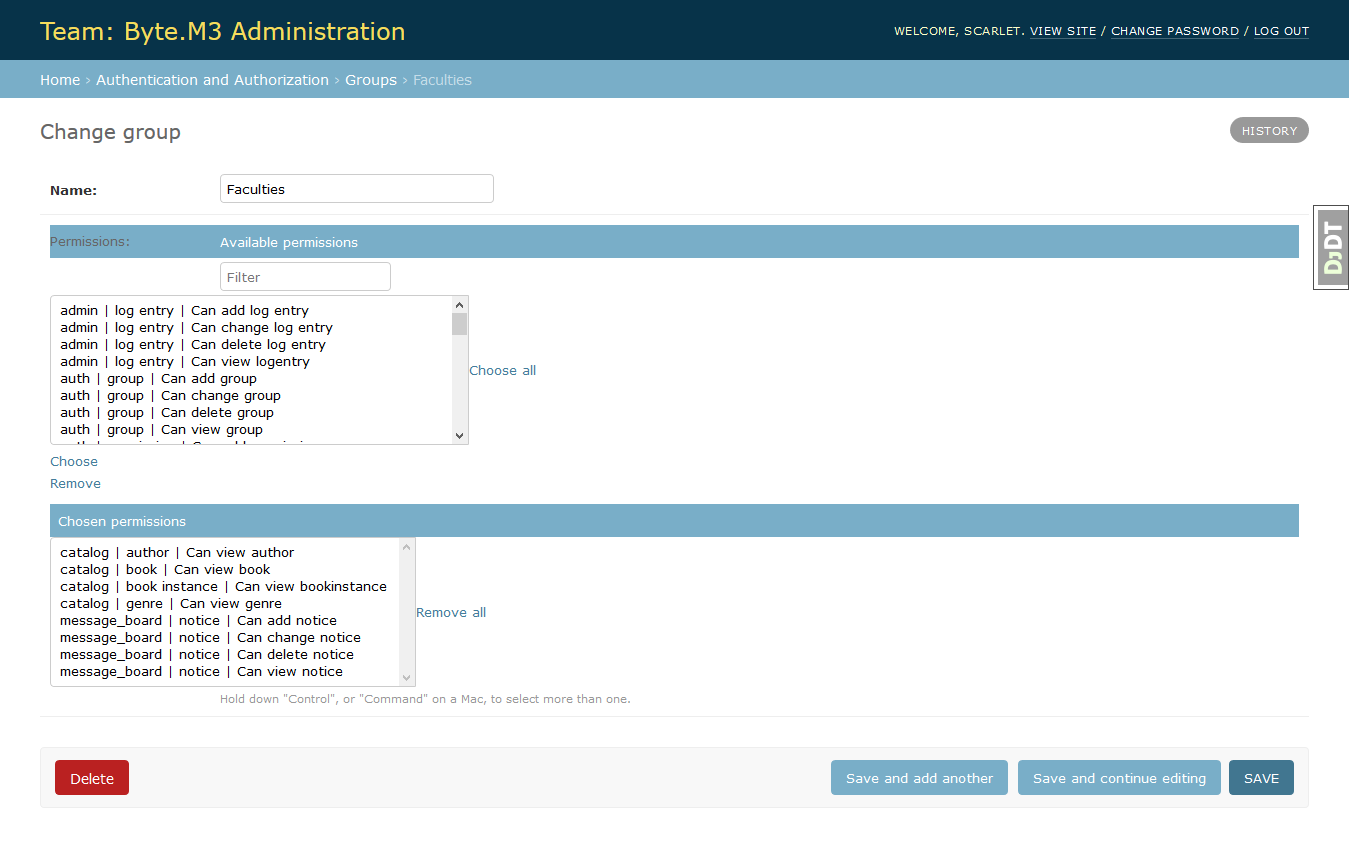


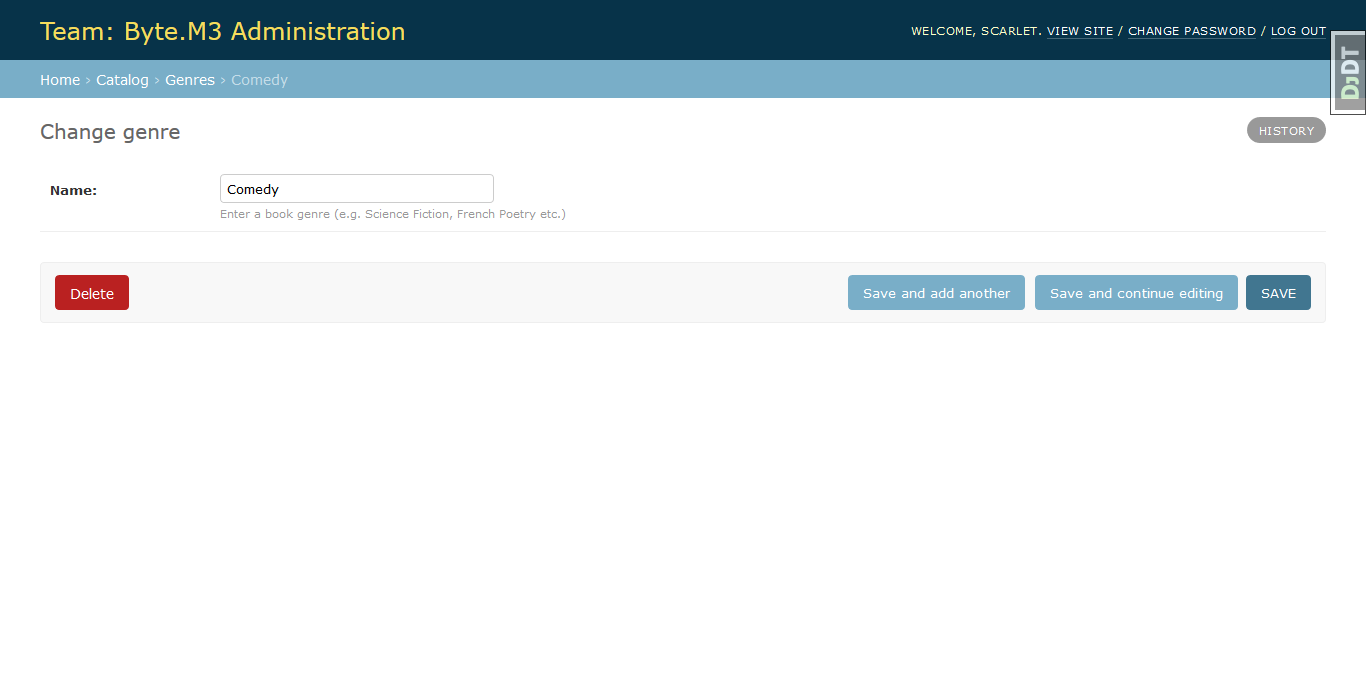


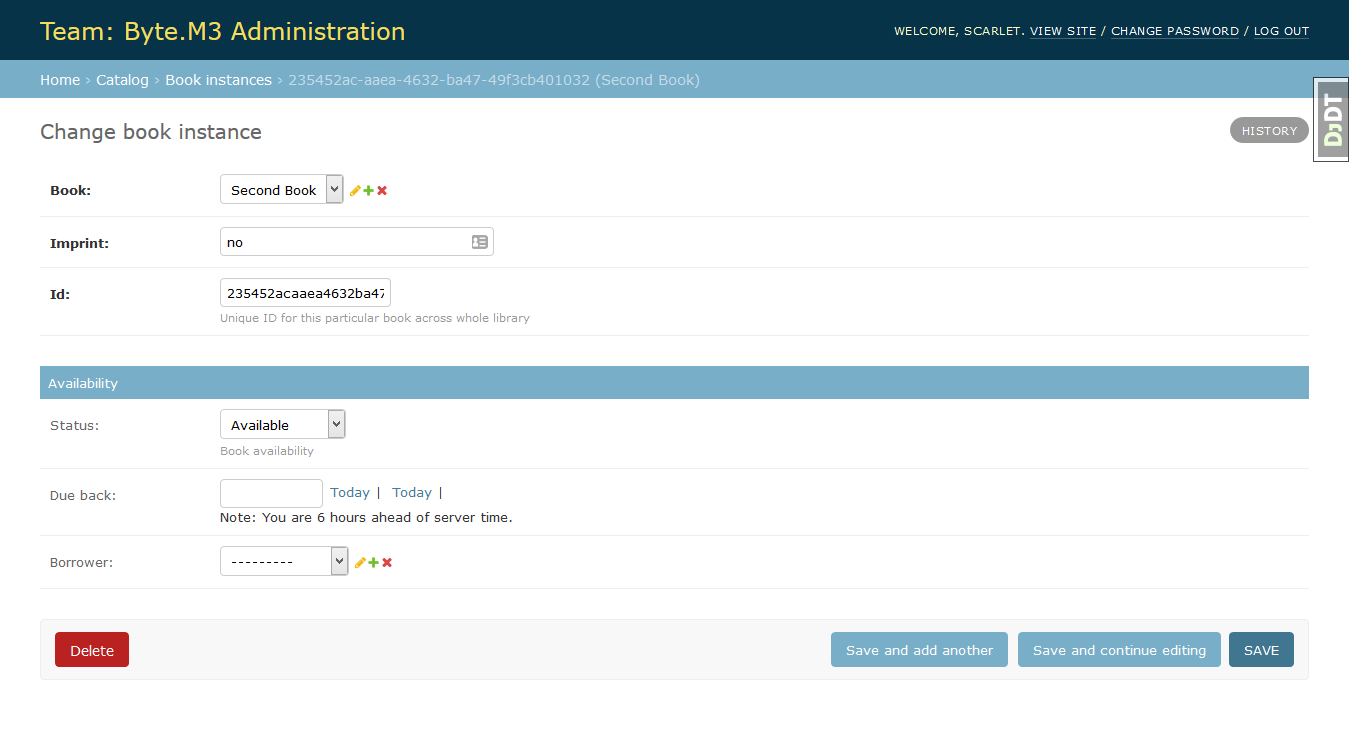


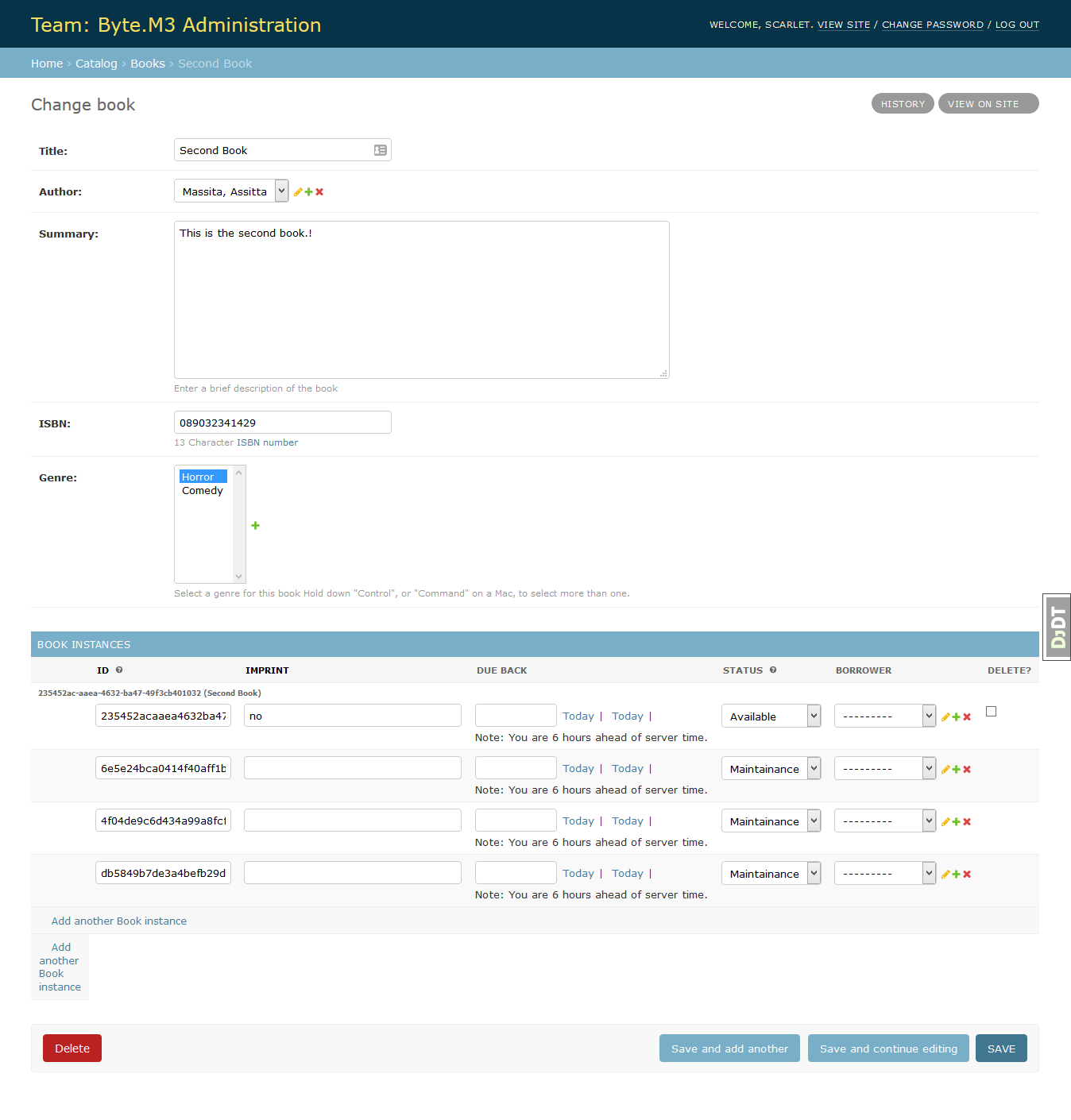


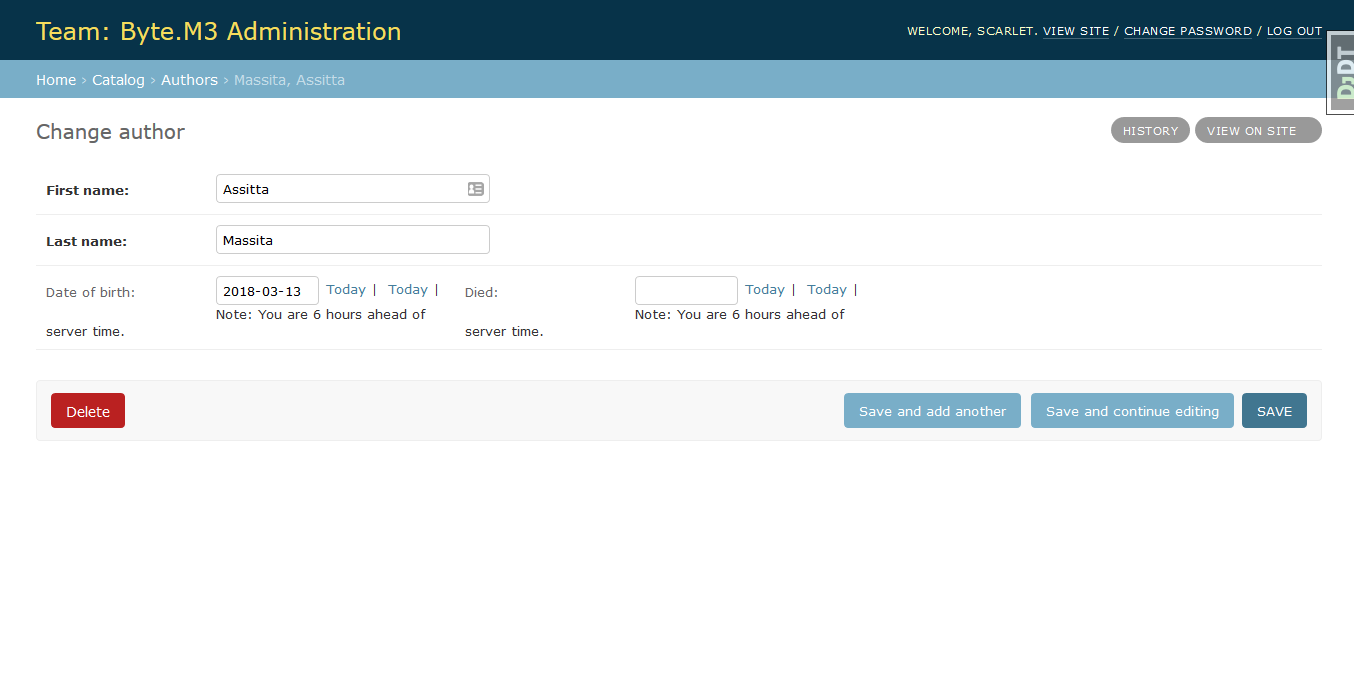


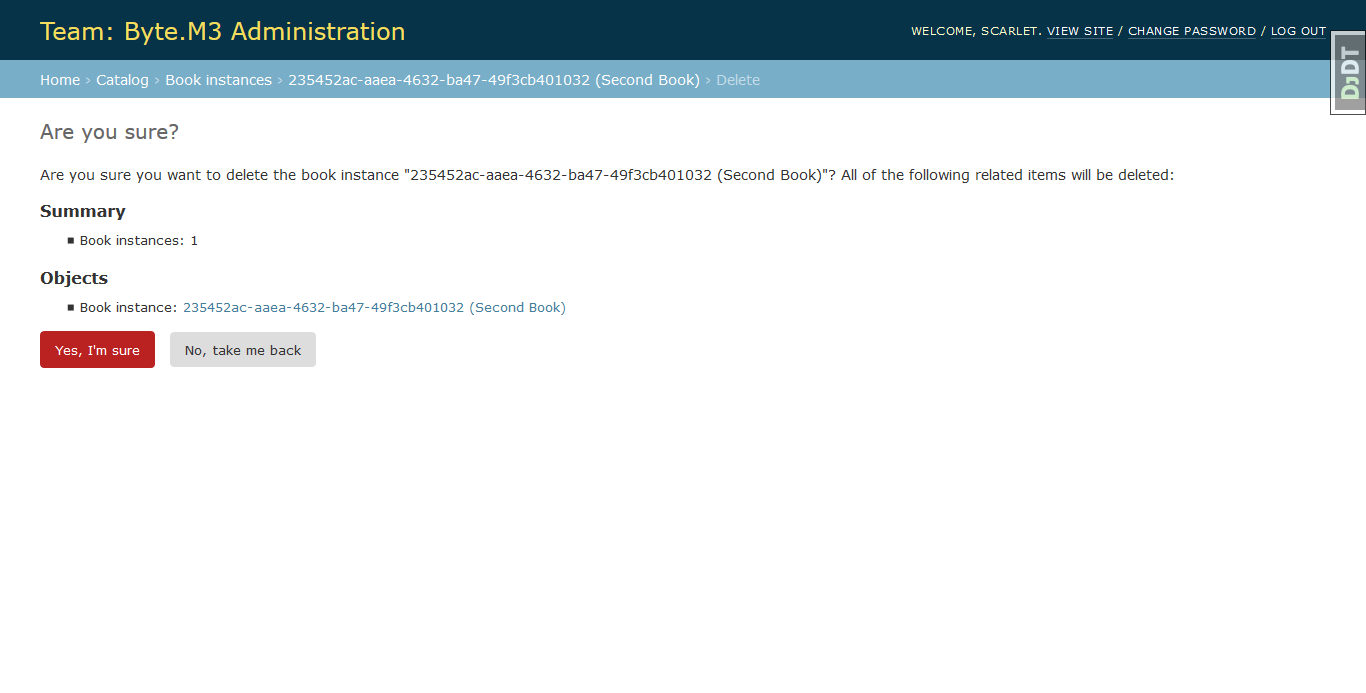


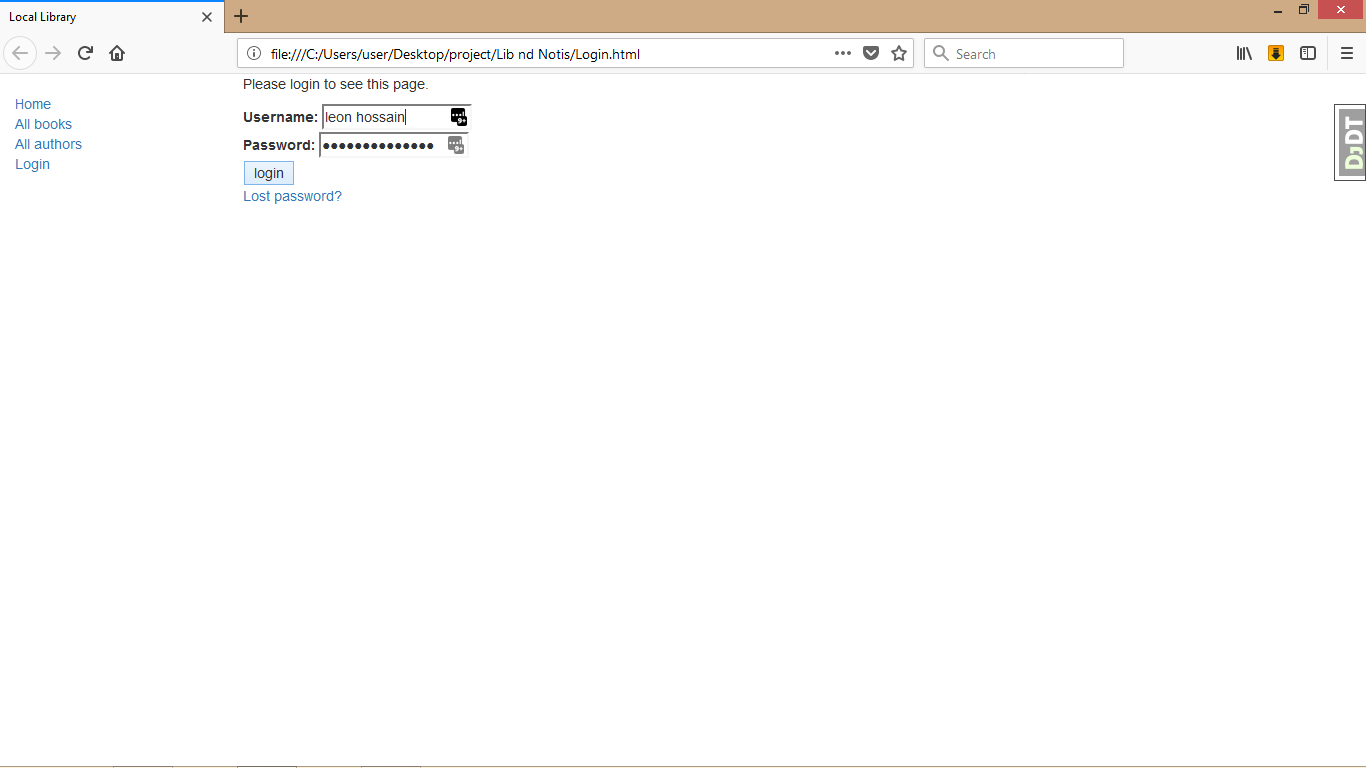


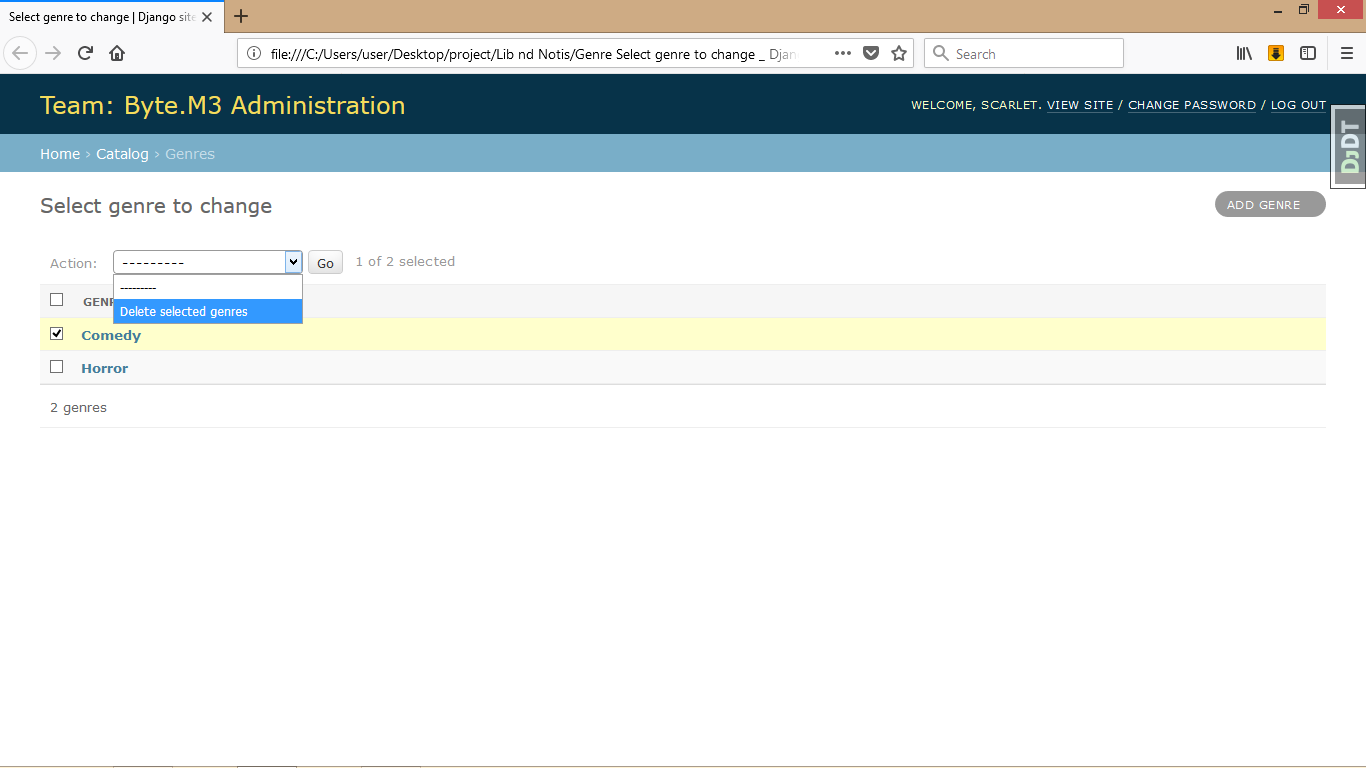


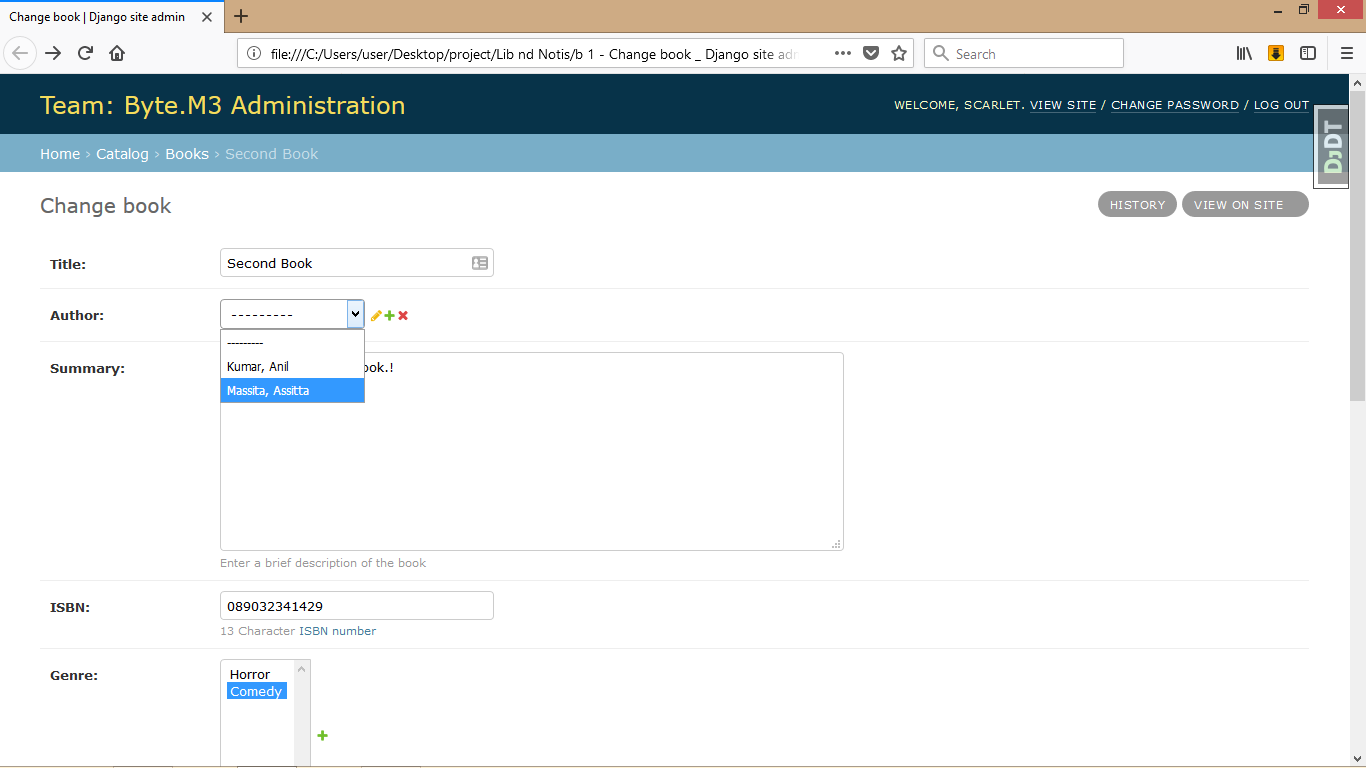


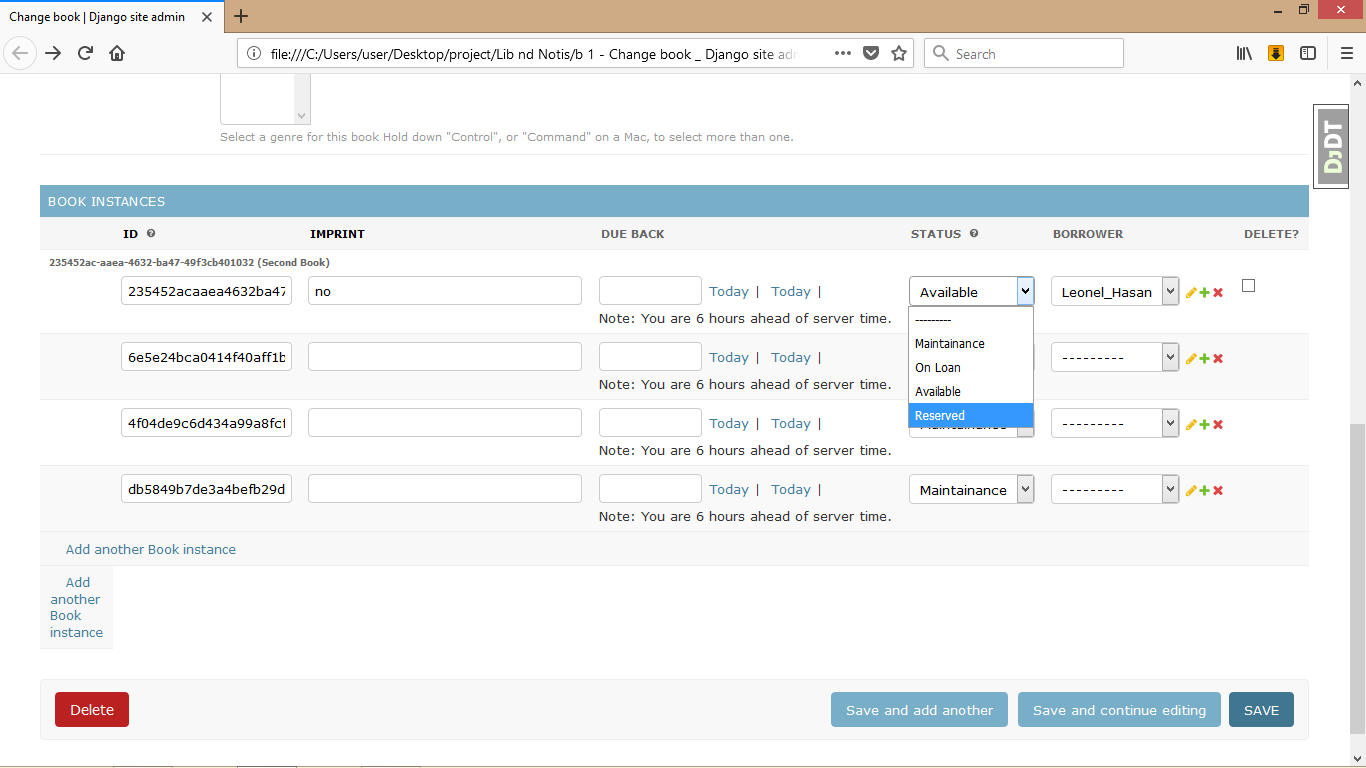


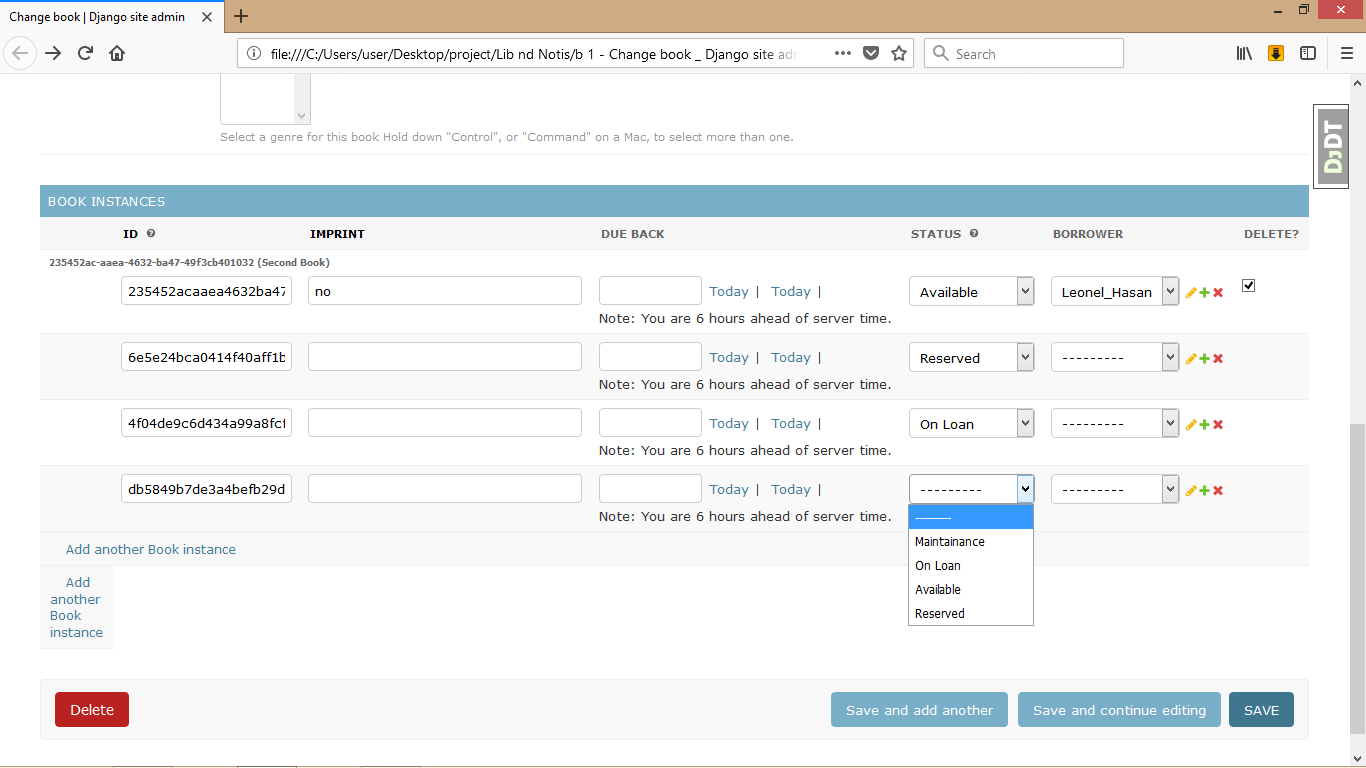


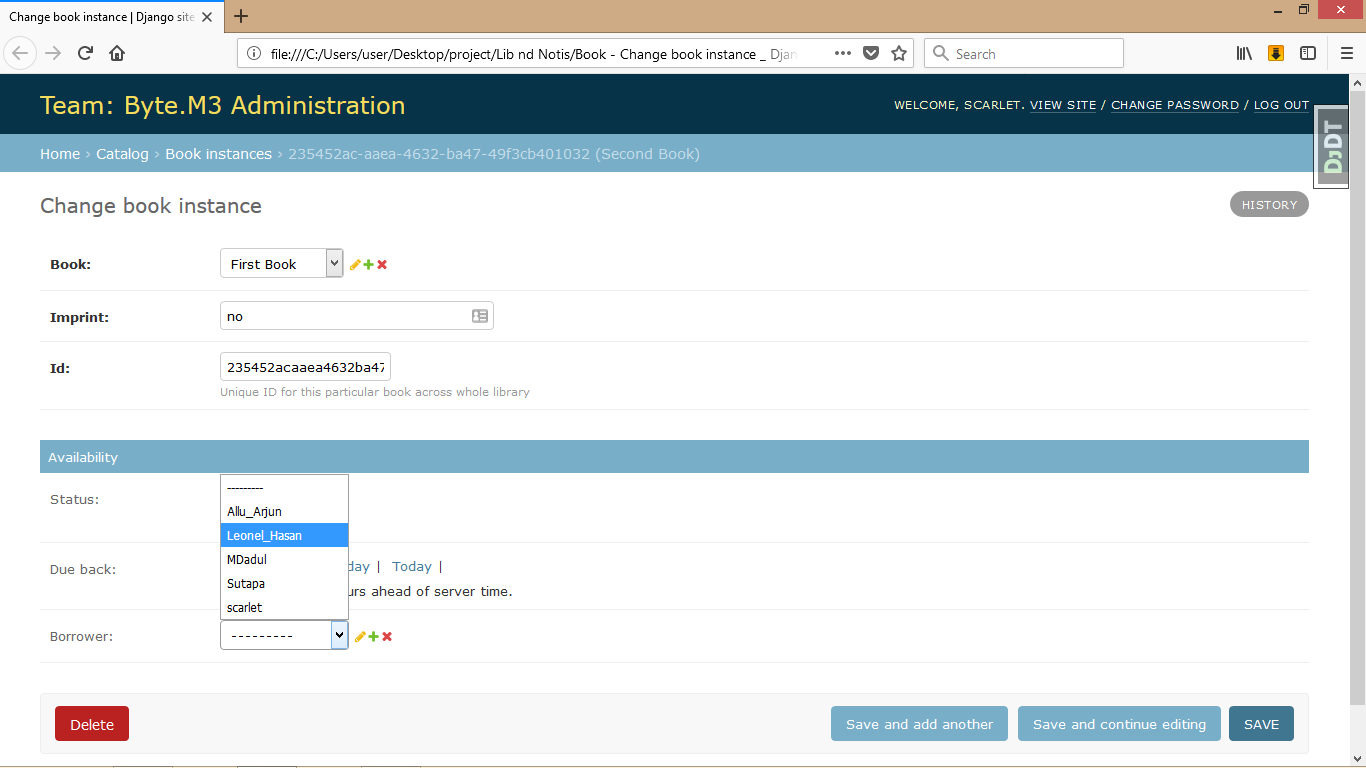


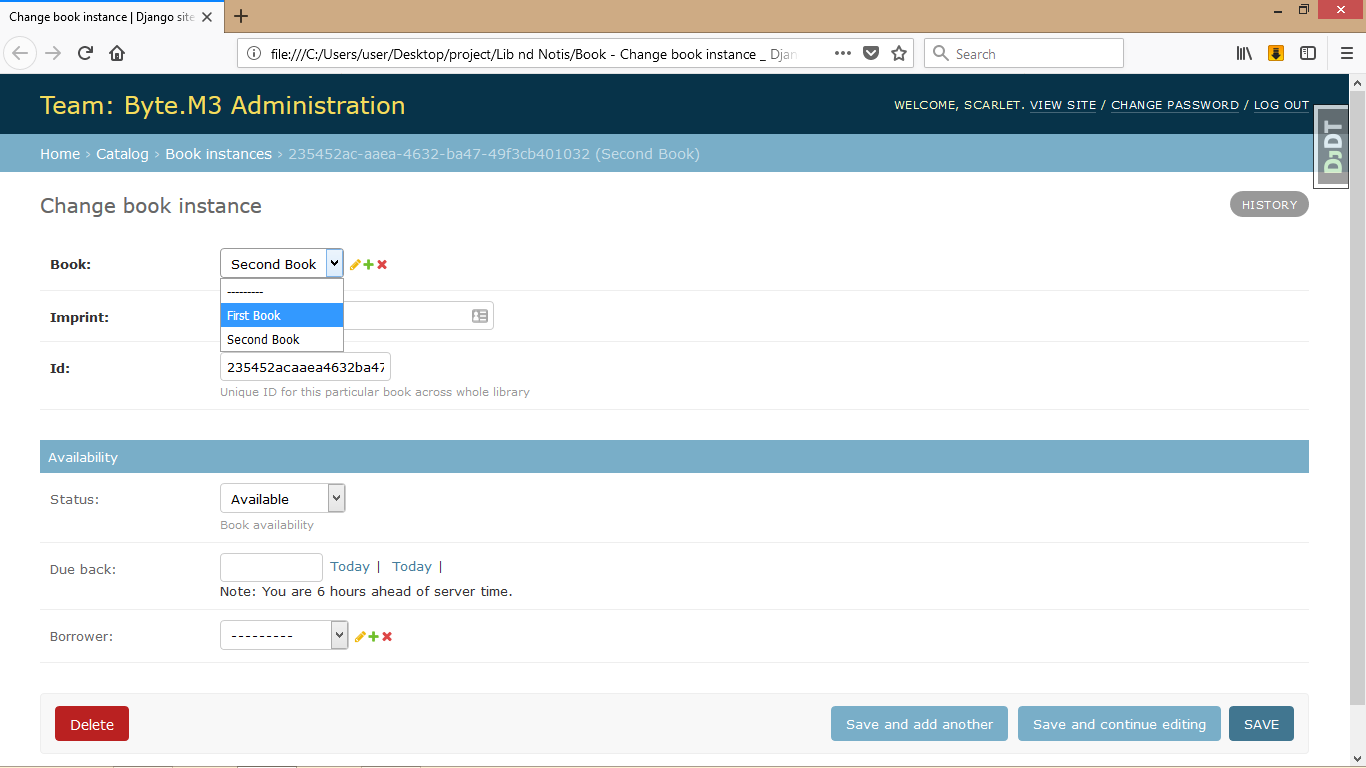


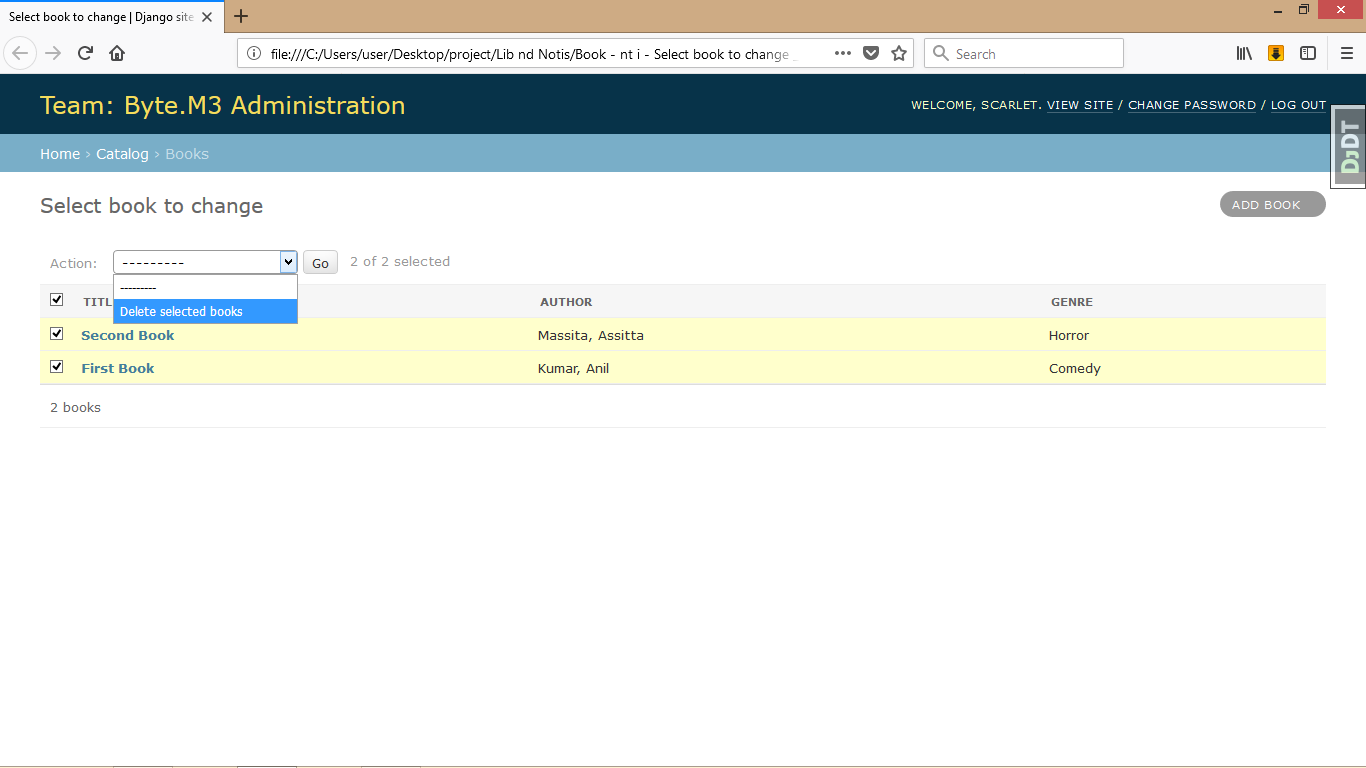










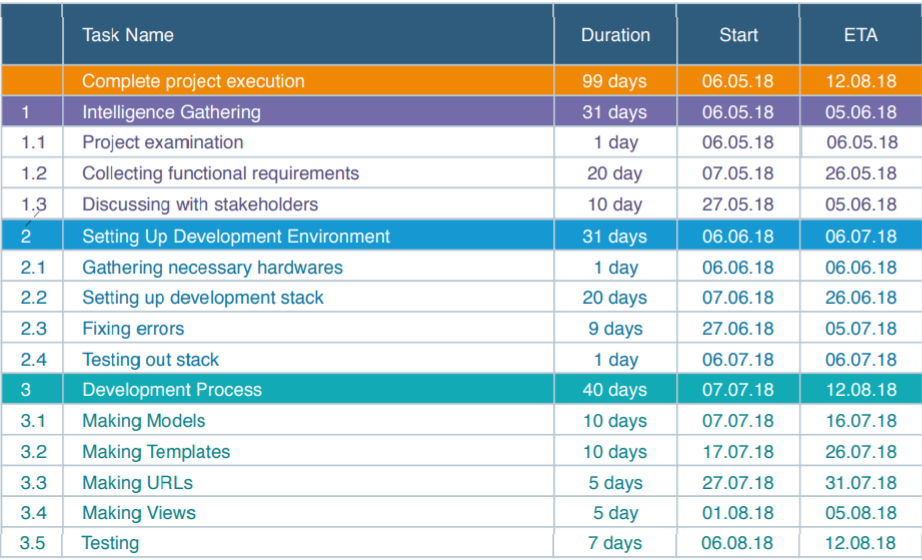


# 9 Project Management

## 9.1 Project Schedule

The Gantt chart shows the basic time allocation we had in mind for our development.

It had to be catered and optimized and altered to ensure completion.



## 9.2Social, Legal, Ethical and Professional Considerations

**Code of Conduct:**

Our project will is open sourced and commited on Github, and free for anyone to fork, use, and edit. But there are a few codes if conduct which we would like developers to follow when using or communicating or contributing to our projects. Much of our code of conduct intersect with the official Django Developer’s Code of Conduct.

Team:Byte.M3 Code of Conduct

*Be friendly and patient.*

Be welcoming. We strive to be a community that welcomes and supports people of all backgrounds and identities. This includes, but is not limited to members of any race, ethnicity, culture, national origin, colour, immigration status, social and economic class, educational level, sex, sexual orientation, gender identity and expression, age, size, family status, political belief, religion, and mental and physical ability.

*Be considerate.*

Your work will be used by other people, and you in turn will depend on the work of others. Any decision you take will affect users and colleagues, and you should take those consequences into account when making decisions. Remember that we're a world-wide community, so you might not be communicating in someone else's primary language.

*Be respectful.*

Not all of us will agree all the time, but disagreement is no excuse for poor behavior and poor manners. We might all experience some frustration now and then, but we cannot allow that frustration to turn into a personal attack. It’s important to remember that a community where people feel uncomfortable or threatened is not a productive one. Members of the Team:Byte.M3 community should be respectful when dealing with other members as well as with people outside the Team:Byte.M3 community.

*Be careful in the words that you choose.*

We are a community of professionals, and we conduct ourselves professionally. Be kind to others. Do not insult or put down other participants. Harassment and other exclusionary behavior aren't acceptable. This includes, but is not limited to:

Violent threats or language directed against another person.

*Discriminatory jokes and language.*

Posting sexually explicit or violent material.

Posting (or threatening to post) other people's personally identifying information ("doxing").

Personal insults, especially those using racist or sexist terms.

Unwelcome sexual attention.

Advocating for, or encouraging, any of the above behavior.

Repeated harassment of others. In general, if someone asks you to stop, then stop.

When we disagree, try to understand why. Disagreements, both social and technical, happen all the time and Team:Byte.M3 is no exception. It is important that we resolve disagreements and differing views constructively. Remember that we’re different. The strength of Team:Byte.M3 comes from its varied community, people from a wide range of backgrounds. Different people have different perspectives on issues. Being unable to understand why someone holds a viewpoint doesn’t mean that they’re wrong. Don’t forget that it is human to err and blaming each other doesn’t get us anywhere. Instead, focus on helping to resolve issues and learning from mistakes.

**Data Protection:**

This section is an overview of our site’s security features to keep the system secured and exploit-free:

1. Cross site scripting (XSS) protection
2. Cross site request forgery (CSRF) protection
3. SQL injection protection
4. Clickjacking protection
5. SSL/HTTPS
6. Host header validation
7. Session security
8. User-uploaded content protection

# 10 Conclusions

## 10.1 Achievements

The achievements we have accomplished after the Project are listed below:

* Developed a system that can replace the manual library managing system.
* Developed a database which stores user details and book details.
* Provided reliable search facility for the user.
* Administrator or librarian should have logins.
* Created an easy to understand user friendly environment.
* Attractive user interfaces to navigate through the system for the users.
* The option for students to search online for books in the library
* See if books are reserved, available or borrowed
* Advanced search features to find books
* A student interface with the options for account creation, login, and forgot your password
* Student accounts can see books they have borrowed, due date, and can borrow new books
* Student accounts can also see books which students with similar tastes or departments have borrowed
* A library staff interface with the options for account creation, login, and forgot your password
* Library staff accounts can see total number of books in library, which are available, which are borrowed, which are reserved
* Library staff can add new books, edit their details, or remove existing books
* Library staff accounts can issue books to students, and verify student accounts
* A notice board web site showing all upcoming seminars, workshops and events or new facilities in university
* University staffs can update notice board and modify it as required on timely basis
* An all access super-user account which has permission to access and privileges to everything
* Special permissions to access and privileges for each type of accounts
* Search filtering
* Special filter sidebar
* Secure password
* Login using Facebook
* Responsive stable web application for all platforms
* A university site page with notices, and information about events, and date and venue
* Departments or register office or staff can add, update, delete events
* Students can see event information
* Has a Powerful look for engine
* Maintaining any media
* Book issuing books Online
* Requesting column for librarian for providing new books
* a separate digital library repository
* Student login page where student can find books issued by him/her and date of return.
* A search column to search for books
* A teacher login page where teacher can add any events being organized in the college and important suggestions regarding books.
* Online notice board about the workshop.
* A librarian login page.

## 10.2 Critical Appraisal

This section contains a dispassionate and detailed discussion and analysis of the work and its outcomes, both positive and negative. It demonstrates the knowledge and expertise that we have gained from our project.

## 10.3 Limitations

One of the major limitations of our Project as of now is that we have not been able to deploy it.

There are several reasons why this has taken place.

There are quite a limited number of sites that provide free web hosting for Django/Python.

Those which are free are not well documented nor user friendly and have a large setup and overhead cost and high difficulty learning curve.

It requires us to be an expert in deploying on a live remote server which with time we will accomplish, but not in the limited time we have at the moment.

Due to time constraints, we have also not been able to complete our documentation and user guide, which is in progress. It will be uploaded on our Github Repository as it progresses.

We have completed the social networking aspect for our site. But we have not been able to complete making the interface as we had hoped.

The email and phone notifications, and Facebook login, are in place, but will require a paid plan to function.

The text and information in the front-end needs to be customized as per company.

Other than this we have met all of the goals we set for the project. We sincerely hope this will be considered as we took on a large project and after months of hard work, we got all of it done, and as an alternative we have added a video showing the Project at work in our Desktop server.

## 

## 10.4 Future Work

One other main parts which we need to work on is getting our web application live.

To do that we will need to learn the in-s and out-s of WSGI (Web Server Gateway Interface) and understand how to write scripts in a remote Linux server which will connect our web application to a domain provider of our choice.

If required we will also scale our application to support more users’ requests, traffic and load.

We will also work on the User Interface more so as to make it more user friendly and provide a better User Experience.

The Documentation and User guide, which is in progress, will be uploaded to our GitHub repository from time to time as it progresses.

Use paid plans to enable the email and phone notifications and the Facebook login.

# 11 Group Reflections

The project we took on has been very ambitious and challenging, and the output very rewarding.

As would be the case, it would be difficult for any individual to carry out all the work on their own, so it was crucial that all the tasks in the project were divided, and shared amongest team members as per their area of expertise.

It was a new first-time experience for the Team, working on a full-fledged production ready real world application.

To prevent the tasks of the individual members from getting too tiresome all members were given parts of the Front-End and Back-End development which we all did quite well.

Kazi Imdadul Haq, Leon Hossain, Sutapa **<tomader naam eikhane diba>** and Syed Shams Elahi, have all worked their level best to try to take this opportunity this semester to learn and make something quite remarkable.

Listed below are some of the problems we encountered through our development cycle and how we solved them:

1. Setting up the development environment:

Problem:

It took us quite a bit of time setting up our Python development environment.

The installation of Python on a Windows-based OS is quite troublesome, with several bugs and errors which require some specific expertise to resolve. And as we are using the latest version of the tools, they were not as well documented at the time.

Solution:

To work around this, we had to get a Desktop Computer with SSD support, and install a Linux Mint Virtual Machine on it. It took time to set up the virtual machine and make it compatible to install and run Python, but after a month of work, consulting forums, blogs, FAQs, Qoura, StackOverFlow, StackExchange and Facebook Groups, we finally managed to fix all the errors and make our development server work on our Desktop Computer.

1. Converting HTML files to Django Template Files:

Problem:

HTML files need to be converted to Django Template Files to be rendered by the server. As of the moment, there is no easy and fast way to do this. One has to edit the files manually, which is quite tiresome and time expensive.

Solution:

We created a small helper library (LINK here !!!!!) to help convert a significant portion of the HTML files to Django Template Files. Leaving only some of it to be done manually.

1. Time Management:

Problem:

It was difficult for the Team to manage free time from our clashing and haphhazard class routine to sit and discuss about the project and work on it together.

Solution:

We set up a Facebook Workplace Group Chat, where we kept in touch with one another through our busy schedule and kept communication smooth to finish our tasks.

1. Task Completion:

Problem:

At times some of the members failed to submit their part of the tasks, due to exams, or the submitted task was faulty, and this led to a clot in the bottleneck of the development process.

Solution:

To remedy this, other members stood up and helped complete the task at hand so as to keep the progress going.

**Lessons Learned:**

Through this semester, this project has taught us several crucial things which we had not comprehended before.

We have learned how to -

1. Communicate with our clients and get the functional requirements from the user and come to terms with what features they want and what would be good for them.
2. Set up our Development Stack.
3. Work as a Team and meet the due-dates.
4. Codify what was required.
5. Work with Django and Python.
6. Full-stack Web Development.
7. Using SQL.
8. Better time management.
9. Better task organization and separation of concerns.

**What could be done better or differently:**

1. We could have set up the development environment faster, as it resulted in taking up much of our time and cutting down progress.
2. We could have learned how to host our web application on a live server.
3. We could work more efficiently if we had a clear schedule (with no classes) and work more effectively.

12 Bibliography and References

Django Documentation

Holovaty, A. (2018). Django Documentation - FAQ | Django. [online] Djangoproject.com. Available at: <https://docs.djangoproject.com/en/2.0/>

[Accessed 31 Jul. 2018].

Django Developer’s Code of Conduct

Holovaty, A. (2018). Django Code of Conduct - FAQ | Django. [online] Djangoproject.com. Available at: <https://docs.djangoproject.com/en/2.0/>

[Accessed 31 Jul. 2018].

13 Appendix E – Project Presentation

[ Include here the slides or documents presented for the Project Presentation ]

14 Appendix X – As required

***Settings.py***

"""

Django settings for locallibrary project.

Generated by 'django-admin startproject' using Django 2.0.4.

For more information on this file, see

https://docs.djangoproject.com/en/2.0/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/2.0/ref/settings/

"""

import os

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = '&$yxf80hkbw%xglo3o6+hzin82&$)t2-f=7u1x!8p05e24&wlu'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'admin\_view\_permission',

# 'django.contrib.admin',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

# 'users',

'catalog.apps.CatalogConfig',

'message\_board.apps.MessageBoardConfig',

"django\_custom\_user\_migration",

# ...

'debug\_toolbar',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

'debug\_toolbar.middleware.DebugToolbarMiddleware',

]

ROOT\_URLCONF = 'locallibrary.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': ['./templates',],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'locallibrary.wsgi.application'

# Database

# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

# Password validation

# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/2.0/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.0/howto/static-files/

STATIC\_URL = '/static/'

CACHES = {

'default': {

'BACKEND': 'django.core.cache.backends.dummy.DummyCache',

}

}

def show\_toolbar(request):

return True

DEBUG\_TOOLBAR\_CONFIG = {

"SHOW\_TOOLBAR\_CALLBACK" : show\_toolbar,

}

LOGIN\_REDIRECT\_URL = '/'

# AUTH\_USER\_MODEL = 'users.CustomUser'

*Urls.py:*

"""locallibrary URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/2.0/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path

from django.conf.urls import include

from django.views.generic import RedirectView

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('admin/', admin.site.urls),

path('library/', include('catalog.urls')),

path('notice/', include('message\_board.urls')),

path('', RedirectView.as\_view(url='/library/')),

path('accounts/', include('django.contrib.auth.urls')),

] + static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

from django.conf.urls import include, url

if settings.DEBUG:

import debug\_toolbar

urlpatterns = [

url(r'^\_\_debug\_\_/', include(debug\_toolbar.urls)),

] + urlpatterns

from django.urls import path

from . import views

from message\_board.views import HomePageView, AboutPageView, ContactPageView

urlpatterns = [

path('', views.index, name='index'),

path('books/', views.BookListView.as\_view(), name='books'),

path('book/<int:pk>', views.BookDetailView.as\_view(), name='book-detail'),

path('authors/', views.AuthorListView.as\_view(), name='authors'),

path('author/<int:pk>', views.AuthorDetailView.as\_view(), name='author-detail'),

path('mybooks/', views.LoanedBooksByUserListView.as\_view(), name='my-borrowed'),

path('about/', AboutPageView.as\_view(), name='notice\_about'),

path('contact/', ContactPageView.as\_view(), name='notice\_contact'),

# re\_path(r'^book/(?P<pk>\d+)$', views.BookDetailView.as\_view(), name='book-detail'),

# path('url/', views.my\_reused\_view, {'my\_template\_name': 'some\_path'}, name='aurl'),

# path('anotherurl/', views.my\_reused\_view, {'my\_template\_name': 'another\_path'}, name='anotherurl'),

]

# from django.urls import path

# from . import views

# urlpatterns = [

# ]

"""bookmarks URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/2.0/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('', include('django.contrib.auth.urls')),

path('admin/', admin.site.urls),

path('account/', include('account.urls')),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL,

document\_root=settings.MEDIA\_ROOT)

from django.urls import path

from message\_board.views import HomePageView, AboutPageView, ContactPageView, NoticeDetailView

# from django.conf import settings

# from django.conf.urls.static import static

urlpatterns = [

path('', HomePageView.as\_view(), name='notice\_home'),

path('<int:pk>', NoticeDetailView.as\_view(), name='notice-detail'),

path('about/', AboutPageView.as\_view(), name='notice\_about'),

path('contact/', ContactPageView.as\_view(), name='notice\_contact'),

]

# + static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

from django.urls import path

from . import views

from django.contrib.auth import views as auth\_views

urlpatterns = [

# post views

# path('login/', views.user\_login, name='login'),

path('login/', auth\_views.LoginView.as\_view(), name='login'),

path('logout/', auth\_views.LogoutView.as\_view(), name='logout'),

path('', views.dashboard, name='dashboard'),

path('password\_change/', auth\_views.PasswordChangeView.as\_view(), name='password\_change'),

path('password\_change/done/',

auth\_views.PasswordChangeDoneView.as\_view(),

name='password\_change\_done'),

path('password\_reset/',

auth\_views.PasswordResetView.as\_view(),

name='password\_reset'),

path('password\_reset/done/',

auth\_views.PasswordResetDoneView.as\_view(),

name='password\_reset\_done'),

path('reset/<uidb64>/<token>/',

auth\_views.PasswordResetConfirmView.as\_view(),

name='password\_reset\_confirm'),

path('reset/done/',

auth\_views.PasswordResetCompleteView.as\_view(),

name='password\_reset\_complete'),

path('register/', views.register, name='register'),

path('edit/', views.edit, name='edit'),

]