Final Year 2 Project Documentation

Online Bookshop with Stock Control

Created by,

*X00179223 -* Raymond McCarthy

*X00167646 -* Taiwo Obadare

*X00162597 -* Vishvajit Sundarrajan

Supervised by,

- Patricia Magee

Contents

[Members Responsibilities 2](#_Toc70325248)

[Responsibilities of team member: Raymond McCarthy – X00179223 2](#_Toc70325249)

[Responsibilities of team member: Taiwo Obadare – X00167646 2](#_Toc70325250)

[Responsibilities of team member: Vishvajit Sundarrajan – X00162597 2](#_Toc70325251)

[Project Introduction 2](#_Toc70325252)

[Project Plan and Requirements 2](#_Toc70325253)

[Introduction 2](#_Toc70325254)

[Project Design 3](#_Toc70325255)

[Overall Goals: 3](#_Toc70325256)

[*Requirements*: 3](#_Toc70325257)

[*High priority:* 3](#_Toc70325258)

[*Medium priority:* 4](#_Toc70325259)

[*Low priority:* 4](#_Toc70325260)

[Class Diagram 4](#_Toc70325261)

[Use Case Diagram 4](#_Toc70325262)

[Use Case Descriptions 5](#_Toc70325263)

[Goals 9](#_Toc70325264)

[Iteration 1 (Prototype) – Delivery date: **5/3/21 – 10pm** 9](#_Toc70325265)

[Iteration 2 (Prototype) – Delivery date: **26/3/21 – 10pm** 10](#_Toc70325266)

[Iteration 3 (Final Iteration) – Delivery date: **26/4/21 – 10pm** 10](#_Toc70325267)

[Assessing Feasibility 12](#_Toc70325268)

[Financial Feasibility 12](#_Toc70325269)

[Technical feasibility 12](#_Toc70325270)

[Resource and time feasibility 12](#_Toc70325271)

[Risk feasibility 12](#_Toc70325272)

[Iteration 1 Report 12](#_Toc70325273)

[Iteration 2 Report 13](#_Toc70325274)

[Group 16: Book Store Project 13](#_Toc70325275)

[Team member: 13](#_Toc70325276)

[Progress made 13](#_Toc70325277)

[Test Cases 13](#_Toc70325278)

[Team member: 14](#_Toc70325279)

[Progress made. 14](#_Toc70325280)

[Test Cases 15](#_Toc70325281)

[Team member 16](#_Toc70325282)

[Progress made: 16](#_Toc70325283)

[Test Cases 17](#_Toc70325284)

[Iteration 3 Report 17](#_Toc70325285)

[Conclusion 17](#_Toc70325286)

[Achievements 17](#_Toc70325287)

[Limitations 17](#_Toc70325288)

[Possible Future Enhancements 17](#_Toc70325289)

[Declaration of Academic Integrity 17](#_Toc70325290)

# Members Responsibilities

## Responsibilities of team member: Raymond McCarthy – X00179223

## 

## Responsibilities of team member: Taiwo Obadare – X00167646

## 

## Responsibilities of team member: Vishvajit Sundarrajan – X00162597

# Project Introduction

# Project Plan and Requirements

## Introduction

For the second-year project our team consisting of Raymond McCarthy, Taiwo Obadare, and Vishvajit Sundarrajan, will create an electronic commerce website with the primary goal of selling goods for profit.

The type of e-commerce store we have chosen to create is a bookstore with stock control capabilities using Python, HTML, CSS, and the Django Framework. We will be using Microsoft Azure DevOps for project management and version control.

## Project Design

### Overall Goals:

Our end goal is to have a fully functional, easy to navigate user friendly website with all the functionality and features necessary to operate as a secure e-commerce store.

We will design database models to store information about our customers, products, and orders. Products will be uniquely identifiable with the use of UUID modelling. We will develop administration account types that are authorized to view, add, and update database information.

The website will have the ability for our customers to sign up, sign in and sign out. Our website will have a clear and organized visual user interface with the use of images to present our products, a navigation bar to traverse the website and drop-down menu links to access account features.

We will include pages for customer reviews as well as a contact us page with relevant information, these pages will use Django forms to allow customers to submit information.

Books will be easily located with a search feature and separated into relevant categories. Products can be viewed by category, product detail, and all products.

We will make use of the Django session framework to store cart information for our users. For credit card processing we will implement a Stripe payment facility. After making payment our customers will be redirected to a thank you page and can view their order details. We will also implement the ability to request a refund from a previous order.

We will make use of the slug field to make sure our products are easily found by search engines. And implement a voucher feature for sales and offers.

Our website will be well tested using software quality and assurance testing techniques for error handling and bug detection.

And finally, when requirements have been met, we will launch the website on a domain server.

## 

### Requirements:

### *High priority:*

* Web site structure.
* Web page design.
* Sign up/log in/log out.
* Database models.
  + Customers
  + Products
  + Orders
  + UUID
* Shopping cart functionality
  + Add items to cart.
  + Remove items from cart.
  + Empty cart.
  + Finalize order.
* Website deployment.
* Make payment.
* Django admin for performing add/update/delete.
* Django session framework to manage shopping cart.
* Stripe API functionality.

### *Medium priority:*

* Graphics and images.
* Review page.
* Paginator.
* Search.
* Navbar links.
* Contact us page.
* Returns.

### *Low priority:*

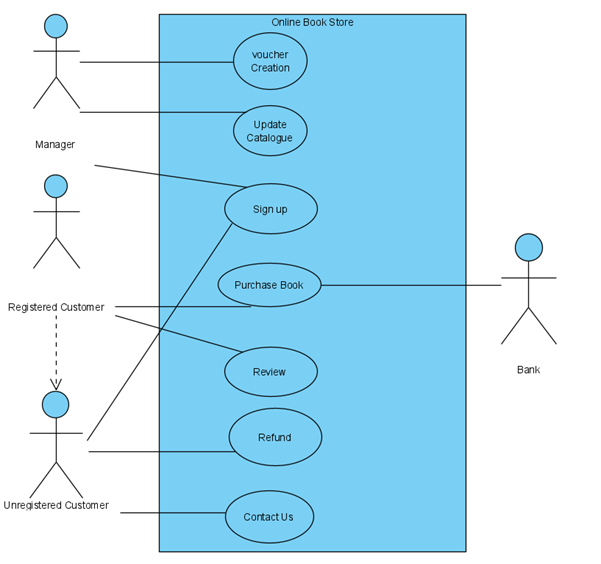
* Slug.
* Voucher

## Class Diagram

Diagram

Description automatically generated

## Use Case Diagram



## Use Case Descriptions

|  |  |
| --- | --- |
| **Use Case:** | Customer Sign up |
| **Actor (s):** | Unregistered customer |
| **Goal:** | Create an account for an individual customer, that stores their information for future logins and purchases |
| **Overview:** | The customer is registered and logged in. The customer is now present in the customer database. |
| **Pre-Condition:** | The customer must be unregistered and not logged into an account. |
| **Post Condition:** | The customer is registered and logged in. The customer is now present in the customer database. |
| **Successful Scenario:** | 1. Customer will land on homepage and be met with a ‘Sign Up’ link. 2. Customer clicks link and is redirected to sign up page. 3. Customer enters their chosen username, email address and password. 4. Customer confirms correct password. 5. Customer clicks ‘Sign Up’ submit button. 6. Customer is redirected to success page. 7. Customer is now registered. |
| **Alternative Scenario (s)** | 1. The customer’s details are unsuccessful due to already being registered. |

|  |  |
| --- | --- |
| **Use Case:** | Registered customer purchases book |
| **Actor (s):** | Registered customer |
| **Goal:** | Registered customer successfully purchases book |
| **Overview:** | The payment has been successful, and the customer is issued an order number. |
| **Pre-Condition:** | The customer must be registered. |
| **Post Condition:** | The customer has made a successful book purchase. |
| **Successful Scenario:** | 1. Customer will land on homepage and select log in, if not already logged in. 2. Customer enters username and password on log in page and clicks submit. 3. Customer is redirected to success page. 4. Customer searches for book. 5. Customer adds book to cart. 6. User navigates to cart and chooses pay now option. 7. Stripe API opens and customer is directed to enter payment details. 8. Customer enters credit card information and clicks submit. 9. Card payment is authenticated by stripe. 10. Payment is successful and order is logged. |
| **Alternative Scenario (s)** | 1. The customer’s payment is unsuccessful, and the ordered is not approved. |

|  |  |
| --- | --- |
| **Use Case:** | Unregistered customer purchases book |
| **Actor (s):** | Unregistered customer |
| **Goal:** | Unregistered customer successfully purchases book |
| **Overview:** | The payment has been successful, and the customer is issued an order number. |
| **Pre-Condition:** | N/A |
| **Post Condition:** | Customer has made a successful book purchase |
| **Successful Scenario:** | 1. Customer searches for book. 2. Customer adds book to cart. 3. Customer navigates to cart and chooses ‘pay now’ option. 4. Stripe API opens and customer is directed to enter payment details. 5. Customer enters credit card information and clicks submit. 6. Card payment is authenticated by stripe. 7. Payment is successful and order is logged. |
| **Alternative Scenario (s)** | 1. The customer’s payment is unsuccessful, and the ordered is not approved. |

|  |  |
| --- | --- |
| **Use Case:** | Manager updates catalogue |
| **Actor (s):** | Manager |
| **Goal:** | Manager adds a new book, deletes an old book, or updates a books details on the database. |
| **Overview:** | The new book is now visible for customers, the old book is no longer visible, or the book details and description have been updated. |
| **Pre-Condition:** | Manager is logged in with permissions. |
| **Post Condition:** | Database has updated book catalogue. |
| **Successful Scenario:** | 1. Manager logs in with manager permissions. 2. Proceeds to GUI version of the Django database admin. 3. Manager clicks, add, update, or edit book. 4. Manager enters new information and images, or manager removes old book from database. |
| **Alternative Scenario (s)** | 1. The manager cancels changes and logs out. |

|  |  |
| --- | --- |
| **Use Case:** | Manager adds voucher codes |
| **Actor (s):** | Manager |
| **Goal:** | Add voucher codes which will reduce price when entered at checkout. |
| **Overview:** | Voucher is stored in database for use. |
| **Pre-Condition:** | Manager is logged in with permissions. |
| **Post Condition:** | Customers can now enter voucher code at checkout and receive a discounted price. |
| **Successful Scenario:** | 1. Manager logs in with manager permissions. 2. Proceeds to GUI version of the Django database admin. 3. Manager clicks vouchers and adds a voucher code, discount amount and date of validity. 4. Clicks save. |
| **Alternative Scenario (s)** | 1. Code is no longer valid and total price remains unchanged. |

|  |  |
| --- | --- |
| **Use Case:** | Registered Customer reviews |
| **Actor (s):** | Registered Customer |
| **Goal:** | Registered Customer successfully leave a review |
| **Overview:** | Customer will land on homepage and select log in, if not already logged in. Customer will click the review page and if they want to give feedback with the services provided, the Customer can leave a review on the review page. |
| **Pre-Condition:** | The Customer must be registered. |
| **Post Condition:** | The Customer has made a successful review. |
| **Successful Scenario:** | 1. Customer logs in 2. Selects review page 3. Clicks the text box 4. Leaves a review 5. Exits page |
| **Alternative Scenario (s)** | 1. The Customer exceeds the text box character limit 2. Unable to leave a review |

|  |  |
| --- | --- |
| **Use Case:** | Customer uses contact us |
| **Actor (s):** | Customer |
| **Goal:** | Customer successfully contacts us |
| **Overview:** | Customer will land on homepage and select log in, if not already logged in. Customer will click the Contact us page. Customer will input info necessary to contact the online bookstore. |
| **Pre-Condition:** | The Customer must be on the website |
| **Post Condition:** | The Customer has successfully contacted the online bookstore |
| **Successful Scenario:** | 1. Customer goes into the website 2. Customer clicks the contact us page 3. Inputs the required fields data 4. Contacts the online bookstore 5. Exits page |
| **Alternative Scenario (s)** | 1. The customer does not have the required data 2. The customer is unable to contact the online webstore |

|  |  |
| --- | --- |
| **Use Case:** | Customer uses refund |
| **Actor (s):** | Customer |
| **Goal:** | Customer successfully refunds the selected product |
| **Overview:** | Customer will request a refund. Constraint will be checked to match user to product purchased. |
| **Pre-Condition:** | The Customer must be on the website |
| **Post Condition:** | The Customer has successfully refunded product. |
| **Successful Scenario:** | 1. Customer goes into the website 2. Customer clicks the refund button 3. Inputs the required fields data 4. Customer receives payment equivalent to items purchased |
| **Alternative Scenario (s)** | 1. The customer did not purchase items |

## Goals

### Iteration 1 (Prototype) – Delivery date: **5/3/21 – 10pm**

* We will meet for our scrum session on Microsoft teams to discuss the iteration and determine our approach for completing tasks on **16/2/21 at 9am**.
* Tasks set out:
* Develop a working system with core functionality.
  + Initialize and fork repository.
  + Create models.
    - Users
    - Products
    - Admin
  + Develop views, templates, and URLs.
  + Develop add, update, and delete functionality by Admin.
  + Design templates.
    - Base.html
    - Navbar.html
    - Products pages
    - Sign up/in/out.
    - Styling
* Paginator & Search functionality.
* Cart application.
* Design a series of tests to verify that the system is working correctly.
* Document any issues or problems encountered.
* Write a 700-word iteration reporting work completed.

### Iteration 2 (Prototype) – Delivery date: **26/3/21 – 10pm**

* By the time we have reached iteration two we plan to have a well-built foundation with basic functionality to build upon.
* We will meet on Microsoft teams for our scrum session to discuss the approach for our next iteration on **9/3/21 at 9am**.
* Tasks set out:
  + We will now implement additional functionality such as:
    - Strip payment API.
    - Orders app.
    - Thank you, page.
    - Contact us page.
    - Sign up, sign in & sign out.
    - Voucher’s functionality.
    - Many to many mapping
      * This will allow us to add the same product to multiple categories, e.g., ‘Crime’ & ‘Thriller’.
    - Incorporating the Django ‘Slug’ field.
  + More testing for our newly implemented code.
  + Removal of any residual bugs from previous tests.
  + Submit a 700-word report on work completed.

### Iteration 3 (Final Iteration) – Delivery date: **26/4/21 – 10pm**

* We will meet on Microsoft teams on **13/4/21 at 9am** for our scrum session to discuss our approach for the final iteration**.**
* At this stage of the project, we will have completed all coding and testing of the functionality of our website and be focusing on testing and making minor adjustments to improve overall functionality and appearance where necessary.
* We will prepare our final iteration documentation and our final prototype for review before **10pm** on the **26th of April 2021.**
* The documentation will include:
  + A table of contents, numbered headings, page numbers and references
  + An outline of each member’s responsibility and participation within the project
  + An introduction to the project, including project plan and requirements document.
  + Reports for each iteration (Iteration 1, Iteration 2 & Iteration 3).
  + Concluding paragraph highlighting the achievements of the project, any shortcomings and what could be improved by future enhancements.
  + The document should be bound with a cover page indicating:
    - Title of project
    - Names & Student IDs of students
    - Name of Supervisor

## Assessing Feasibility

### Financial Feasibility

This website will be created as an amateur project without labor costs. The tools we have decided to use in developing our project are open source, and the hosting service we have chosen is free of charge. So, our current expenses for the project total are €0.

### Technical feasibility

The tools which we intend to use are:

* HTML5
* CSS
* JavaScript
* Python
* Django

The three members of our team all have relevant experience and knowledge in all the listed technologies and our technical skills have been determined to be sufficient based on our work history.

### Resource and time feasibility

The resources required for this project are:

* Programming devices (laptops/computers)
* Hosting space (freely available)
* Programming tools (open source)
* Programmers

We have concluded that there will be no issue in having access to all the necessary resources we need to produce our final product.

The time we have been allocated for each iteration of the project is 3 weeks which we have deemed perfectly feasible based on previous projects.

### Risk feasibility

Expert understanding of our chosen technologies is not evident amongst the team, although confident, we may assume likelihood of running into time consuming issues while trying to implement functionality into our project with the use of multiple coding languages.

It is our understanding from personal experience that the time that has been allocated to complete our project should be sufficient to allow for coding issues to arise and be dealt with accordingly, using online tutorials and forums such as stackoverflow.org, brainstorming amongst each-other and conferring with our supervisor.

# Iteration 1 Report

# Iteration 2 Report

### Group 16: Book Store Project

*Raymond McCarthy, Taiwo Obadare, Vishvajit Sundarrajan.*

*Supervisor: Patricia Magee.*

Team member: **Raymond McCarthy**

### Progress made

**Cart application:** Development of cart application and template to display cart content, with view functions to allow a user to add books to cart, remove a single book from cart, and remove all items from the cart. And models to store cart content information.

**Manager/customer roles:** Use of Django roles, groups, and permissions to develop manager and customer user types. Manager was given high-level permissions, whilst customer was given limited permissions.

**Sign up/in/out functionality:** Development of templates that display forms for users to sign up and sign in. View functions that allow users to sign up, sign in and sign out and store their details into models in the database.

**'Contact Us' application:** Development of a contact template with a submission form and a ‘thank you/confirmation’ template which renders upon successful submission. A view function to ‘post’ entries and a model to store the data entered.

**Refund functionality:** Development of templates with a form to submit a refund application and success page. Updated order model to include a reference code, delivery status, refund status. View function to take form data and update order object. Model to store refund records.

**Slug:** Updated book and category models to include a slug field with autoslug. Updated all templates and functions that use a get method or dot operator on books or categories, to get objects by slug field. Updated all uuid URLs to slug URLs.

**Password change form:** Developed a view function to change password, and template to change the style of the Django admin password\_change form to fit the front-end styling.

### Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Matrix** | | | | | |
| **Test Case No** | **Test Case Name** | **Input** | **Expected Outcome** | **Actual Outcome** | **Result (Pass/Fail)** |
| 01 | Test\_blog\_create\_success | Create blog object, test model fields present | Blog record created and initialised with fields | Blog record created and initialised with fields | Pass |
| 02 | test\_cart\_create\_success | Create cart object, test model fields present | Cart record created and initialised with fields | Cart record created and initialised with fields | Pass |
| 03 | Test\_cart\_item\_create\_success | Create cart item object, test model is present | Cart item record created and initialised with fields | Cart item record created and initialised with fields | Pass |
| 04 | test\_contact\_create\_success | Create contact object, test model is present | Contact record created and initialised with fields | Contact record created and initialised with fields | Pass |
| 05 | test\_order\_create\_success | Create order object, test model fields present | Order record created and initialised with fields | Order record created and initialised with fields | Pass |
| 06 | test\_order\_item\_create\_success | Create order item object, test model fields present | Order item record created and initialised with fields | Order item record created and initialised with fields | Pass |
| 07 | test\_book\_is\_assigned\_slug\_on\_creation | Create book object, test slug field is assigned on creation | Book is created and automatically assigned a slug field | Book is created and automatically assigned a slug field | Pass |
| 08 | test\_category\_is\_assigned\_slug\_on\_creation | Create category object, test slug field is assigned on creation | Category is created and automatically assigned a slug field | Category is created and automatically assigned a slug field | Pass |

Team member: **Taiwo Obadare**

### Progress made.

**User Review functionality:** Authenticated user can write a review, which is displayed with the date posted and username. Was queried with newest review displayed first on the page. JavaScript was used to show number of reviews displayed currently on page and to hide the review form at first and display it on user click. Furthermore

**Blog Functionality:** App allows a user to write a blog about particular book, displays book at the end of blog body. Reviews by writer of blog displayed at the right and a link to the relevant book user has reviewed in the past. Was queried in the view to only allow the first 5 reviews from the author

**Wish List Functionality:** This feature is created by a model, that has the user, the book etc. The book\_detail.html page has a link that when clicked calls on the add\_wishList view wich accepts the book slug on the link and adds to the Wishlist database by current user. It tells user already added through django message framework, else if it does not exist in the get query it creates the object and saves the object to the database displaying the success message. User can view their Wishlist and delete/add to cart.

**Stripe/PayPal payment:** Payment functionality using stripe and PayPal, which displays the relevant price of the book(s). And allows a user complete purchase.

**Website Styling:** Improved overall website design.

**Manager Authentication/CRUD Front End:** Allows for the manger to access the add, view, delete, edit functionality only. And updates the book in the database and add all fields except image when updating and adding book.

**\*Only first testcase URL from each app will be written.**

### Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Matrix (URLS)** | | | | | |
| **Test Case No** | **Test Case Name** | **Input** | **Expected Outcome** | **Actual Outcome** | **Result (Pass/Fail)** |
| 01 | Test blog list URL is resolved with function blogView. | Reverse of blog list url and resolve to get function | BlogView  The relevant view corresponding to the URL | BlogView  The relevant view corresponding to the URL | Pass |
| 02 | Test add\_cart URL is resolved with function add\_cart | Reverse of add\_cart url and resolve to get function | add\_cart  The relevant view corresponding to the URL | add\_cart  The relevant view corresponding to the URL | Pass |
| 03 | Test contact\_url  URL is resolved with function contactView | Reverse of contact\_url and resolve to get function | contactView  The relevant view corresponding to the URL | ContactView  The relevant view corresponding to the URL | Pass |
| 04 | Test topRatedBooks URL is resolved with function topRatedBooks | Reverse of topRatedBooks url and resolve to get function | topRatedBooks  The relevant view corresponding to the URL | TopRatedBooks  The relevant view corresponding to the URL | Pass |
| 05 | Test apply\_url  is resolved with function voucher\_apply | Reverse of apply url and resolve to get function | voucher\_apply  The relevant view corresponding to the URL | voucher\_apply  The relevant view corresponding to the URL | Pass |
| 6 | Test apply\_url  signup\_url  is resolved with function signupView | Test signup\_url  is resolved with function signupView | signupView  The relevant view corresponding to the URL | signupView  evant view corresponding to the URL | Pass |
| 7 | Test Thanks\_url  is resolved with function voucher\_apply | Test  Thanks\_url  is resolved with function voucher\_apply  thanks | thanks  The relevant view corresponding to the URL | thanks  The relevant view corresponding to the URL | Pass |

Team member: **Vishvajit Sundarrajan**

### Progress made:

**Thanks Page:** Developed a thank you template which upon creation of a successful order displays a confirmation of the user's purchase gives them order id as well as displaying message of thanks to the customer.

**Order Functionality:**  This functionality stores the user's personal information in the order model. Then calculates the total costs of the user’s purchases based on the number of the items in the user’s cart. The user’s information is then displayed with relevant corresponding payment information of the user at checkout.

**Order history page:**  Users who have made purchases have the ability to view all purchases made on this page. When the users click on a view order link it will display all the information of the previously made purchases associated with their accounts.

**Vouchers:** Implemented voucher functionality which allows users with manager privilege to create

discount vouchers. The which users can use before payment to reduce overall price of order items.

**Website Styling:**  Improved the website layout.

**Search Functionality:** Implemented the search functionality of the website in which the user can search a book based on category or author name and is taken to the relevant page for the book requested.

### Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Matrix** | | | | | |
| **Test Case No** | **Test Case Name** | **Input** | **Expected Outcome** | **Actual Outcome** | **Result (Pass/Fail)** |
| 01 | test\_cheap\_books | Cheap books url | Status code of cheap book page received | Status code of page received | Pass |
| 02 | test\_topRatedBooks | Top rated books url | Status code of top rated book page received | Status code of cheap book page received | Pass |
| 03 | test\_book\_detail | Book detail url | Status code of top rated book page received | Status code of top rated book page received | Pass |
| 04 | Test\_add\_to\_wishList | Wish List url | Status code of top rated book page received | Status code of top rated book page received | Pass |
| 05 | Test\_delete \_f |  |  |  |  |

# Iteration 3 Report

# Conclusion

## Achievements

## Limitations

## Possible Future Enhancements

# Declaration of Academic Integrity