National College of Ireland

Higher Diploma in Science in Web Technologies

Final Report

The Book Boutique

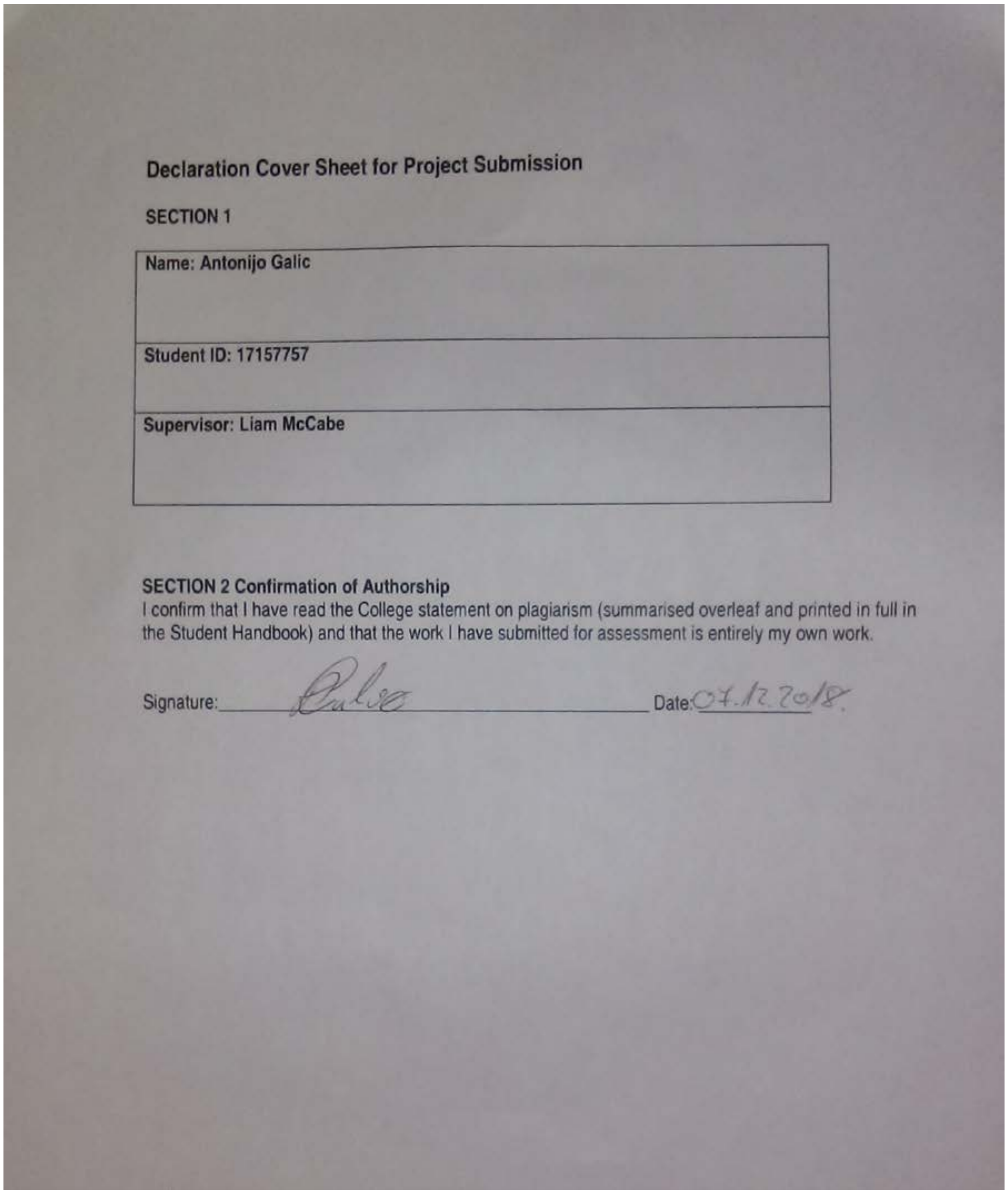
Student Name: Antonijo Galic

Student Number: 17157757

Email address: x17157757@student.ncirl.ie

07. 12. 2018.

Declaration Cover Sheet for Project Submission



Declaration Cover Sheet for Project Submission

SECTION 1

Name: Antonijo Galic
Student ID: 17157757
Supervisor: Liam McCabe

SECTION 2 Confirmation of Authorship

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: [illegible] Date: 07.12.2018.

Table of Contents

Executive Summary

An online bookstore system is the main function of the web application where the user can check the book details, can register/login to buy the book, can add a book to wishlist or can add a book on the reading list.

In order to implement the web application, it is used Node.js, Express, MongoDB to create a RESTful web application. Node.js run server together with Express what is Node framework. The server storing data in the MongoDB database hosted at mlab.com. Passport.js and bcryptjs are two Node middleware which are responsible for user authentication (passport) and encryption of sensible data (bcryptjs). For validation and charging credit cards, and available payment process it is implemented Stripe. To dynamically generate HTML page, it is used Handlebars logic-less templating engine. Bootstrap framework which is based on flexbox layout providing responsiveness of web application. After successfully developed web application is deployed to Heroku service.

In order to give the best Performance, Accessibility, Best practices and excellent SEO (Search Engine Optimization) for evaluation is used Lighthouse. During three steps of evaluation the Performance of web application raised from 45% to 68%, Accessibility from 57% to 70% and SEO from 89% to 100% when Best Practices remain all time on 93%. After the last report the developer is satisfied with web application.

1 Introduction

The purpose of this document is to give a final overview of developing a web application to function as an online e-commerce web application called The Book Boutique.

The document itself is a combination of Project Proposal document, Requirements Specification document and Product Design Specification document together with a final overview on project code and design of web application.

1.1 Background

An online bookstore is a form of e-commerce and book sales industry in one form, it has many advantages, such as bookstore size is relatively small, cost savings, transaction activities can be anytime, anywhere, improve service efficiency. (Liu, 2015) Online bookstore system is the main function of the trading platform for the site, consumers can connect to the Internet through the computer into the online bookstore and then check the book information, if you need to purchase should be registered landing, select their own books, submit orders and pay to complete the entire book ordering process, to achieve online transactions. (Zhai & Lu, 2017)

In July 1995, Amazon, the first online bookstore, launched a new business model by selling books online. This event-initiated competition between online bookstores and physical (brick-and-mortar) bookstores. Now, Amazon has grown to become the largest online bookstore in the world. Amazon’s success leads to the popularity of online bookstores.

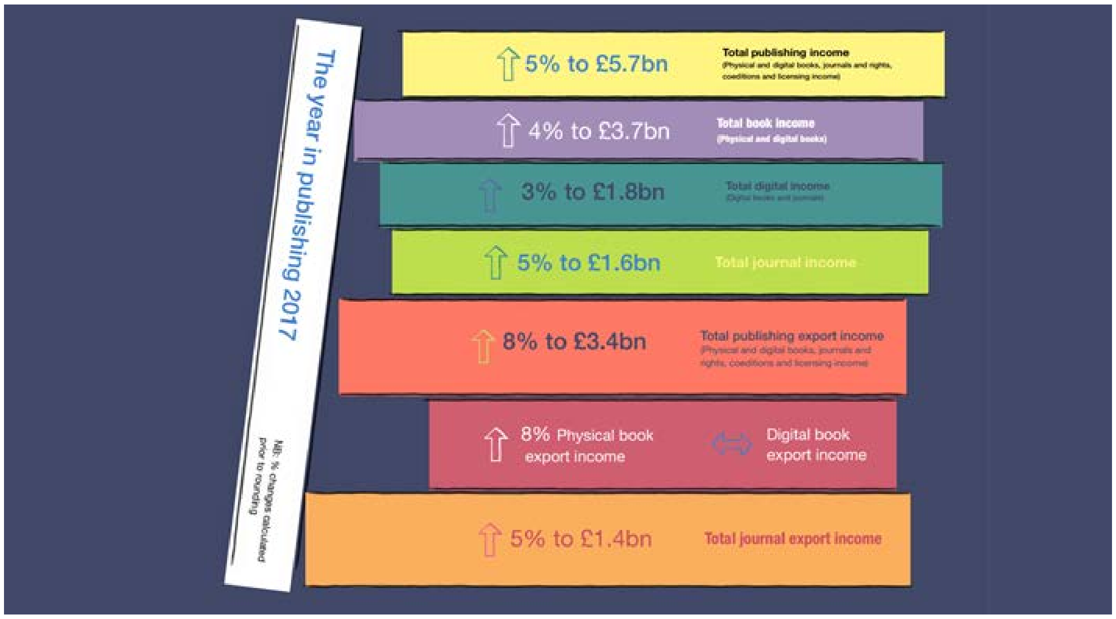


Figure 1. The Year in Publishing 2017. (Anderson, 2018)

Another sign to take before developing an online bookstore is the size of the publishing industry. The book publishing industry is a billion-dollar industry (on the image above numbers about UK publishing industry). According to statista.com (Unknown, 2018), the revenue from the global book publishing market is forecast to slightly increase in the coming years, growing from around 113 billion U.S. dollars in 2015 to about 123 billion U.S. dollars by 2020. British company Pearson is the largest publishing house in the world as of 2015. Besides Pearson, Thomson Reuters, RELX Group, Wolters Kluwer and Penguin Random House are also leading book publishers in the world.

Based on this information, selling books online has a big potential.

1.2 Aims

The general aim of the project is to create a web application to function as an online e-commerce web application for books called The Book Boutique. In order to develop an e-commerce web application that allows fluid and fast environment of the application, the main technologies used to develop this application are:

- Node.js – as an asynchronous vent driven JavaScript runtime, Node is designed to build scalable network applications,

- Express – is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications,
- MongoDB – stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time,
- Stripe – software allows individuals and businesses to receive payments over the Internet. Stripe provides the technical, fraud prevention, and banking infrastructure required to operate online payment systems.

The completed web application is user-friendly and fully responsive so it can automatically adapt to suit all screen resolutions and devices (desktop, laptop, tablet and phone).

The web application has to accomplish the general aim of a web application, aims on user side and aims on the admin side. The general aim of the web application is to introduce the site visitor to the site with an overview of the books where application displaying visually a book with a view to keep user interest and motivate the user to buy a book(s). Aims on user side are based on functionality that allows the user to register/login to interact with the application so the user can:

- buy a product – adding the product to cart and complete payment process after entering card details,
- leave a review – a user can leave a review for a specific book,
- add a book to wishlist – after adding book(s) to wishlist user will have a reminder about the book(s) for purchasing in the future,
- add a book on the reading list – by adding a book on the reading list user has the ability to track books which user already read,
- view orders – user can check own history orders.

Admin side aim is to give admin ability to add, edit and delete books which means that admin has full CRUD (Create, Read, Update, Delete) functionality over the application.

1.3 Technologies

Considering the background of this project, an online bookstore system is the main function that needs to be achieved. To accomplish that functionality user should be able to register/login into the system, add book(s) to cart and enter card details during checkout operation.

Separately from main functionality user is able to add a book to wish list. Also, the user has the possibility to leave a review for each book. Research phase shows that a lot online bookstore doesn’t have the functionality to list the book on reading list so a user can track books in own collection in that way.

The developing of web application is made using Node.js, Express, MongoDB and Stripe to create a RESTful web application. Node.js run server together with Express what is Node framework and it is using JavaScript programming language. For creating an application skeleton, Express application generator is used. The server storing data in the MongoDB database and it is hosted at mlab.com. Mongoose is the ODM (Object Document Mapper) which is used to communicate with MongoDB database.

To dynamically generate an HTML page, it is used Handlebars logic-less templating engine. Handlebars weren’t the first choice but after researching what is in trend (which you can see in the picture below) the application using handlebars instead of the pug(jade). The web application is designed to be fully responsive using Bootstrap (version 4) framework which is based on flexbox layout.

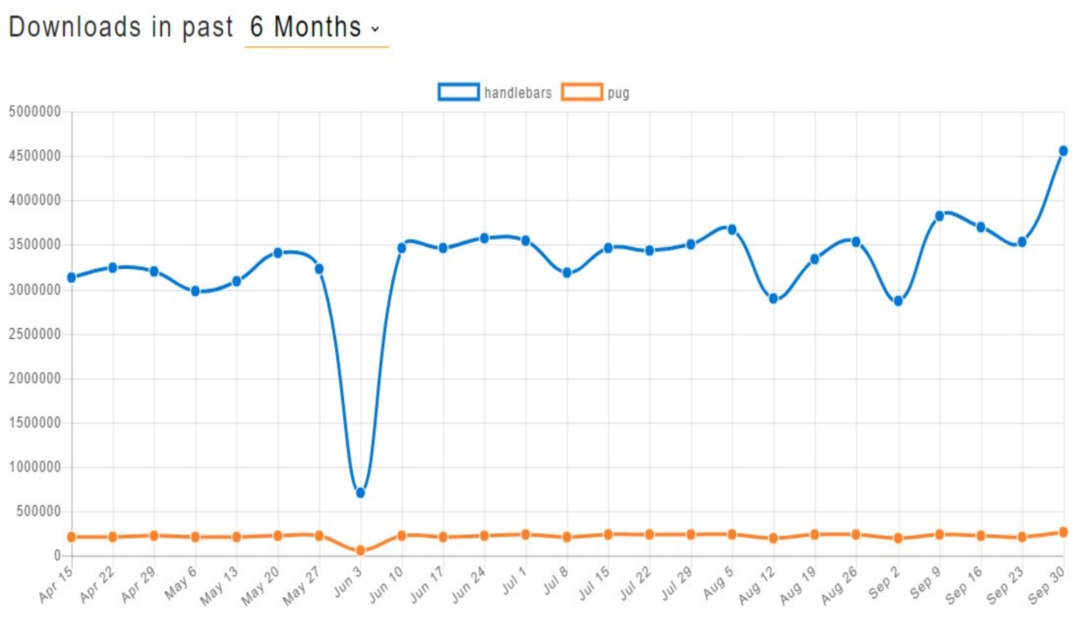


Figure 2. Handlebars vs Pug, npmtrends.com (Unknown, 2018)

Sensible data are encrypted with bcrypt and for user authentication is used passport.js local strategy. For validation and charging credit cards, and available payment process it is implemented Stripe.

Some of the main npm modules that are used in the application are:

- bcryptjs – helps to encrypt passwords
- body-parser – to handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called body-parser. Body-parser extracts the entire body portion of an incoming request stream and exposes it on req.body. The middleware was a part of Express.js earlier but now you have to install it separately. This body-parser module parses the JSON, buffer, string and URL encoded data submitted using an HTTP POST request.
- connect-mongo – a MongoDB session store backed by mongoose
- cookie-parser – passing a secret string, which assigns req.secret so it may be used by other middleware
- csurf - node.js CSRF(Cross-site request forgery) protection middleware
- express – a fast, minimalist, web framework for node

- express-handlebars – a Handlebars view engine for Express
- express-session – create a session middleware with the given options
- hbs – express.js view engine for handlebars.js
- helmet - helps you secure your Express apps by setting various HTTP headers. *It's not a silver bullet*, but it can help! Helmet is a collection of 13 smaller middleware functions that set HTTP headers.
- method-override – allow developer/user to use HTTP verbs such as put or delete
- mongoose – elegant MongoDB object modelling for node.js
- passport – Passport is authentication middleware for Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped into any Express-based web application. A comprehensive set of strategies support authentication using a username and password, Facebook, Twitter, etc. To implement this middleware into our web application it is used passport local strategy.
- stripe – the Stripe Node library provides convenient access to the Stripe API from applications written in server-side JavaScript.

The coding part of the application is done by using Cloud9 IDE. Cloud9 IDE is an online integrated development environment. During the development phase, the code is pushed to GitHub using git version control to prevent loss of data, to have time track about the project and to easy manipulate with the project, especially during the testing phase.

1.4 Structure

Chapter 2: System

This chapter gives a detailed overview of the development of the system for The Book Boutique e-commerce web application. Main areas which are included in this chapter are the Requirements, Design and Architecture, Implementation, Testing, Graphical User Interface (GUI) Layout, Customer testing and Evaluation.

Chapter 3: Conclusions

The third chapter of this document outlines conclusions from the development of web application. These conclusions including advantages and disadvantages in the development of The Book Boutique web application.

Chapter 4: Further development or research

Further development or research chapter displaying where is web application today and where can be in the future. Does application will remain on the same market or it will have upgrade to expand on another market.

Chapter 5: References

This chapter lists all references used or referenced during the development of web application. For this project, it is used Harvard style referencing.

Chapter 6: Appendix

The appendix includes all other documentation or point in that documents which are important for developing of the e-commerce web application. Other documents including the Project Proposal, Project Plan, Requirement Specification, Product Design Specification and Table of Figures.

2 System

2.1 Requirements

This section outlines all functional and non-functional requirements of the e-commerce web application.

System requirement specification or SRS frameworks software development, it documents every operation and dictates how software should behave, it can be as detailed as what a button should do and should be as complete and correct as possible. The purpose of a specification document is to describe the behaviour as well as the different functionalities of an application or software in a specific environment. (Osetskyi, 2018)

2.1.1 Functional requirements

This section usually consists of a hierarchical organization of requirements, with the business/functional requirements at the highest level and the detailed system requirements listed as their child items. (Inflectra, 2018) This section also contains a use case diagram that illustrates how the actors of web application interact with the system and describe all use cases necessary for functional work of e-commerce web application.

2.1.1.1 Use Case Diagram

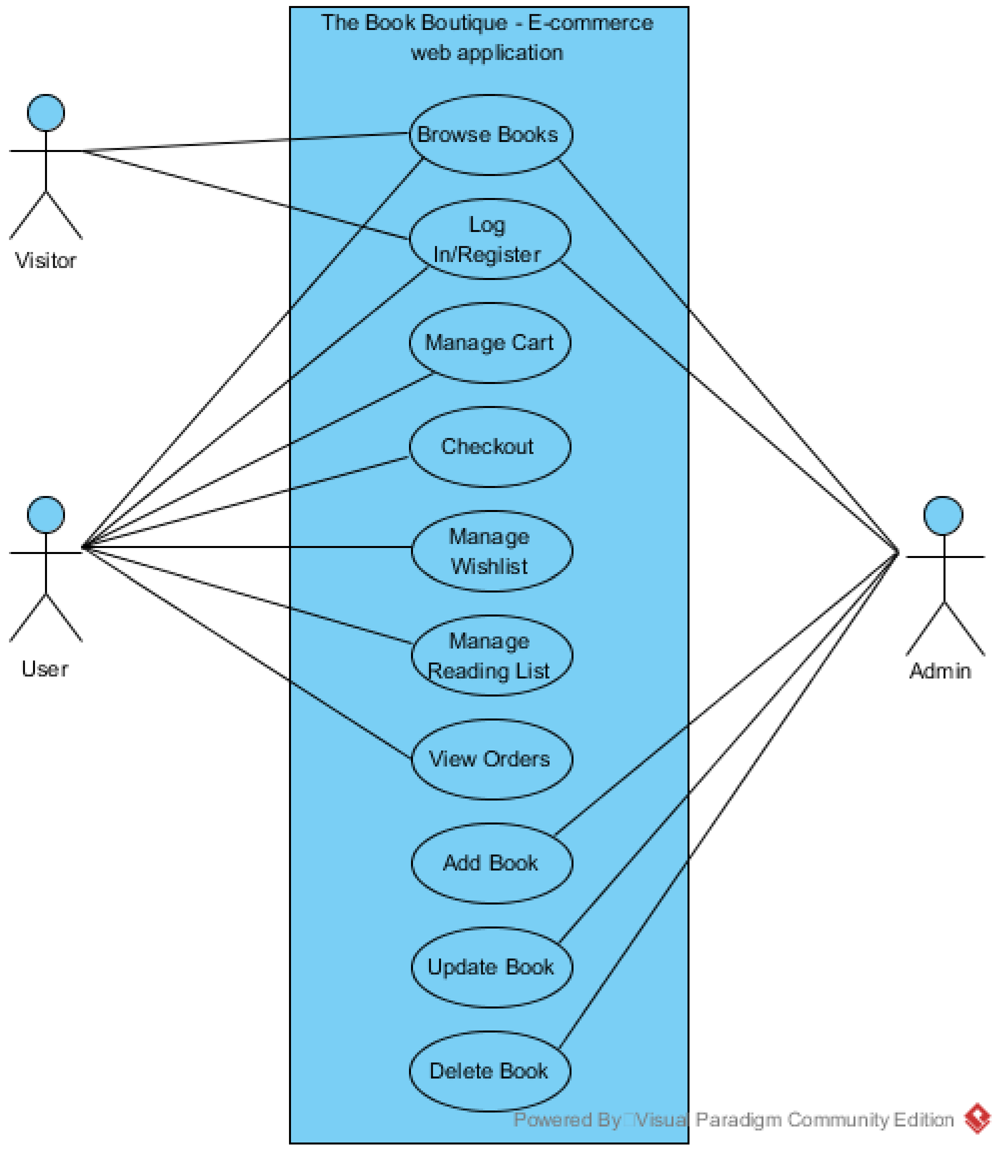


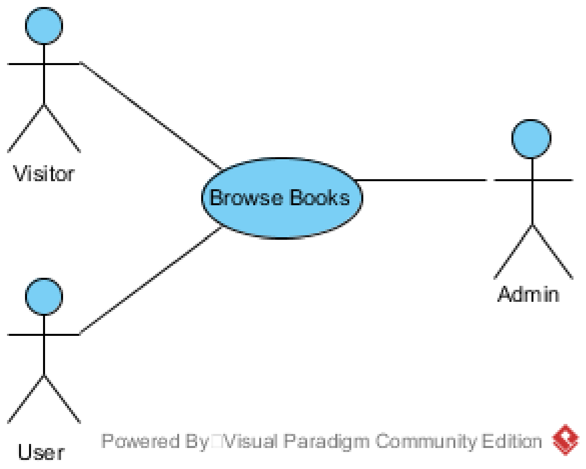
Figure 3. Use case diagram

2.1.1.2 Requirement 1: Browse Books

2.1.1.2.1 Description & Priority

The user navigates to The Book Boutique e-commerce web application to browse books. Browsing books is the first step in using e-commerce web application.

2.1.1.2.2 Use Case



Scope

The scope of this use case is for a user to browse books inside e-commerce web application.

Description

This use case describes how user is browsing books.

Flow Description

Precondition

The user has internet access and wants to browse (potentially buy) books.

Activation

This use case starts when the user opens The Book Boutique e-commerce web application.

Main flow

1. The user can browse books using the navigation bar. [A1 Search books]
2. The e-commerce web application displayed books based on user selection.
3. The user selects a book to view details.
4. The e-commerce web application presents the details about the book with book price and options to add a book to cart, add a book to wishlist or add a book on the reading list.
5. The user selects one of the options. [A2 No interest in the specific book]
6. The user goes back to browse more books.

Alternate flow

(A1 Search books)

1. The user is using the search bar to browse books.
2. The e-commerce web application display search results depend on search keyword.
3. The use case continues at position 3 of the main flow.

(A2 No interest in the specific book)

1. The user doesn’t have an interest in a specific book.
2. The use case continues at position 6 of the main flow.

Termination

The user proceeds to the shopping cart or left the web application.

Postcondition

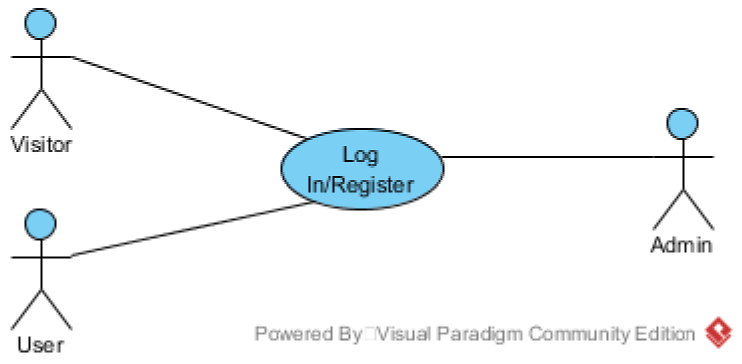
The e-commerce web application goes to wait state.

2.1.1.3 Requirement 2: Log In/Register

2.1.1.3.1 Description & Priority

The use case describes how user is entering into own account on the web application in order to be verified to interact with the application. This requirement is necessary in order to buy a book, add a book to wishlist, leave a review or add a book on the reading list.

2.1.1.3.2 Use Case



Scope

The scope of this use case is to enable the user to enter into own account on the web application.

Description

This use case describes steps to log in into a web application.

Flow Description

Precondition

The e-commerce web application is ready to be activated.

Activation

The user navigates to the web application.

Main flow

1. The user navigates to the navigation bar and clicks Log In button. [A1 Register account]

2. The e-commerce web application returns Log In page with log in form.
3. The user enters the log in details.
4. The user clicks on the Submit button.
5. The e-commerce web application verifies the log in details. [E1 Incorrect log in details]
6. The e-commerce web application allows authenticate access to the web application.

Alternate flow

(A1 Register account)

1. The new user navigates to the navigation bar and clicks Register button.
2. The e-commerce web application returns Register page with the registration form.
3. The user enters personal details to the registration form.
4. The user clicks on the Submit button.
5. The e-commerce web application stores the information.
6. The use case continues at position 6 of the main flow.

Exceptional flow

(E1 Incorrect log in details)

1. The e-commerce web application returns message with incorrect log in details.
2. The e-commerce web application asks user to re-enter log in detail(s).
3. The user re-enters log in detail(s).
4. The use case continues at position 4 of the main flow.

Termination

This use case terminates when web application redirects the user to the web application like authenticate user.

Postcondition

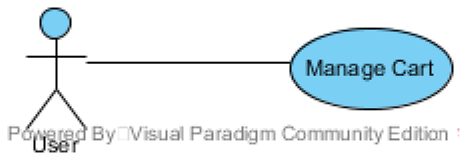
The e-commerce web application goes to wait state.

2.1.1.4 Requirement 3: Manage Cart

2.1.1.4.1 Description & Priority

The use case describes how a user can manage books in the shopping cart. The user can choose a quantity for a specific book, can remove the book from the cart and can see the total price per book and the total amount of cart.

2.1.1.4.2 Use Case



Scope

The scope of this use case is to enable the user to manage own shopping cart.

Description

This use case describes steps to manage books in the shopping cart.

Flow Description

Precondition

The user is logged in into e-commerce web application and the user has added book(s) to the shopping cart.

Activation

The user navigates to Cart button on navigation bar.

Main flow

1. The user navigates to the navigation bar and clicks Cart button.
2. Cart page is displayed to the user.
3. The user checks shopping cart one more time before checkout.
4. The user is satisfied with the book(s) in the shopping cart. [A1 Remove book] [A2 Decrease quantity]
5. The user clicks on the Checkout button.

Alternate flow

(A1 Remove book)

1. The user is not satisfied with the book in the shopping cart.
2. The user clicks Remove button on a specific book in the shopping cart.
3. The book is not anymore displayed in shopping cart.
4. The use case continues at position 3 of the main flow.

(A2 Decrease quantity)

1. The user is not satisfied with the quantity for a specific book.
2. The user decrease quantity for a specific book.
3. The quantity of book is decreased in the shopping cart list.
4. The use case continues at position 3 of the main flow.

Termination

This use case terminates when the user clicks the Checkout button.

Postcondition

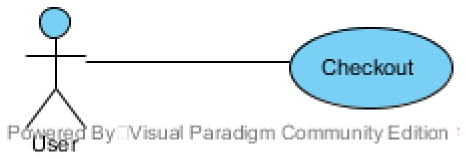
The e-commerce web application goes to wait state.

2.1.1.5 Requirement 4: Checkout

2.1.1.5.1 Description & Priority

The use case describes how user is making payment in e-commerce web application. The user needs to fill up the card detail form with valid information in order to finish the transaction.

2.1.1.5.2 Use Case



Scope

The scope of this use case is to enable the user to make a transaction.

Description

This use case describes steps from Checkout button to the finished transaction.

Flow Description

Precondition

The user is logged in and the cart is populated with the book(s) on user satisfaction.

Activation

The user clicks on the Checkout button in the shopping cart.

Main flow

1. The user clicks on the Checkout button in the shopping cart.
2. The e-commerce web application displays the Checkout page with a form. [A1 Back to shopping cart]
3. The user enters personal details including card details.
4. The user clicks Confirm Payment button.

5. The e-commerce web application together with Stripe API verifies inputs entered from the user. [E1 Incorrect card details]
6. The e-commerce web application redirects the user to the Home page.
7. The message “You have successfully bought the product!” is displayed to the user.

Alternate flow

(A1 Back to the shopping cart)

1. The user decides to change the shopping cart detail(s).
2. The user clicks on the link Return to Cart.
3. The e-commerce web application displays Cart page.
4. The user changes the shopping cart detail(s).
5. The use case continues at position 1 of the main flow.

Exceptional flow

(E1 Incorrect card details)

1. The web application returns a message with incorrect card details.
2. The web application asks from user to re-enter card detail(s).
3. The user re-enters card detail(s).
4. The use case continues at position 4 of the main flow.

Termination

This use case terminates when the user receives a message “You have successfully bought the product!”.

Postcondition

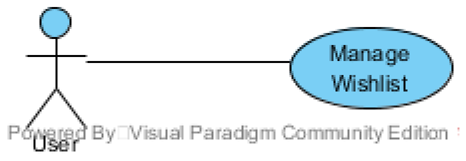
The e-commerce web application goes to wait state.

2.1.1.6 Requirement 5: Manage Wishlist

2.1.1.6.1 Description & Priority

The use case describes how a user can manage a book(s) added to the wishlist. The user can add a book to the cart or remove book(s) from the wishlist.

2.1.1.6.2 Use Case



Scope

The scope of this use case is to enable the user to manage own wishlist.

Description

This use case describes steps to manage books in the wishlist.

Flow Description

Precondition

The user is logged in into e-commerce web application and the user is adding book(s) to wishlist.

Activation

The user navigates to Wishlist button on navigation bar.

Main flow

1. The user navigates to the navigation bar and clicks Wishlist button.
2. Wishlist page is displayed to the user.
3. The user checks books in the wishlist. [A1 Add book to cart]
4. The user is satisfied with the book(s) in the wishlist. [A2 Remove book]
5. The user leaves the Wishlist page.

Alternate flow

(A1 Add book to cart)

1. The user decides to buy a book from wishlist.
2. The user clicks Add to cart button for a specific book.
3. The user bought a specific book.
4. The use case continues at position 3 of the main flow.

(A2 Remove book)

5. The user is not satisfied with the book in the wishlist.
6. The user clicks Remove button on a specific book in the wishlist.
7. The book is not anymore displayed in the wishlist.
8. The use case continues at position 3 of the main flow.

Termination

This use case terminates when the user left wishlist page.

Postcondition

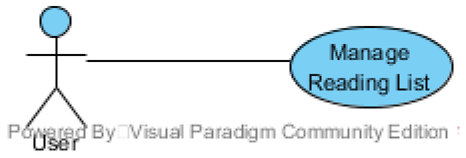
The e-commerce web application goes to wait state.

2.1.1.7 Requirement 6: Manage Reading List

2.1.1.7.1 Description & Priority

The use case describes how a user can manage a book(s) added to the reading list. Reading list represents a list of books which user has already read. The user can remove book(s) from reading list or sort books per name or author.

2.1.1.7.2 Use Case



Scope

The scope of this use case is to enable the user to manage own reading list.

Description

This use case describes steps to manage books in the reading list.

Flow Description

Precondition

The user is logged in into e-commerce web application and the user is adding book(s) to the reading list.

Activation

The user navigates to Reading list button on navigation bar.

Main flow

1. The user navigates to the navigation bar and clicks Reading list button.
2. Reading list page is displayed to the user.
3. The user checks books on the reading list. [A1 Remove book]
4. The user sorts books per author. [A2 Sort books per name]

5. The e-commerce web application displays books sorted per author.
6. The user checks one more time books in the reading list.
7. The user leaves Reading list page.

Alternate flow

(A1 Remove book)

1. The user has accidentally added a book to the reading list.
2. The user clicks Remove button on a specific book on the reading list.
3. The book is not anymore displayed on the reading list.
4. The use case continues at position 6 of the main flow.

(A2 Sort books per name)

1. The user sort books per the name of the book.
2. The e-commerce web application displays books sorted per the name of the book.
3. The use case continues at position 6 of the main flow.

Termination

This use case terminates when the user left Reading list page.

Postcondition

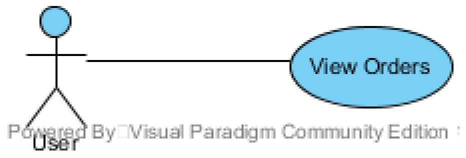
The e-commerce web application goes to wait state.

2.1.1.8 Requirement 7: View Orders

2.1.1.8.1 Description & Priority

The use case describes how a user can view orders on the own orders page. On the order list, the user can see details for a specific order. Order details are the name of the book, author of the book, quantity, price per book and total price.

2.1.1.8.2 Use Case



Scope

The scope of this use case is to enable the user to view order list on own orders page.

Description

This use case describes steps for the user to check own orders.

Flow Description

Precondition

The user already made the transaction(s).

Activation

The user activates e-commerce web application.

Main flow

1. The user navigates to the navigation bar and clicks the button to navigate to the My orders page. [A1 Log In requested]
2. The e-commerce web application allows access to the My orders page of the web application.
3. The e-commerce web application displays orders made by the user.
4. The user returns to the Homepage of e-commerce web application.

Alternate flow

(A1 Log In requested)

1. The button to navigate to the order page is not displayed because the user is not logged in into e-commerce web application.
2. The user navigates to the navigation bar and clicks Log In button.
3. The e-commerce web application returns to Log In page with log in form.
4. The user enters the log in details.
5. The user clicks on the Submit button.
6. The e-commerce web application verifies the log in details. [E1 Incorrect log in details]
7. The use case continues at position 2 of the main flow.

Exceptional flow

(E1 Incorrect log in details)

1. The e-commerce web application returns message with incorrect log in details.
2. The e-commerce web application asks from user to re-enter log in detail(s).
3. The user re-enters log in detail(s).
4. The use case continues at position 5 of the A1 alternate flow.

Termination

This use case terminates when the user left My orders page.

Postcondition

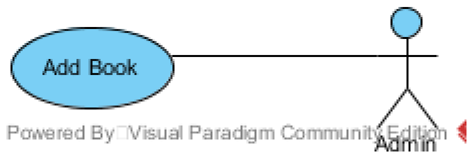
The e-commerce web application goes to wait state.

2.1.1.9 Requirement 8: Add Book

2.1.1.9.1 Description & Priority

The use case describes how a user(admin) can add a book(s) to the e-commerce web application. This requirement is necessary in order to populate the database and to display book(s) in the e-commerce web application.

2.1.1.9.2 Use Case



Scope

The scope of this use case is to enable a user(admin) to create a book(s) which will be displayed in the e-commerce web application.

Description

This use case describes steps for user(admin) to add/create a book(s) to display in the e-commerce web application.

Flow Description

Precondition

The user is logged in into e-commerce web application like admin.

Activation

The user(admin) clicks on the Manage button.

Main flow

1. The user(admin) clicks on Manage Books button to navigate to Manage Books page.
2. The e-commerce web application displays Manage Books page.
3. The user(admin) clicks on Add book button.
4. The e-commerce web application returns Add book page with a form for adding/creating a book to the e-commerce web application.

5. The user(admin) enters the book details into the form.
6. The user clicks on the Submit button.
7. The e-commerce web application verifies the book details. [E1 Incorrect book details]
8. The e-commerce web application redirects user(admin) to the All books page.

Exceptional flow

(E1 Incorrect book details)

1. The e-commerce web application returns a message with incorrect book detail(s).
2. The e-commerce web application asks user(admin) to re-enter book detail(s).
3. The user(admin) re-enters book detail(s).
4. The use case continues at position 6 of the main flow.

Termination

This use case terminates when the user(admin) returns to the All books page of e-commerce web application.

Postcondition

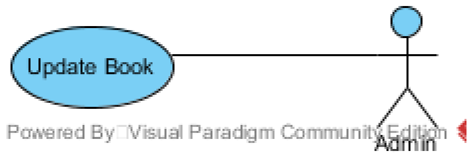
The e-commerce web application goes to wait state.

2.1.1.10 Requirement 9: Update Book

2.1.1.10.1 Description & Priority

The use case describes how a user(admin) can update(edit) book(s) in the e-commerce web application. This requirement is necessary in order to update pre-populated book detail(s) in order to display book(s) in the e-commerce web application.

2.1.1.10.2 Use Case



Scope

The scope of this use case is to enable a user(admin) to update pre-populated book detail(s).

Description

This use case describes steps for user(admin) to update book detail(s).

Flow Description

Precondition

The user is logged in into e-commerce web application like admin.

Activation

The user(admin) clicks on the Manage Books button.

Main flow

1. The user(admin) clicks on Manage Books button to navigate to Manage Books page.
2. The e-commerce web application displays Manage Books page.
3. The user(admin) clicks on the Update Book button for a specific book.

4. The e-commerce web application returns Edit page with a pre-populated form for editing book details.
5. The user(admin) changes the book detail(s).
6. The user clicks on the Submit button.
7. The e-commerce web application verifies the book details. [E1 Incorrect book details]
8. The e-commerce web application redirects user(admin) to the Manage Books page.

Exceptional flow

(E1 Incorrect book details)

1. The e-commerce web application returns a message with incorrect book detail(s).
2. The e-commerce web application asks from the user(admin) to re-enter book detail(s).
3. The user(admin) re-enters book detail(s).
4. The use case continues at position 6 of the main flow.

Termination

This use case terminates when the user(admin) returns to the Manage Books page of e-commerce web application.

Postcondition

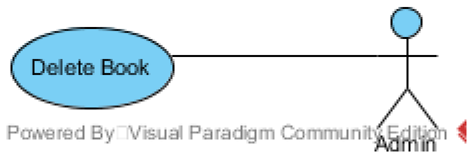
The e-commerce web application goes to wait state.

2.1.1.11 Requirement 10: Delete Book

2.1.1.11.1 Description & Priority

The use case describes how a user(admin) can delete the book(s) in the e-commerce web application. This requirement is necessary in order to delete the book(s) from the database and from the e-commerce web application.

2.1.1.11.2 Use Case



Scope

The scope of this use case is to enable a user(admin) to delete existing book(s).

Description

This use case describes steps for user(admin) to delete the book(s).

Flow Description

Precondition

The user is logged in into e-commerce web application like admin.

Activation

The user(admin) clicks on the Manage Books button.

Main flow

1. The user(admin) clicks on Manage Books button to navigate to Manage Books page.
2. The e-commerce web application displays Manage Books page.
3. The user(admin) clicks on the Delete Book button for a specific book.
4. The e-commerce web application refreshes the Manage Books page.
5. The e-commerce web application is not displaying anymore deleted book.

Termination

This use case terminates when the user(admin) returns to the Manage Books page of e-commerce web application and web application is not displaying anymore deleted book.

Postcondition

The e-commerce web application goes to wait state.

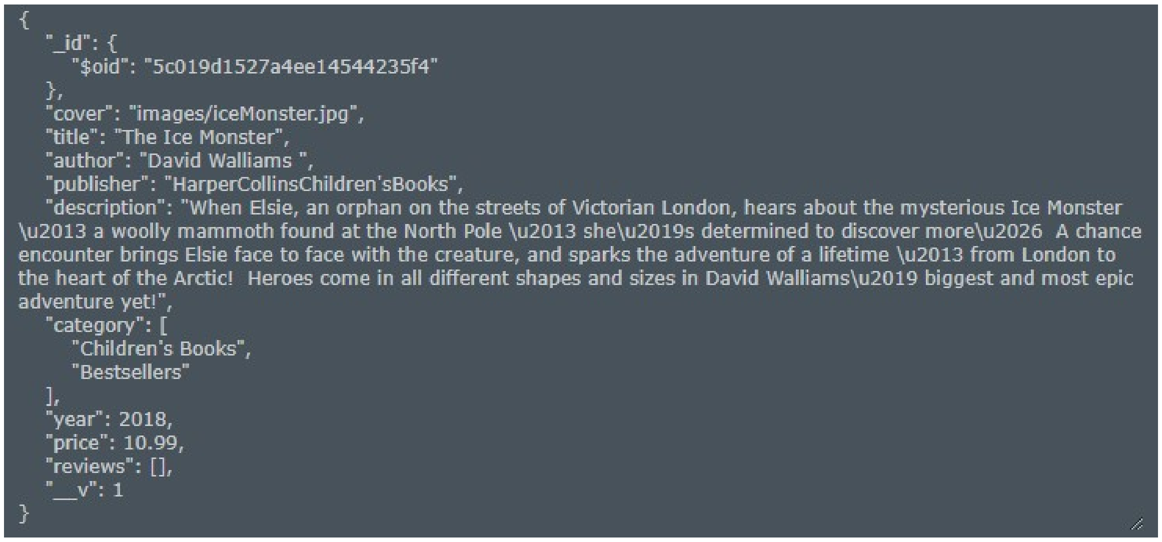
2.1.2 Data requirements

The server storing data in the MongoDB database which is hosted at mlab.com. Mongoose is the ODM (Object Document Mapper) and is used to communicate with our MongoDB database. MongoDB database has next database collections (tables).

Books	Orders	Sessions	Users
bookId	orderId	sessionId	userId
cover	userId	session	email
title	cart { items { item { All from books collection }, qty, price }, totalQty, totalPrice }	expires	password
author			confirmpassword
publisher			address
description			secondaddress
category			city
year	address		state
price	name		zip
reviews	paymentId		

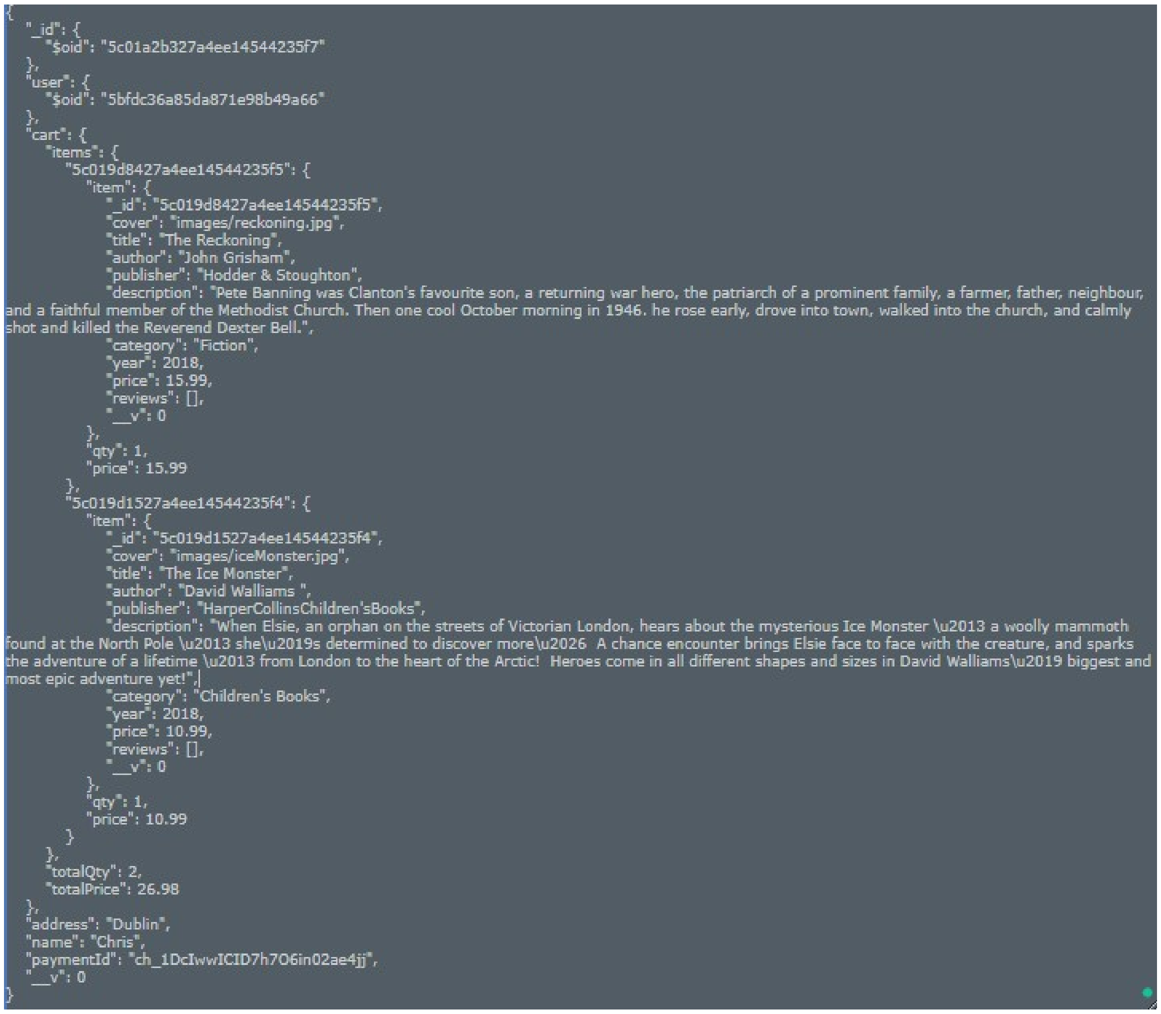
Table 1. MongoDB bookboutique database collections

Next screenshots show a record of specific collection from Table 1 to understand more about the construction of database needed for this web application.



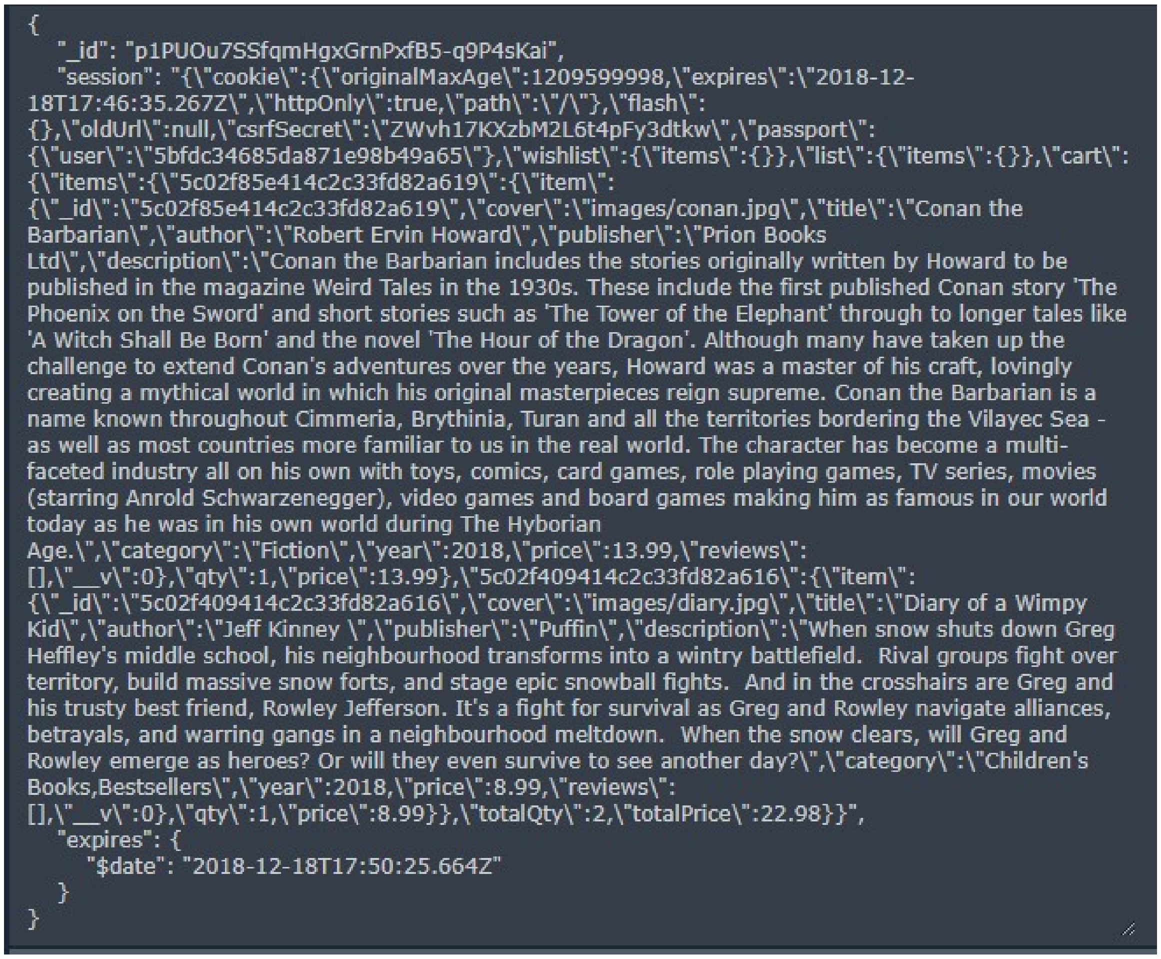
```
{
   "_id": {
      "$oid": "5c019d1527a4ee14544235f4"
   },
   "cover": "images/iceMonster.jpg",
   "title": "The Ice Monster",
   "author": "David Walliams ",
   "publisher": "HarperCollinsChildren'sBooks",
   "description": "When Elsie, an orphan on the streets of Victorian London, hears about the mysterious Ice Monster
\u2013 a woolly mammoth found at the North Pole \u2013 she\u2019s determined to discover more\u2026  A chance
encounter brings Elsie face to face with the creature, and sparks the adventure of a lifetime \u2013 from London to
the heart of the Arctic!  Heroes come in all different shapes and sizes in David Walliams\u2019 biggest and most epic
adventure yet!",
   "category": [
      "Children's Books",
      "Bestsellers"
   ],
   "year": 2018,
   "price": 10.99,
   "reviews": [],
   "__v": 1
}
```

Figure 4. Record from book collection



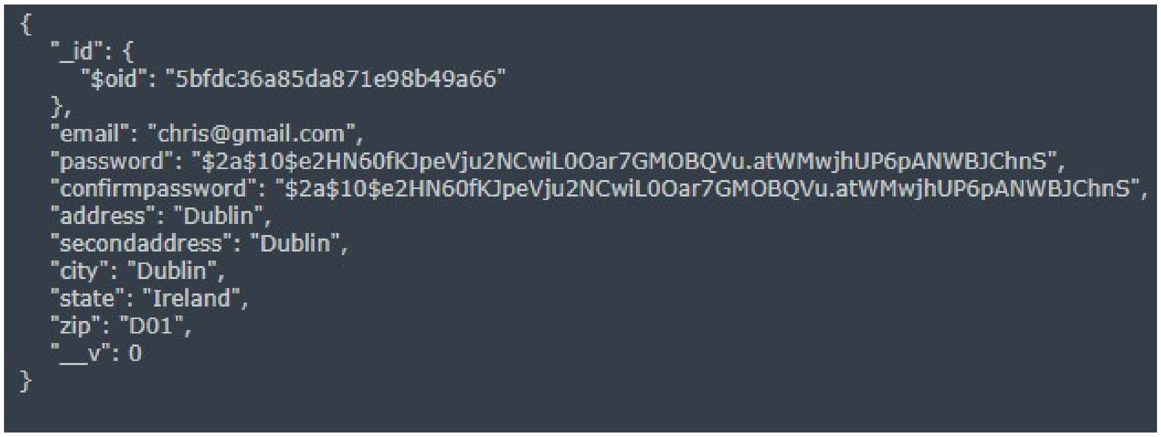
```
{
   "_id": {
      "$oid": "5c01a2b327a4ee14544235f7"
   },
   "user": {
      "$oid": "5bfdc36a85da871e98b49a66"
   },
   "cart": {
      "items": {
         "5c019d8427a4ee14544235f5": {
            "item": {
               "_id": "5c019d8427a4ee14544235f5",
               "cover": "images/reckoning.jpg",
               "title": "The Reckoning",
               "author": "John Grisham",
               "publisher": "Hodder & Stoughton",
               "description": "Pete Banning was Clanton's favourite son, a returning war hero, the patriarch of a prominent family, a farmer, father, neighbour,
and a faithful member of the Methodist Church. Then one cool October morning in 1946. he rose early, drove into town, walked into the church, and calmly
shot and killed the Reverend Dexter Bell.",
               "category": "Fiction",
               "year": 2018,
               "price": 15.99,
               "reviews": [],
               "__v": 0
            },
            "qty": 1,
            "price": 15.99
         },
         "5c019d1527a4ee14544235f4": {
            "item": {
               "_id": "5c019d1527a4ee14544235f4",
               "cover": "images/iceMonster.jpg",
               "title": "The Ice Monster",
               "author": "David Walliams ",
               "publisher": "HarperCollinsChildren'sBooks",
               "description": "When Elsie, an orphan on the streets of Victorian London, hears about the mysterious Ice Monster \u2013 a woolly mammoth
found at the North Pole \u2013 she\u2019s determined to discover more\u2026  A chance encounter brings Elsie face to face with the creature, and sparks
the adventure of a lifetime \u2013 from London to the heart of the Arctic!  Heroes come in all different shapes and sizes in David Walliams\u2019 biggest and
most epic adventure yet!",
               "category": "Children's Books",
               "year": 2018,
               "price": 10.99,
               "reviews": [],
               "__v": 0
            },
            "qty": 1,
            "price": 10.99
         }
      },
      "totalQty": 2,
      "totalPrice": 26.98
   },
   "address": "Dublin",
   "name": "Chris",
   "paymentId": "ch_1DcIwwICID7h7O6in02ae4jj",
   "__v": 0
}
```

Figure 5. Record from order collection



```
{
   "_id": "p1PUOu7SSfqmHgxGrnPxfB5-q9P4sKai",
   "session": "{\"cookie\":{\"originalMaxAge\":1209599998,\"expires\":\"2018-12-
18T17:46:35.267Z\",\"httpOnly\":true,\"path\":\"/\"},\"flash\":
{},\"oldUrl\":null,\"csrfSecret\":\"ZWvh17KXzbM2L6t4pFy3dtkw\",\"passport\":
{\"user\":\"5bfdc34685da871e98b49a65\"},\"wishlist\":{\"items\":{}},\"list\":{\"items\":{}},\"cart\":
{\"items\":{\"5c02f85e414c2c33fd82a619\":{\"item\":
{\"_id\":\"5c02f85e414c2c33fd82a619\",\"cover\":\"images/conan.jpg\",\"title\":\"Conan the
Barbarian\",\"author\":\"Robert Ervin Howard\",\"publisher\":\"Prion Books
Ltd\",\"description\":\"Conan the Barbarian includes the stories originally written by Howard to be
published in the magazine Weird Tales in the 1930s. These include the first published Conan story 'The
Phoenix on the Sword' and short stories such as 'The Tower of the Elephant' through to longer tales like
'A Witch Shall Be Born' and the novel 'The Hour of the Dragon'. Although many have taken up the
challenge to extend Conan's adventures over the years, Howard was a master of his craft, lovingly
creating a mythical world in which his original masterpieces reign supreme. Conan the Barbarian is a
name known throughout Cimmeria, Brythinia, Turan and all the territories bordering the Vilayec Sea -
as well as most countries more familiar to us in the real world. The character has become a multi-
faceted industry all on his own with toys, comics, card games, role playing games, TV series, movies
(starring Anrold Schwarzenegger), video games and board games making him as famous in our world
today as he was in his own world during The Hyborian
Age.\",\"category\":\"Fiction\",\"year\":2018,\"price\":13.99,\"reviews\":
[],\"__v\":0},\"qty\":1,\"price\":13.99},\"5c02f409414c2c33fd82a616\":{\"item\":
{\"_id\":\"5c02f409414c2c33fd82a616\",\"cover\":\"images/diary.jpg\",\"title\":\"Diary of a Wimpy
Kid\",\"author\":\"Jeff Kinney \",\"publisher\":\"Puffin\",\"description\":\"When snow shuts down Greg
Heffley's middle school, his neighbourhood transforms into a wintry battlefield.  Rival groups fight over
territory, build massive snow forts, and stage epic snowball fights.  And in the crosshairs are Greg and
his trusty best friend, Rowley Jefferson. It's a fight for survival as Greg and Rowley navigate alliances,
betrayals, and warring gangs in a neighbourhood meltdown.  When the snow clears, will Greg and
Rowley emerge as heroes? Or will they even survive to see another day?\",\"category\":\"Children's
Books,Bestsellers\",\"year\":2018,\"price\":8.99,\"reviews\":
[],\"__v\":0},\"qty\":1,\"price\":8.99}},\"totalQty\":2,\"totalPrice\":22.98}}",
   "expires": {
      "$date": "2018-12-18T17:50:25.664Z"
   }
}
```

Figure 6. Record from session collection



```
{
   "_id": {
      "$oid": "5bfdc36a85da871e98b49a66"
   },
   "email": "chris@gmail.com",
   "password": "$2a$10$e2HN60fKJpeVju2NCwiL0Oar7GMOBQVu.atWMwjhUP6pANWBJChnS",
   "confirmpassword": "$2a$10$e2HN60fKJpeVju2NCwiL0Oar7GMOBQVu.atWMwjhUP6pANWBJChnS",
   "address": "Dublin",
   "secondaddress": "Dublin",
   "city": "Dublin",
   "state": "Ireland",
   "zip": "D01",
   "__v": 0
}
```

Figure 7. Record from user collection

The e-commerce web application will use body-parser node module to read the incoming request and parse it into a format from which it can easily extract relevant information in case that needs it.

To communicate with the MongoDB database e-commerce web application will use mongoose. Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in typecasting, validation, query building, business logic hooks and more, out of the box.

2.1.3 User requirements

To access and interact with e-commerce web application user must have some essential requirements:

- Device connected to the Internet access – to interact with application user will need a device (desktop, laptop, tablet or mobile phone) connected to the internet access,
- Email account – to register/login into web application user needs to have an email account,
- Debit or credit card – to purchase the book(s) the user must have valid debit or credit card.

2.1.4 Environmental requirements

As mentioned previously, web application operates in online mode and required internet connection and hardware with an installed web browser. Any hardware which has installed web browser with relatively low bandwidth of internet connection would be good enough to run the web application effectively.

2.1.5 Usability requirements

The user interface of a web application should be easy to learn how to use and easy to remember how to use without instructions. Usability requirements for an interface design support the following from the perspective of user:

- Efficiency – user goals are easy to accomplish and with few or no user errors.
- Intuitiveness – the interface is easy to learn and navigate. Buttons, headings and success or error messages are simple to understand.
- Low perceived workload – the interface of web application appears easy to use, rather than demanding and frustrating.

2.2 Design and Architecture

Web application architecture defines the interactions between applications, middleware systems and databases to ensure multiple applications can work together. (Stringfellow, 2017) The essential purpose of a web server architecture is to complete requests made by clients for a website. The clients are typically browsers and mobile apps that make requests using secure HTTPs protocol, either for page resources or a REST API. (Svitla, 2018)

This section outlines the e-commerce web application design architecture, how the e-commerce web application interacts with other applications and how is the appropriate data correctly passed between applications.

2.2.1 Hardware Architecture

The web application normally follows three-tier web architecture consisting of the client, web server, and data source. (Mithun & D'mello, 2017). The image below gives an overview of a hardware architecture for e-commerce web application The Book Boutique.

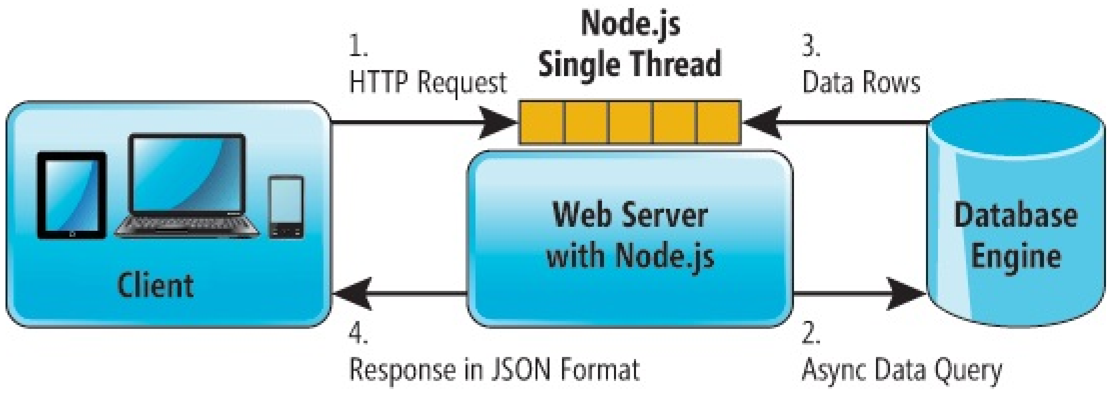


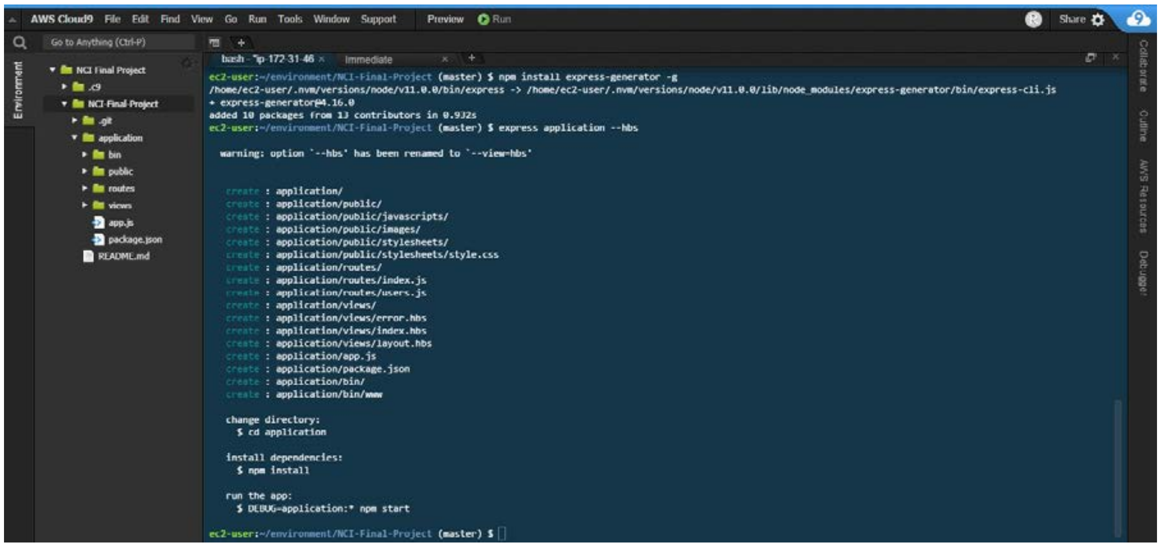
Figure 8. Hardware architecture (Vmoksha, 2017)

2.2.2 Software Architecture

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. (Eeles, 2006) This section of product design specification document gives details about software architecture for e-commerce web application The Book Boutique.

The e-commerce web application is developed using online Cloud9 IDE (Integrated Development Environment). In order to develop an e-commerce shopping site that allows fluid and fast environment of the application, the main technologies are Node.js, Express, MongoDB and Stripe.

To create an application skeleton, it is used express application generator. On image below, it is a screenshot from the initial setup of a web application with the basic structure of files and folders.



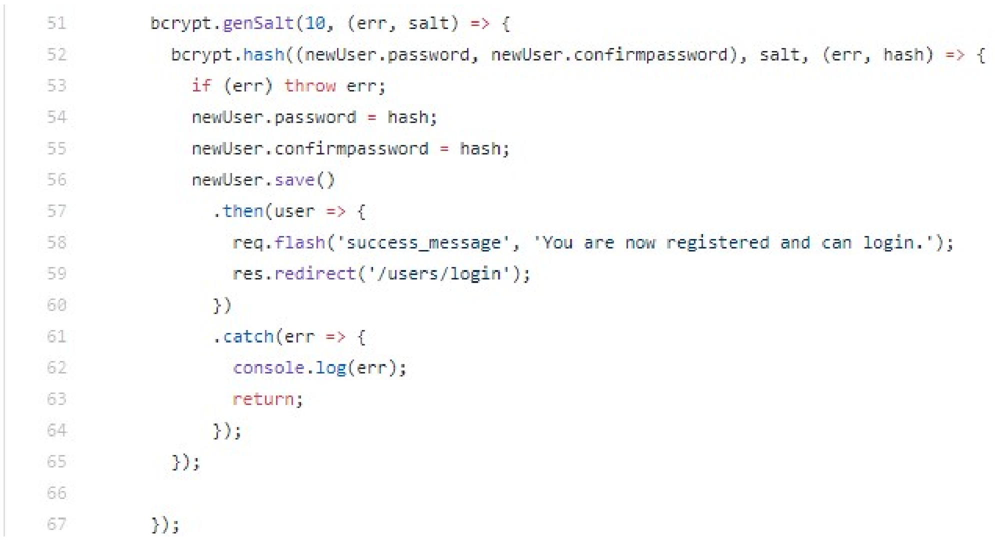
```
ec2-user:~/environment/NCI-Final-Project (master) $ npm install express-generator -g
/home/ec2-user/.nvm/versions/node/v11.0.0/bin/express -> /home/ec2-user/.nvm/versions/node/v11.0.0/lib/node_modules/express-generator/bin/express-cli.js
+ express-generator@4.16.0
added 10 packages from 13 contributors in 0.932s
ec2-user:~/environment/NCI-Final-Project (master) $ express application --hbs

  warning: option `--hbs' has been renamed to `--view=hbs'


   create : application/
   create : application/public/
   create : application/public/javascripts/
   create : application/public/images/
   create : application/public/stylesheets/
   create : application/public/stylesheets/style.css
   create : application/routes/
   create : application/routes/index.js
   create : application/routes/users.js
   create : application/views/
   create : application/views/error.hbs
   create : application/views/index.hbs
   create : application/views/layout.hbs
   create : application/app.js
   create : application/package.json
   create : application/bin/
   create : application/bin/www

   change directory:
     $ cd application

   install dependencies:
     $ npm install

   run the app:
     $ DEBUG=application:* npm start

ec2-user:~/environment/NCI-Final-Project (master) $
```

Figure 9. Initial setup with express generator

2.2.3 Security Architecture

Security architecture refers to a plan and set of principles that describe the security services that a system is required to provide to meet the needs of its users, the system elements required to provide to implement the services, and also the performance levels required in the elements to deal with the threat environment. (Beal, 2018)

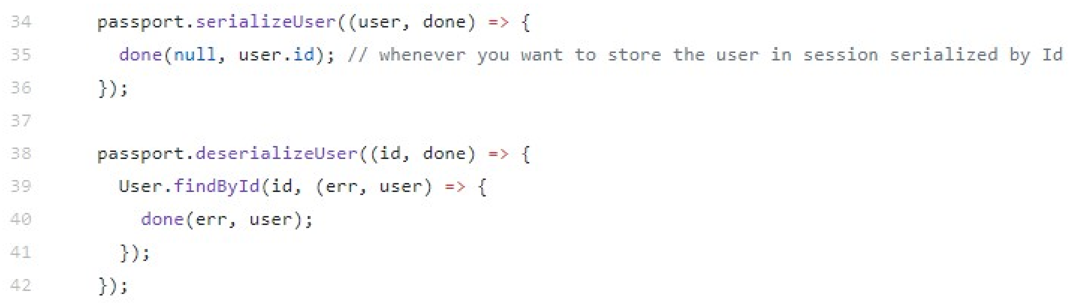
The user needs to be logged in into the application to access specific functionalities of the e-commerce web application. Sensible data about the user are protected in the way that e-commerce web application uses the bcryptjs node module for data encryption and Passport.js for user authentication. On the next image, it is shown main code which allows data (password) encryption. Passport and bcrypt code is implemented using video tutorial delivery by (Traversy, 2018).



```
bcrypt.genSalt(10, (err, salt) => {
  bcrypt.hash((newUser.password, newUser.confirmpassword), salt, (err, hash) => {
    if (err) throw err;
    newUser.password = hash;
    newUser.confirmpassword = hash;
    newUser.save()
      .then(user => {
        req.flash('success_message', 'You are now registered and can login.');
        res.redirect('/users/login');
      })
      .catch(err => {
        console.log(err);
        return;
      });
  });

});
```

Figure 10. Bcryptjs function for data encryption

The user needs to provide login details to verify credentials. If the user is verified, the user will be serialized into the session and logged in to the application.



```
passport.serializeUser((user, done) => {
  done(null, user.id); // whenever you want to store the user in session serialized by Id
});

passport.deserializeUser((id, done) => {
  User.findById(id, (err, user) => {
    done(err, user);
  });
});
```

Figure 11. Passport function to serialize user

On the image, below is explained the Passport.js authentication flow in an e-commerce web application.

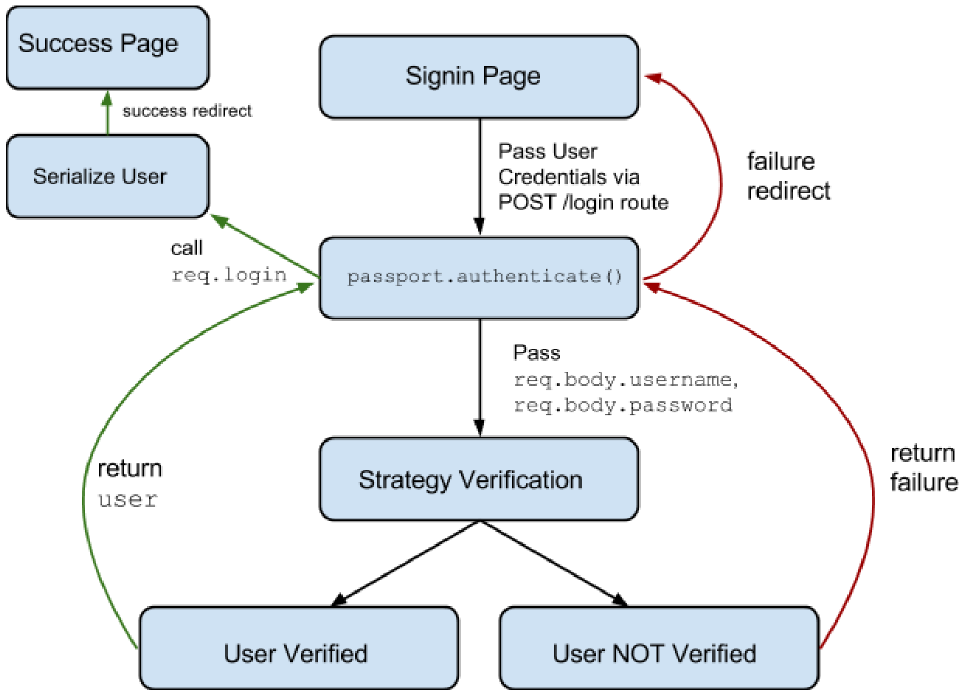
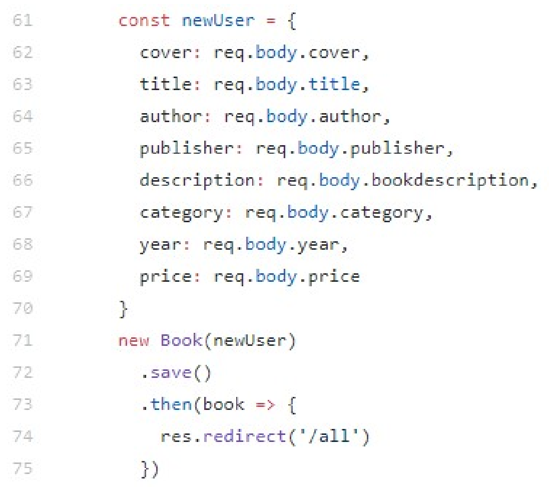


Figure 12. Passport authentication flow (Arneson, 2014)

2.2.4 Communication Architecture

Web-based communication is critical for web application architecture since the majority of global network traffic, and every single application and device uses that kind of communication. Web-based communication deals with scale, efficiency, robustness and security. Including the server and the client side, two programs running concurrently:

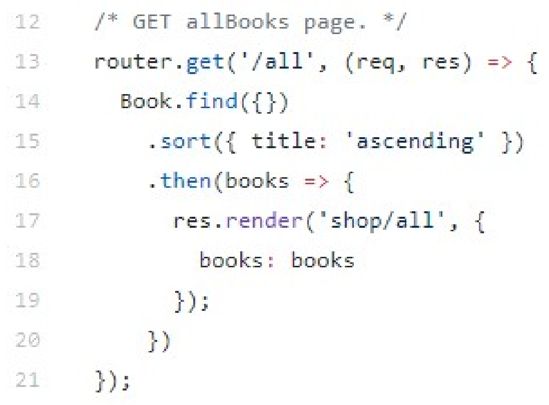
- the code which lives in the browser and responds to user input and



```
const newUser = {
  cover: req.body.cover,
  title: req.body.title,
  author: req.body.author,
  publisher: req.body.publisher,
  description: req.body.bookdescription,
  category: req.body.category,
  year: req.body.year,
  price: req.body.price
}
new Book(newUser)
  .save()
  .then(book => {
    res.redirect('/all')
  })
```

Figure 13. Saving data to MongoDB

- the code which lives on the server and responds to HTTP requests.



```
/* GET allBooks page. */
router.get('/all', (req, res) => {
  Book.find({})
    .sort({ title: 'ascending' })
    .then(books => {
      res.render('shop/all', {
        books: books
      });
    })
});
```

Figure 14. Fetching data from MongoDB

The image gives a general overview of communication architecture in the e-commerce web application. The image doesn't include communication with Stripe platform.

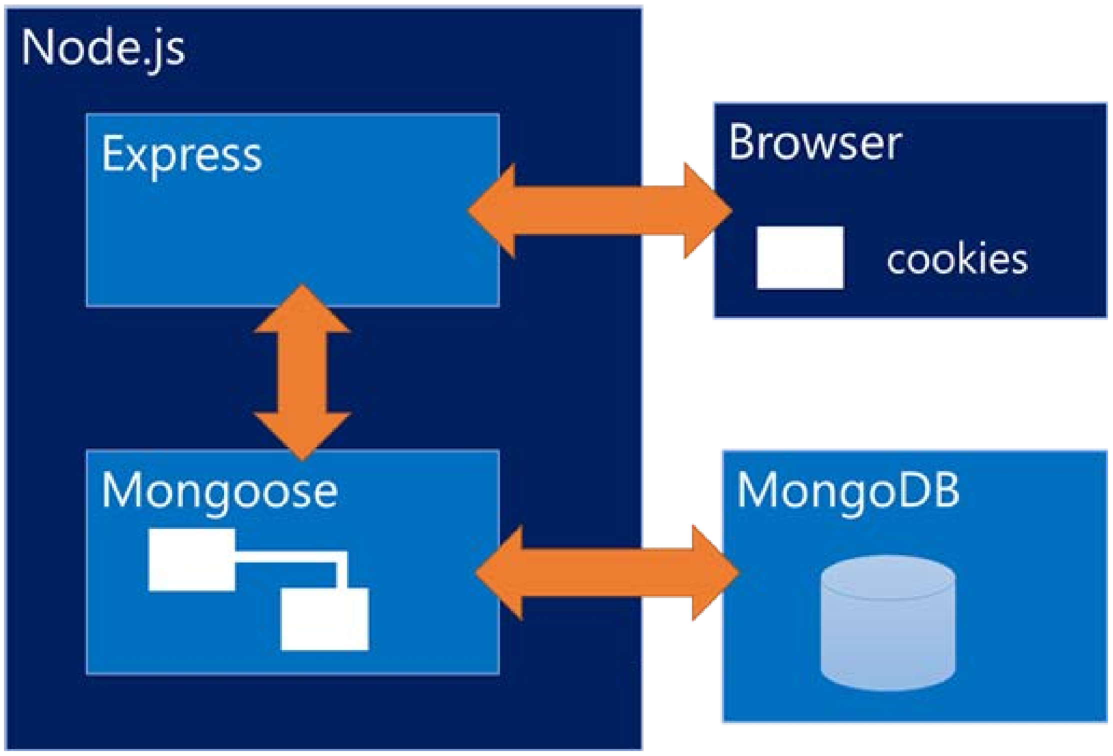
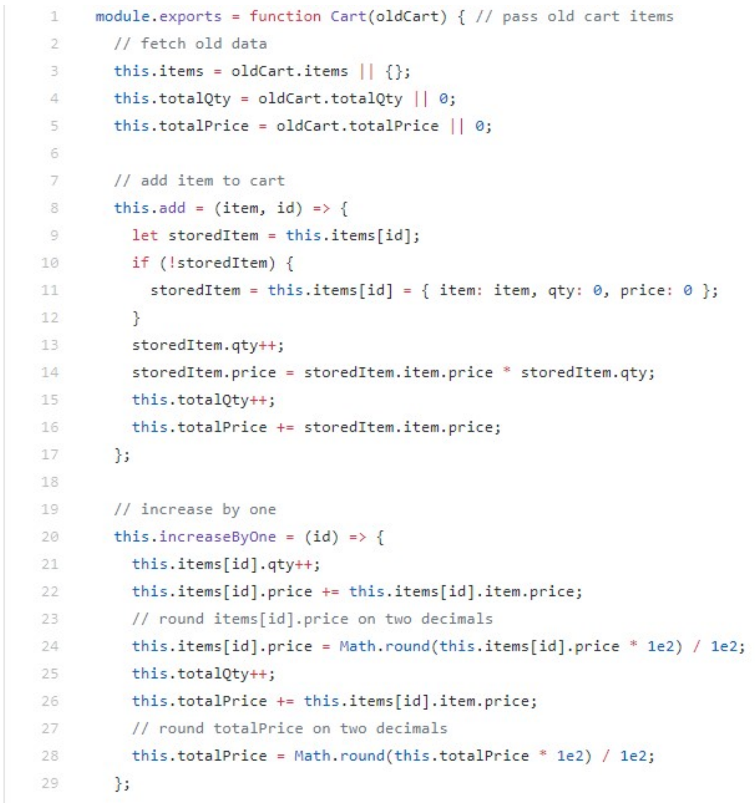


Figure 15. Communication Architecture (Terkaly, 2014)

2.3 Implementation

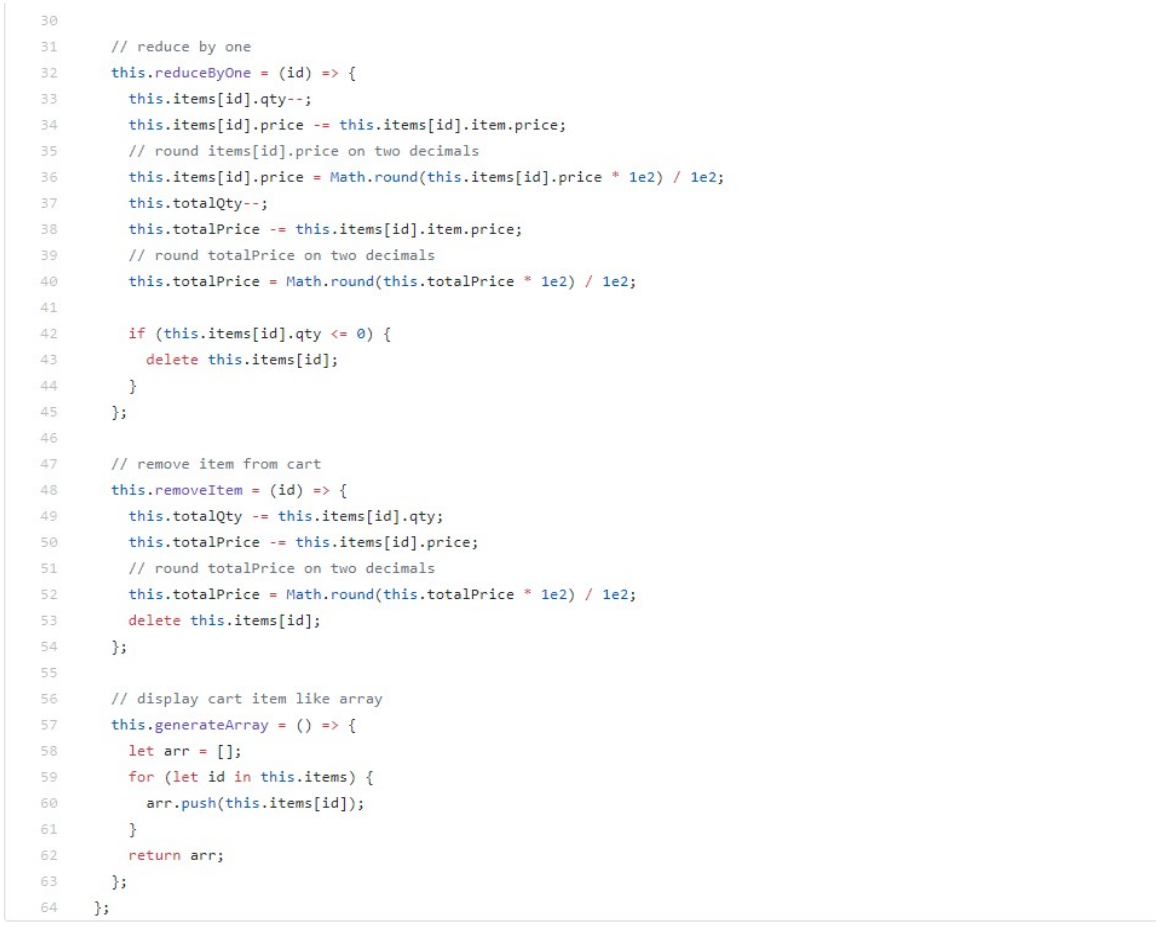
The Book Boutique is an e-commerce web application so because of that, the main functions used in project code are cart functions and checkout functions. The cart and checkout functions are inspired by (Academind, 2016).

2.3.1 Cart model



```
module.exports = function Cart(oldCart) { // pass old cart items
  // fetch old data
  this.items = oldCart.items || {};
  this.totalQty = oldCart.totalQty || 0;
  this.totalPrice = oldCart.totalPrice || 0;

  // add item to cart
  this.add = (item, id) => {
    let storedItem = this.items[id];
    if (!storedItem) {
      storedItem = this.items[id] = { item: item, qty: 0, price: 0 };
    }
    storedItem.qty++;
    storedItem.price = storedItem.item.price * storedItem.qty;
    this.totalQty++;
    this.totalPrice += storedItem.item.price;
  };

  // increase by one
  this.increaseByOne = (id) => {
    this.items[id].qty++;
    this.items[id].price += this.items[id].item.price;
    // round items[id].price on two decimals
    this.items[id].price = Math.round(this.items[id].price * 1e2) / 1e2;
    this.totalQty++;
    this.totalPrice += this.items[id].item.price;
    // round totalPrice on two decimals
    this.totalPrice = Math.round(this.totalPrice * 1e2) / 1e2;
  };
```

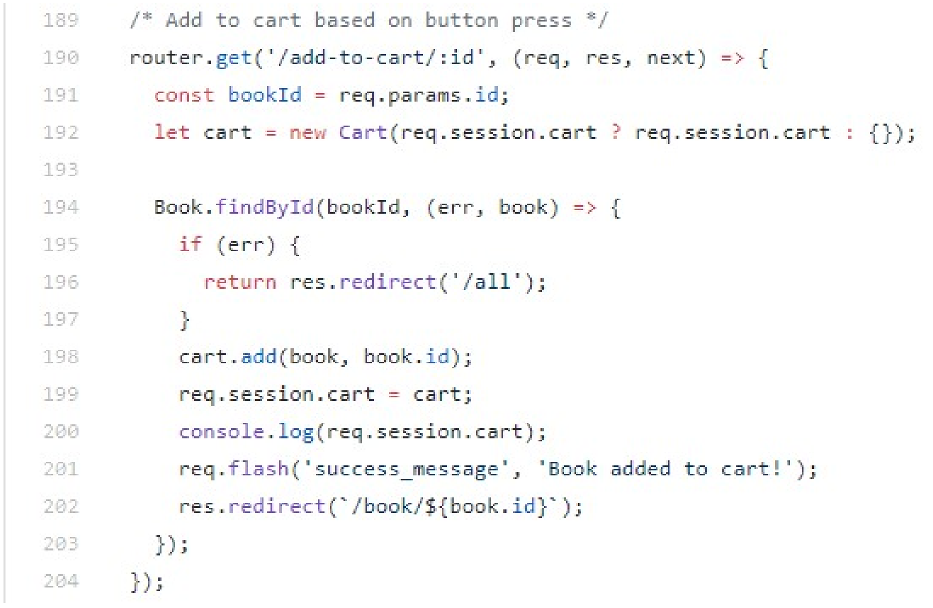
Figure 16. Cart model – part 1



```

  // reduce by one
  this.reduceByOne = (id) => {
    this.items[id].qty--;
    this.items[id].price -= this.items[id].item.price;
    // round items[id].price on two decimals
    this.items[id].price = Math.round(this.items[id].price * 1e2) / 1e2;
    this.totalQty--;
    this.totalPrice -= this.items[id].item.price;
    // round totalPrice on two decimals
    this.totalPrice = Math.round(this.totalPrice * 1e2) / 1e2;

    if (this.items[id].qty <= 0) {
      delete this.items[id];
    }
  };

  // remove item from cart
  this.removeItem = (id) => {
    this.totalQty -= this.items[id].qty;
    this.totalPrice -= this.items[id].price;
    // round totalPrice on two decimals
    this.totalPrice = Math.round(this.totalPrice * 1e2) / 1e2;
    delete this.items[id];
  };

  // display cart item like array
  this.generateArray = () => {
    let arr = [];
    for (let id in this.items) {
      arr.push(this.items[id]);
    }
    return arr;
  };
};
```

Figure 17. Cart model - part 2

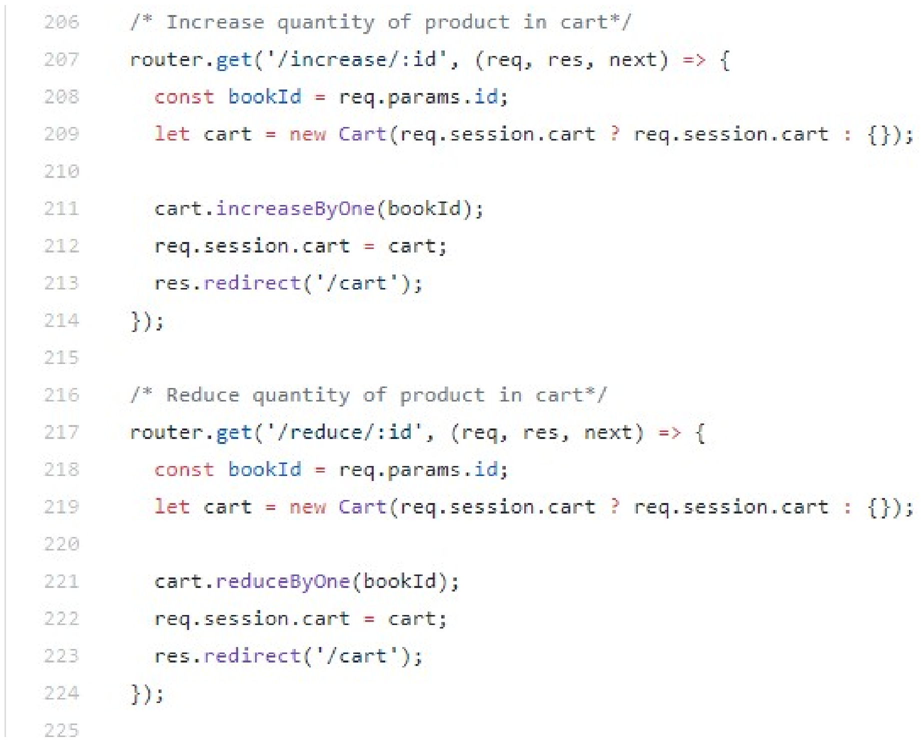
2.3.2 Add to cart



```
/* Add to cart based on button press */
router.get('/add-to-cart/:id', (req, res, next) => {
  const bookId = req.params.id;
  let cart = new Cart(req.session.cart ? req.session.cart : {});

  Book.findById(bookId, (err, book) => {
    if (err) {
      return res.redirect('/all');
    }
    cart.add(book, book.id);
    req.session.cart = cart;
    console.log(req.session.cart);
    req.flash('success_message', 'Book added to cart!');
    res.redirect(`/book/${book.id}`);
  });
});
```

Figure 18. Add to cart route

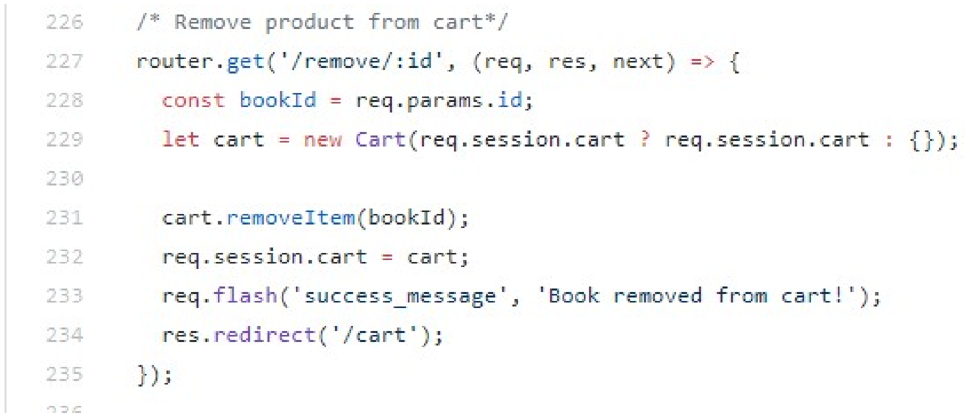
2.3.3 Increase and reduce quantity of product in cart



```
/* Increase quantity of product in cart*/
router.get('/increase/:id', (req, res, next) => {
  const bookId = req.params.id;
  let cart = new Cart(req.session.cart ? req.session.cart : {});

  cart.increaseByOne(bookId);
  req.session.cart = cart;
  res.redirect('/cart');
});

/* Reduce quantity of product in cart*/
router.get('/reduce/:id', (req, res, next) => {
  const bookId = req.params.id;
  let cart = new Cart(req.session.cart ? req.session.cart : {});

  cart.reduceByOne(bookId);
  req.session.cart = cart;
  res.redirect('/cart');
});
```

Figure 19. Increase and reduce quantity in cart

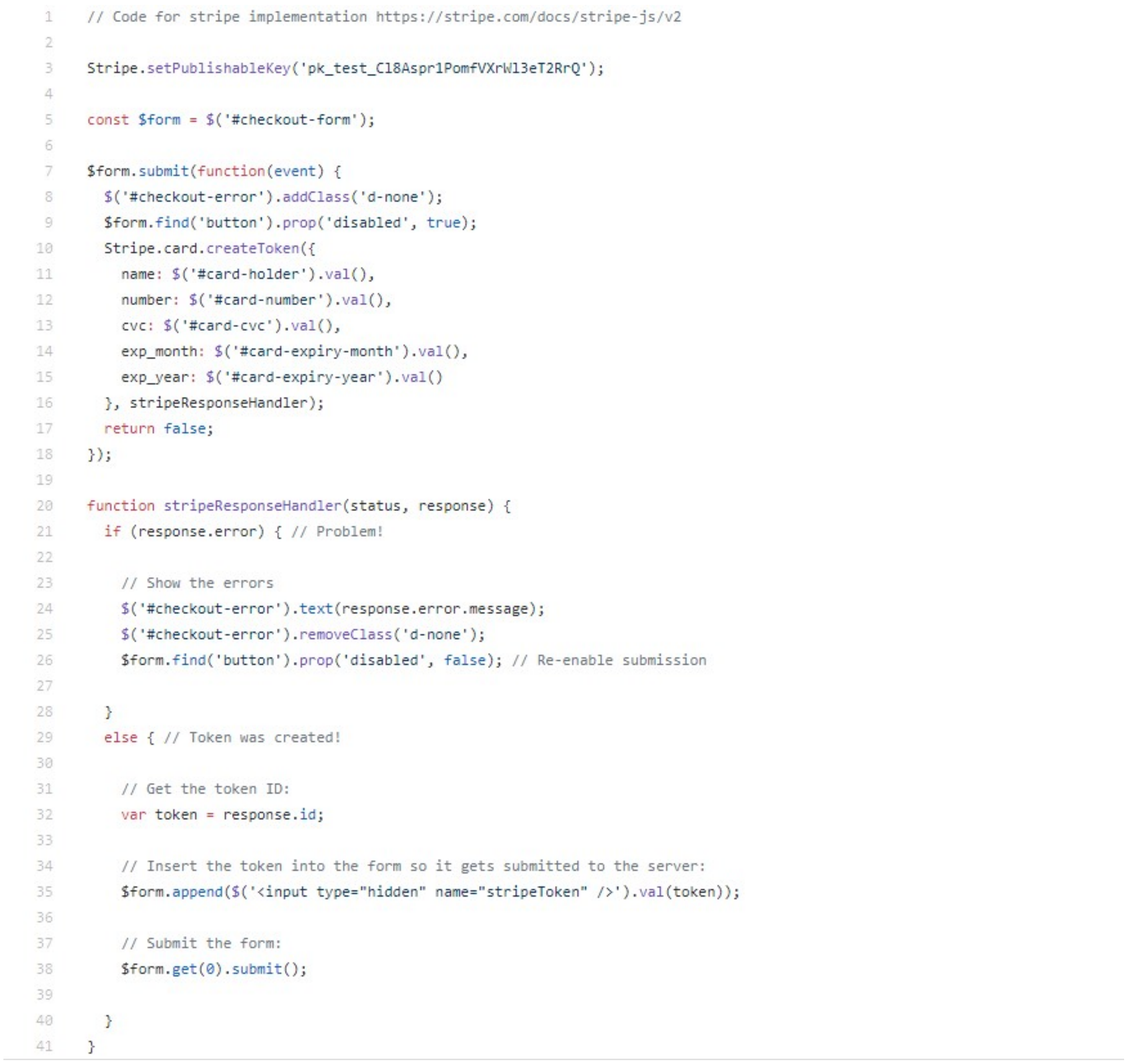
2.3.4 Remove product from cart



```
/* Remove product from cart*/
router.get('/remove/:id', (req, res, next) => {
  const bookId = req.params.id;
  let cart = new Cart(req.session.cart ? req.session.cart : {});

  cart.removeItem(bookId);
  req.session.cart = cart;
  req.flash('success_message', 'Book removed from cart!');
  res.redirect('/cart');
});
```

Figure 20. Remove product from cart

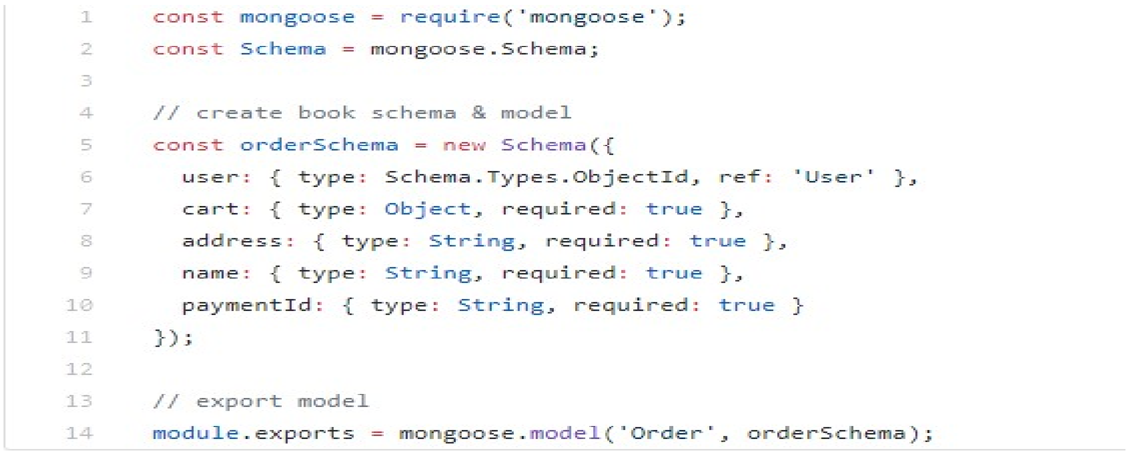
2.3.5 Verifying credit cards using stripe payment process



```javascript
// Code for stripe implementation https://stripe.com/docs/stripe-js/v2

Stripe.setPublishableKey('pk_test_Cl8Aspr1PomfVXrWl3eT2RrQ');

const $form = $('#checkout-form');

$form.submit(function(event) {
  $('#checkout-error').addClass('d-none');
  $form.find('button').prop('disabled', true);
  Stripe.card.createToken({
    name: $('#card-holder').val(),
    number: $('#card-number').val(),
    cvc: $('#card-cvc').val(),
    exp_month: $('#card-expiry-month').val(),
    exp_year: $('#card-expiry-year').val()
  }, stripeResponseHandler);
  return false;
});

function stripeResponseHandler(status, response) {
  if (response.error) { // Problem!

    // Show the errors
    $('#checkout-error').text(response.error.message);
    $('#checkout-error').removeClass('d-none');
    $form.find('button').prop('disabled', false); // Re-enable submission

  }
  else { // Token was created!

    // Get the token ID:
    var token = response.id;

    // Insert the token into the form so it gets submitted to the server:
    $form.append($('<input type="hidden" name="stripeToken" />').val(token));

    // Submit the form:
    $form.get(0).submit();

  }
}
```

Figure 21. Verify credit cards using stripe

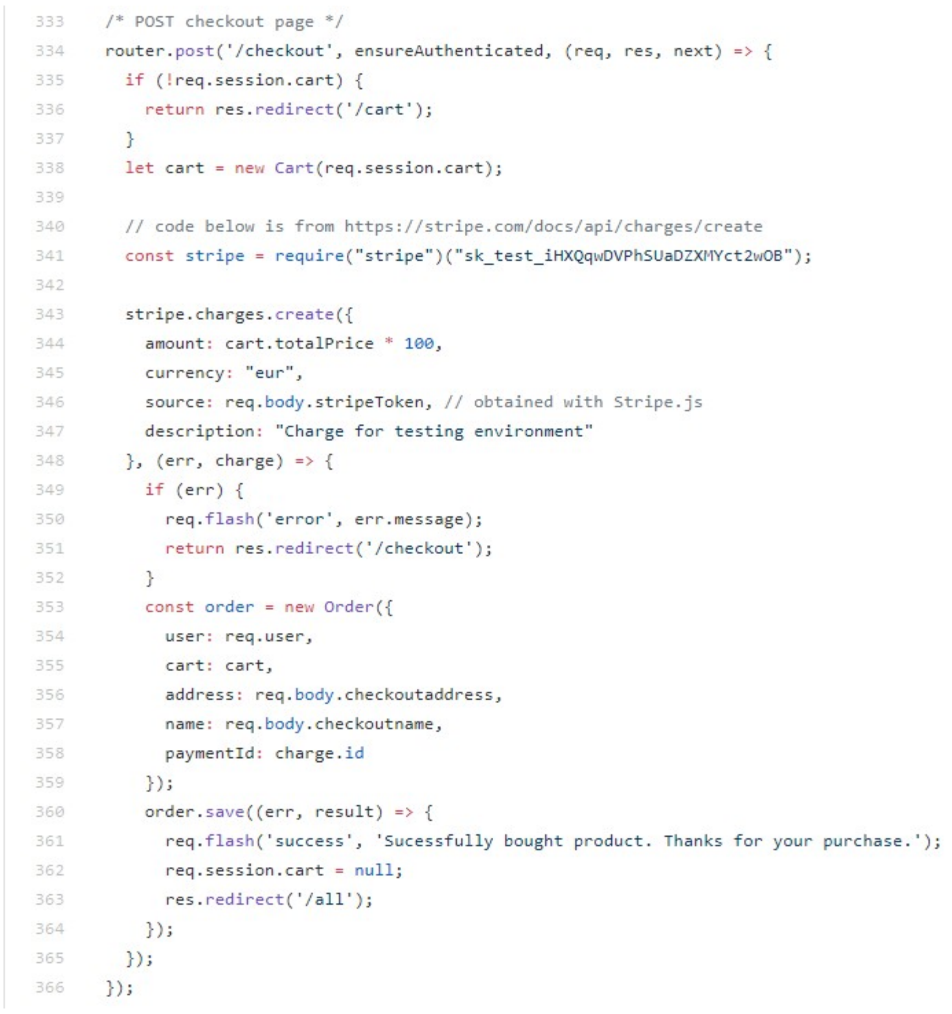
2.3.6 Order model



```javascript
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

// create book schema & model
const orderSchema = new Schema({
  user: { type: Schema.Types.ObjectId, ref: 'User' },
  cart: { type: Object, required: true },
  address: { type: String, required: true },
  name: { type: String, required: true },
  paymentId: { type: String, required: true }
});

// export model
module.exports = mongoose.model('Order', orderSchema);
```

Figure 22. Order model

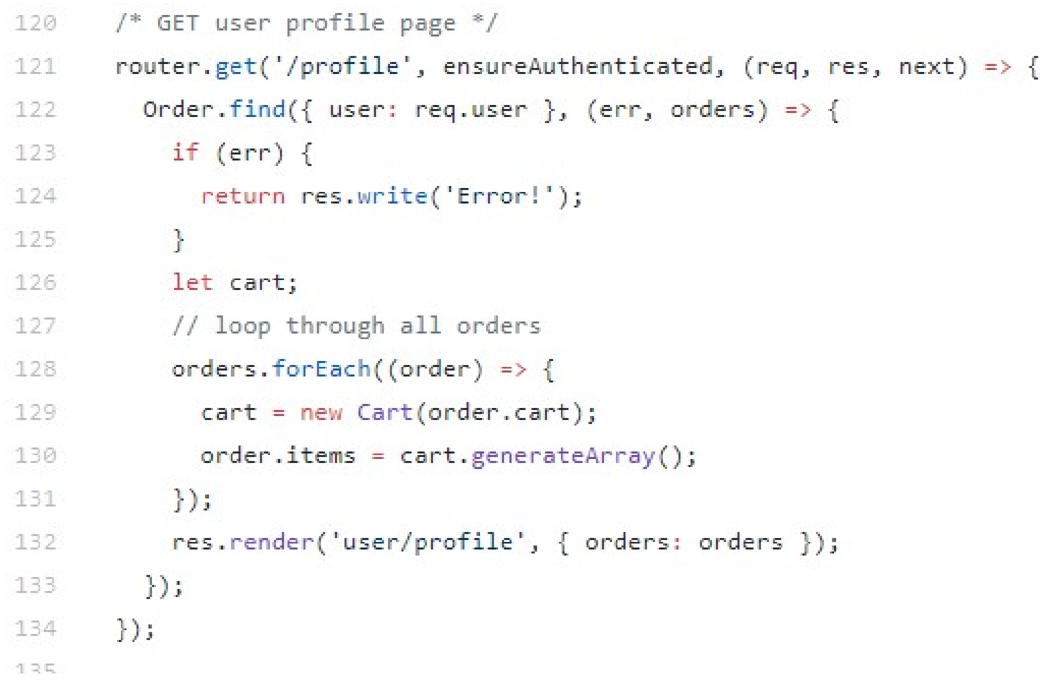
2.3.7 POST Checkout page and storing order



```javascript
/* POST checkout page */
router.post('/checkout', ensureAuthenticated, (req, res, next) => {
  if (!req.session.cart) {
    return res.redirect('/cart');
  }
  let cart = new Cart(req.session.cart);

  // code below is from https://stripe.com/docs/api/charges/create
  const stripe = require("stripe")("sk_test_iHXQqwDVPhSUaDZXMYct2wOB");

  stripe.charges.create({
    amount: cart.totalPrice * 100,
    currency: "eur",
    source: req.body.stripeToken, // obtained with Stripe.js
    description: "Charge for testing environment"
  }, (err, charge) => {
    if (err) {
      req.flash('error', err.message);
      return res.redirect('/checkout');
    }
    const order = new Order({
      user: req.user,
      cart: cart,
      address: req.body.checkoutaddress,
      name: req.body.checkoutname,
      paymentId: charge.id
    });
    order.save((err, result) => {
      req.flash('success', 'Sucessfully bought product. Thanks for your purchase.');
      req.session.cart = null;
      res.redirect('/all');
    });
  });
});
```

Figure 23. POST Checkout page with storing order in database

2.3.8 Display order

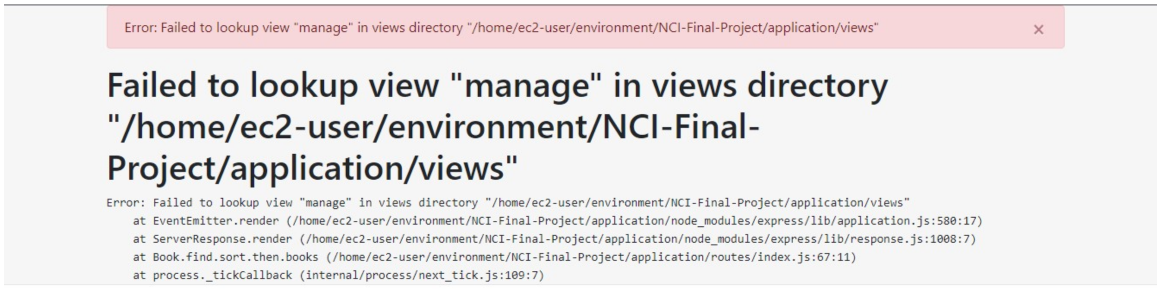


```
/* GET user profile page */
router.get('/profile', ensureAuthenticated, (req, res, next) => {
  Order.find({ user: req.user }, (err, orders) => {
    if (err) {
      return res.write('Error!');
    }
    let cart;
    // loop through all orders
    orders.forEach((order) => {
      cart = new Cart(order.cart);
      order.items = cart.generateArray();
    });
    res.render('user/profile', { orders: orders });
  });
});
```

Figure 24. Display order on user profile

2.4 Testing

Every time when a new code of line was added and it could be tested in the browser the function was tested. Testing was done continuously for Handlebars and main functionality of web application. The screenshot (Figure 25) was recorded during development of “Manage Books” page and represent the broken route of the web application.



Error: Failed to lookup view "manage" in views directory "/home/ec2-user/environment/NCI-Final-Project/application/views"

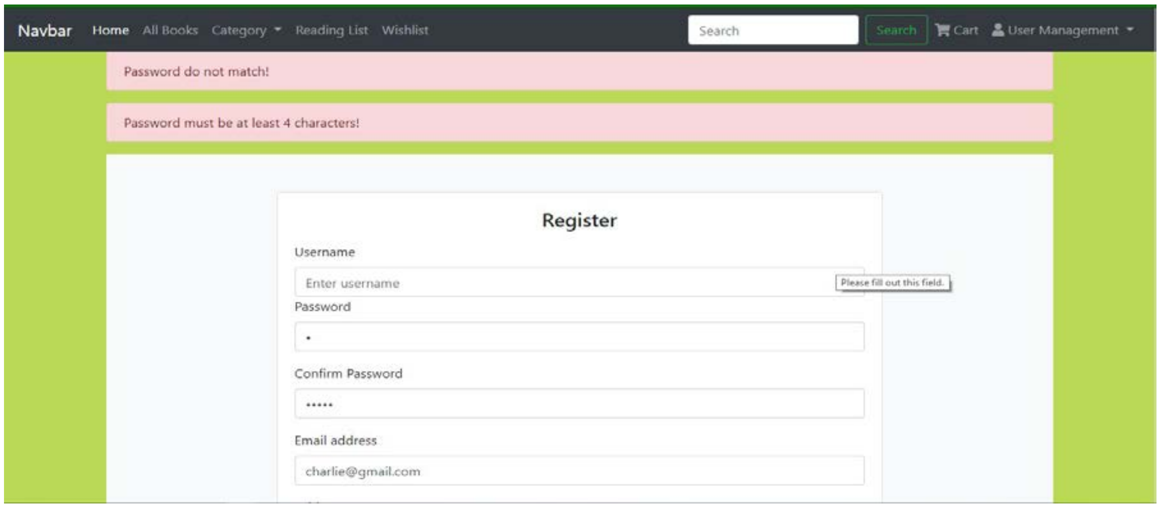
Failed to lookup view "manage" in views directory "/home/ec2-user/environment/NCI-Final-Project/application/views"

```
Error: Failed to lookup view "manage" in views directory "/home/ec2-user/environment/NCI-Final-Project/application/views"
    at EventEmitter.render (/home/ec2-user/environment/NCI-Final-Project/application/node_modules/express/lib/application.js:580:17)
    at ServerResponse.render (/home/ec2-user/environment/NCI-Final-Project/application/node_modules/express/lib/response.js:1008:7)
    at Book.find.sort.then.books (/home/ec2-user/environment/NCI-Final-Project/application/routes/index.js:67:11)
    at process._tickCallback (internal/process/next_tick.js:109:7)
```

Figure 25. Route error

The broken route is immediately fixed added the correct name of the file which exists in views directory to route code for this page.

A lot of testing is done with forms especially for register and login users. On image below is password testing for input field during the registration process.



Password do not match!

Password must be at least 4 characters!

Register

Username: Enter username

Please fill out this field.

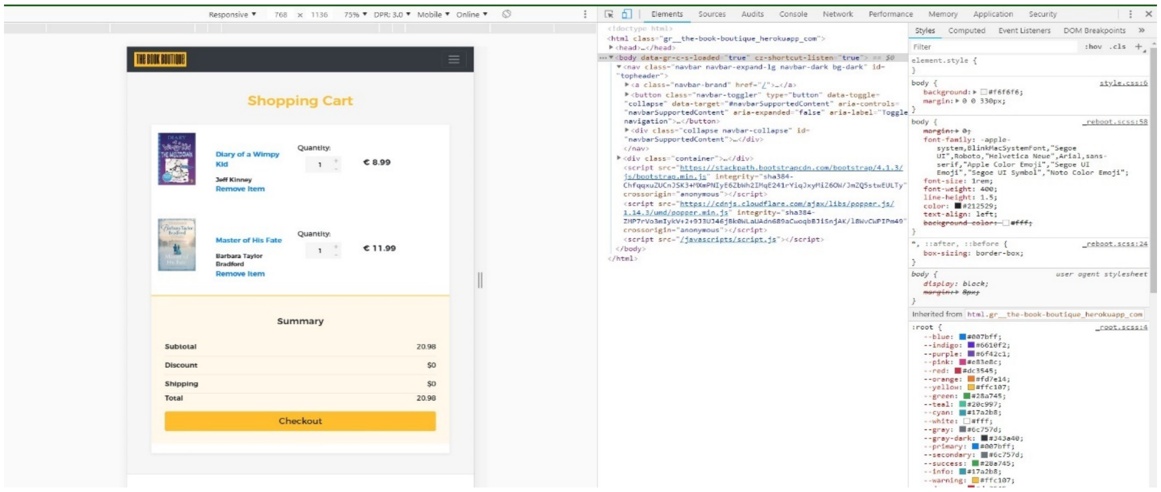
Password: •

Confirm Password: •••••

Email address: charlie@gmail.com

Figure 26. Password test

The last testing needed to be done was responsive web design testing. CSS was implemented to fit every screen resolution and that was implemented with help of Google Chrome developer tools (Figure 26).


Responsive 768 × 1136 75% DPR: 3.0 Mobile Online
Elements Sources Audits Console Network Performance Memory Application Security
Styles Computed Event Listeners DOM Breakpoints
Filter :hov .cls +
THE BOOK BOUTIQUE
Shopping Cart
Checkout

Diary of a Wimpy Kid
Jeff Kinney
Remove Item
Quantity: 1
€ 8.99

Master of His Fate
Barbara Taylor Bradford
Remove Item
Quantity: 1
€ 11.99

Summary

Subtotal	20.98
Discount	$0
Shipping	$0
Total	20.98

```
<!doctype html>
<html class="gr__the-book-boutique_herokuapp_com">
 ▶<head>…</head>
··· ▼<body data-gr-c-s-loaded="true" cz-shortcut-listen="true"> == $0
   ▼<nav class="navbar navbar-expand-lg navbar-dark bg-dark" id=
   "topheader">
     ▶<a class="navbar-brand" href="/">…</a>
     ▶<button class="navbar-toggler" type="button" data-toggle=
     "collapse" data-target="#navbarSupportedContent" aria-controls=
     "navbarSupportedContent" aria-expanded="false" aria-label="Toggle
     navigation">…</button>
     ▶<div class="collapse navbar-collapse" id=
     "navbarSupportedContent">…</div>
    </nav>
   ▶<div class="container">…</div>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/
    js/bootstrap.min.js" integrity="sha384-
    ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
    crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/
    1.14.3/umd/popper.min.js" integrity="sha384-
    ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
    crossorigin="anonymous"></script>
    <script src="/javascripts/script.js"></script>
  </body>
</html>
```

```
element.style {
}
body {                                        style.css:6
   background: ▶ #f6f6f6;
   margin: ▶ 0 0 330px;
}
body {                                     _reboot.scss:58
   ~~margin: ▶ 0;~~
   font-family: -apple-
      system,BlinkMacSystemFont,"Segoe
      UI",Roboto,"Helvetica Neue",Arial,sans-
      serif,"Apple Color Emoji","Segoe UI
      Emoji","Segoe UI Symbol","Noto Color Emoji";
   font-size: 1rem;
   font-weight: 400;
   line-height: 1.5;
   color: #212529;
   text-align: left;
   ~~background-color: #fff;~~
}
*, ::after, ::before {                     _reboot.scss:24
   box-sizing: border-box;
}
body {                              user agent stylesheet
   display: block;
   ~~margin: ▶ 8px;~~
}
Inherited from html.gr__the-book-boutique_herokuapp_com
:root {                                      _root.scss:4
   --blue: #007bff;
   --indigo: #6610f2;
   --purple: #6f42c1;
   --pink: #e83e8c;
   --red: #dc3545;
   --orange: #fd7e14;
   --yellow: #ffc107;
   --green: #28a745;
   --teal: #20c997;
   --cyan: #17a2b8;
   --white: #fff;
   --gray: #6c757d;
   --gray-dark: #343a40;
   --primary: #007bff;
   --secondary: #6c757d;
   --success: #28a745;
   --info: #17a2b8;
   --warning: #ffc107;
```

Figure 27. Google Chrome developer tools

2.5 Graphical User Interface (GUI) Layout

A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers, hand-held devices and other appliances. This interface uses icons, menus and another visual indicator (graphics) representations to display information and related user controls, unlike text-based interfaces, where data and commands are in the text. GUI representations are manipulated by a pointing device such as a mouse, trackball, stylus, or a finger on a touch screen. (Techopedia, 2018)

The following screenshots represented how is graphical user interface implemented for this e-commerce web application.

2.5.1 Homepage

On the top of the web application is displayed a menu with buttons necessary for user interaction with the web application. Homepage displaying bestselling books and giving a short overview of the web application.

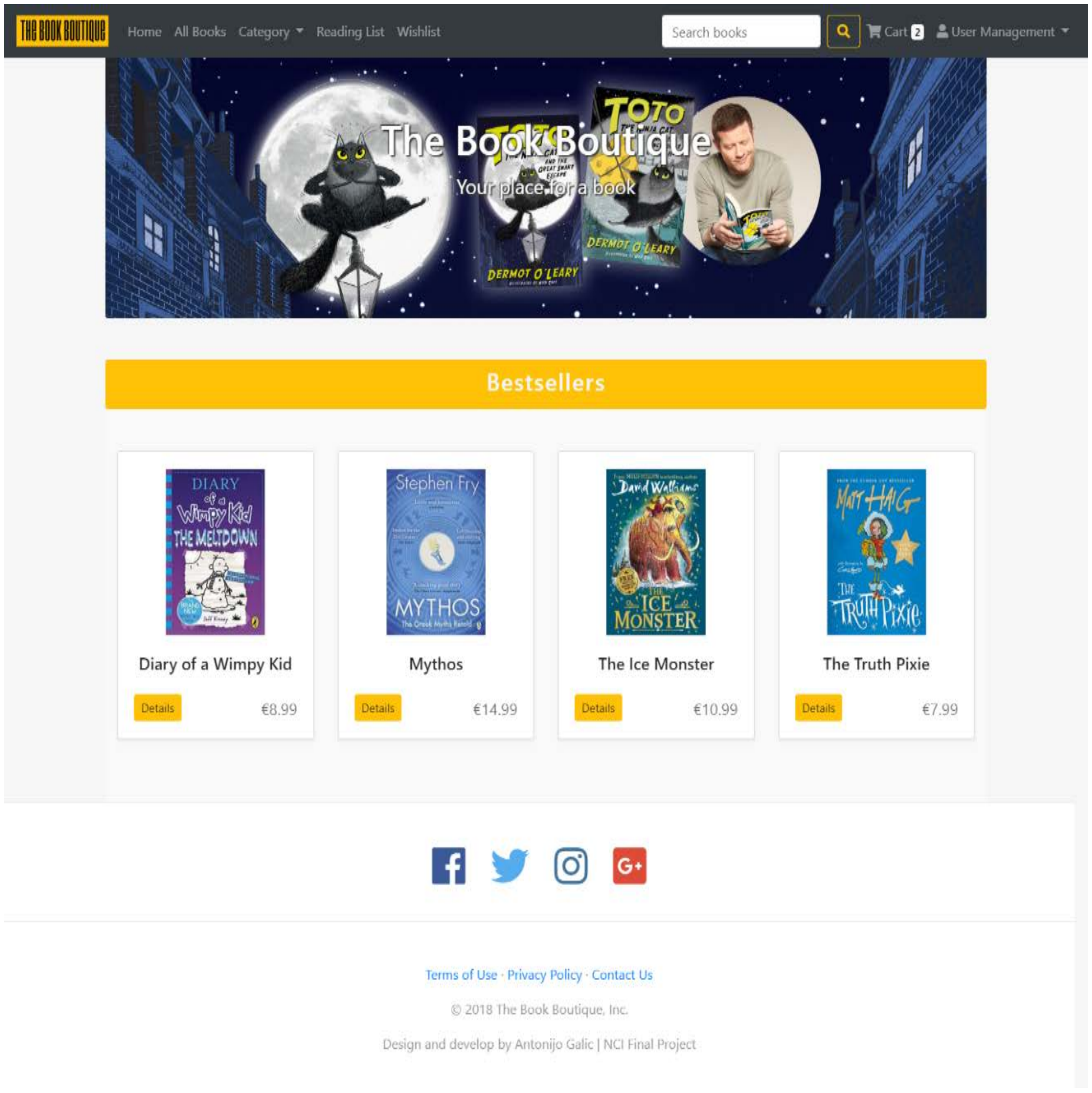


Figure 28. Homepage

2.5.2 All books

This page displaying all books in shop sorted ascending. Each book displays the price of the book and shows a button with a value of “Details” to access detail page of the specific book.

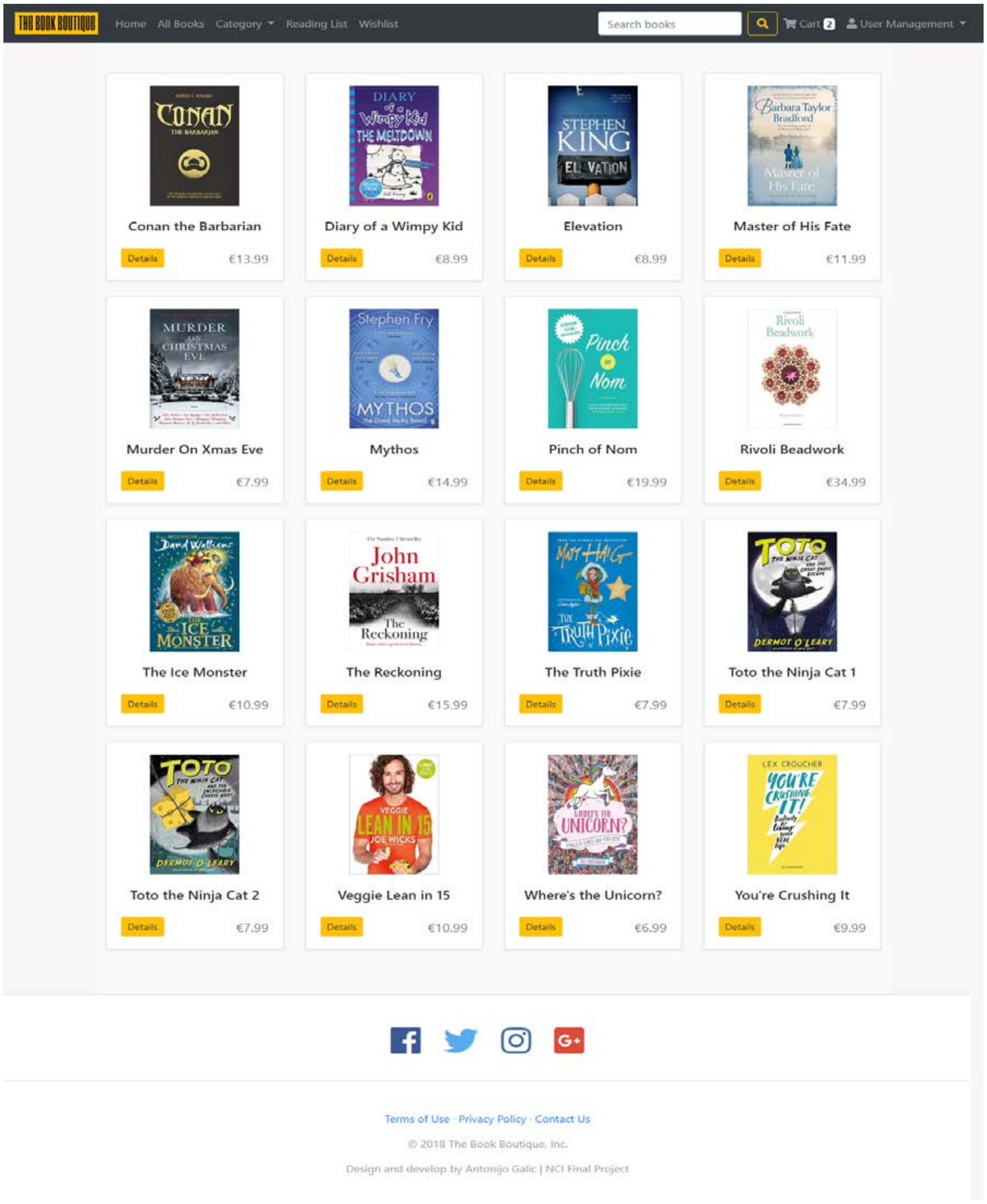
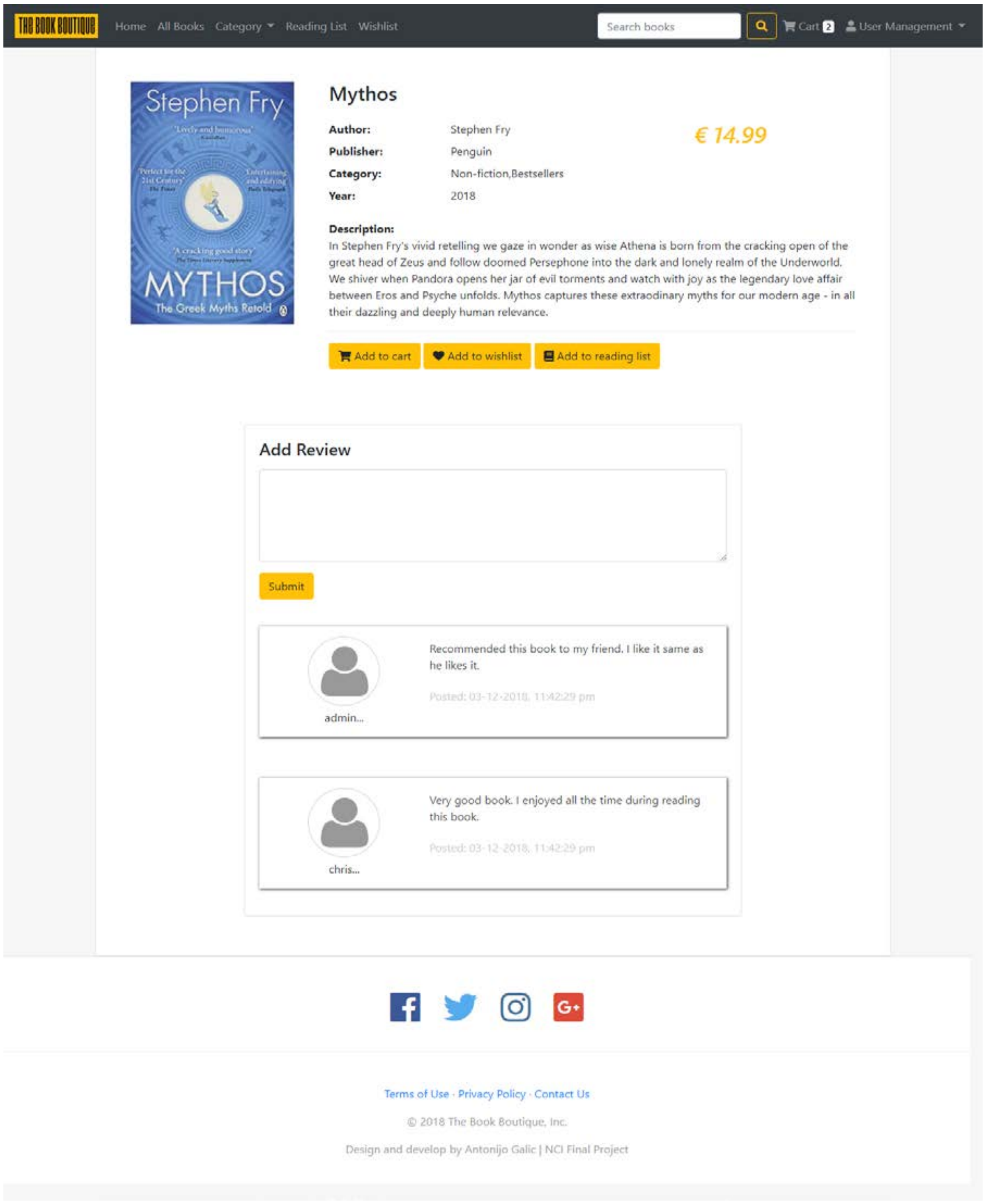


Figure 29. Displayed all books

2.5.3 Book detail page

Book detail page gives a description of the book, book reviews and displaying buttons for interaction with the user.



Mythos

Author: Stephen Fry

Publisher: Penguin

Category: Non-fiction,Bestsellers

Year: 2018

€ 14.99

Description:
In Stephen Fry's vivid retelling we gaze in wonder as wise Athena is born from the cracking open of the great head of Zeus and follow doomed Persephone into the dark and lonely realm of the Underworld. We shiver when Pandora opens her jar of evil torments and watch with joy as the legendary love affair between Eros and Psyche unfolds. Mythos captures these extraodinary myths for our modern age - in all their dazzling and deeply human relevance.

Add Review

admin...
Recommended this book to my friend. I like it same as he likes it.
Posted: 03-12-2018, 11:42:29 pm

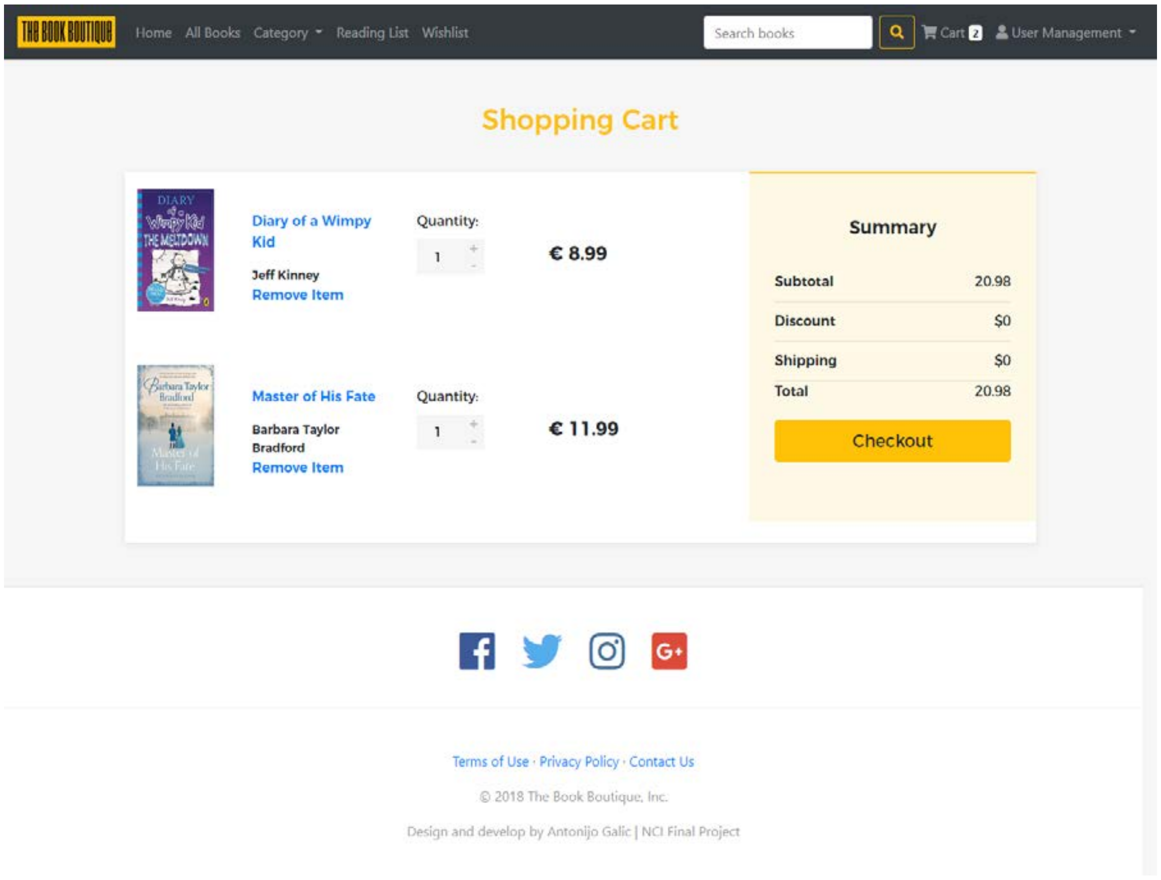
chris...
Very good book. I enjoyed all the time during reading this book.
Posted: 03-12-2018, 11:42:29 pm

Terms of Use · Privacy Policy · Contact Us
© 2018 The Book Boutique, Inc.
Design and develop by Antonijo Galic | NCI Final Project

Figure 30. Book detail page

2.5.4 Cart

Cart page displays books added to cart with quantity, price and total price.



Diary of a Wimpy Kid
Jeff Kinney
Quantity: 1
€ 8.99

Master of His Fate
Barbara Taylor Bradford
Quantity: 1
€ 11.99

Summary

Subtotal	20.98
Discount	$0
Shipping	$0
Total	20.98

Terms of Use · Privacy Policy · Contact Us

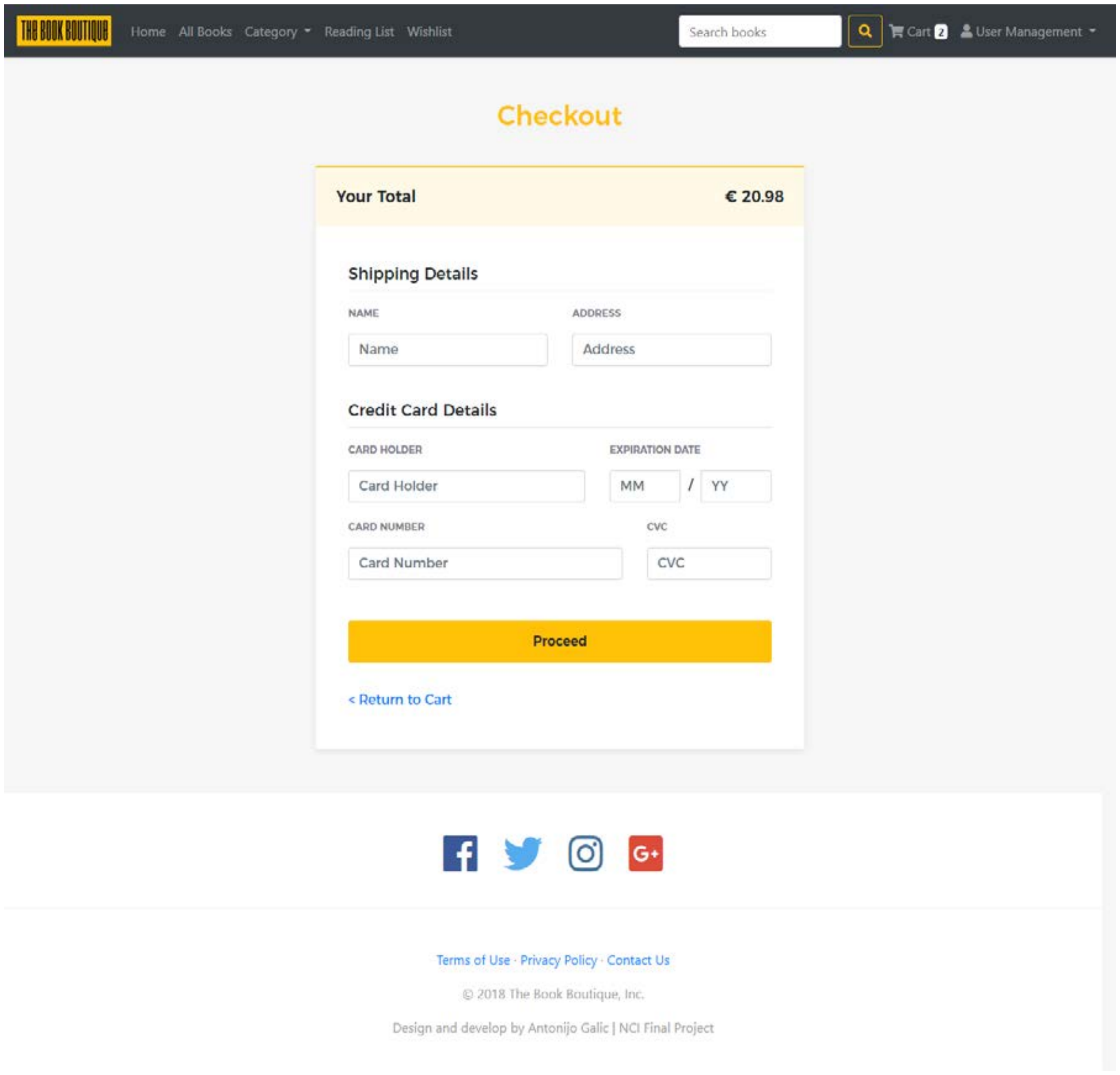
© 2018 The Book Boutique, Inc.

Design and develop by Antonijo Galic | NCI Final Project

Figure 31. Cart page

2.5.5 Checkout

On cart page is a button which will take the user to the Checkout page where the user needs to fill up form with card details.



Checkout

Your Total € 20.98

Shipping Details

NAME: Name

ADDRESS: Address

Credit Card Details

CARD HOLDER: Card Holder

EXPIRATION DATE: MM / YY

CARD NUMBER: Card Number

CVC: CVC

Proceed

< Return to Cart

Terms of Use · Privacy Policy · Contact Us

© 2018 The Book Boutique, Inc.

Design and develop by Antonijo Galic | NCI Final Project

Figure 32. Checkout page

2.5.6 Wishlist

Wishlist page displaying books added on the wish list from a user. Each book has the option to add a book to the cart or remove the book from wishlist. With a click on book name, user can see more details about the specific book.

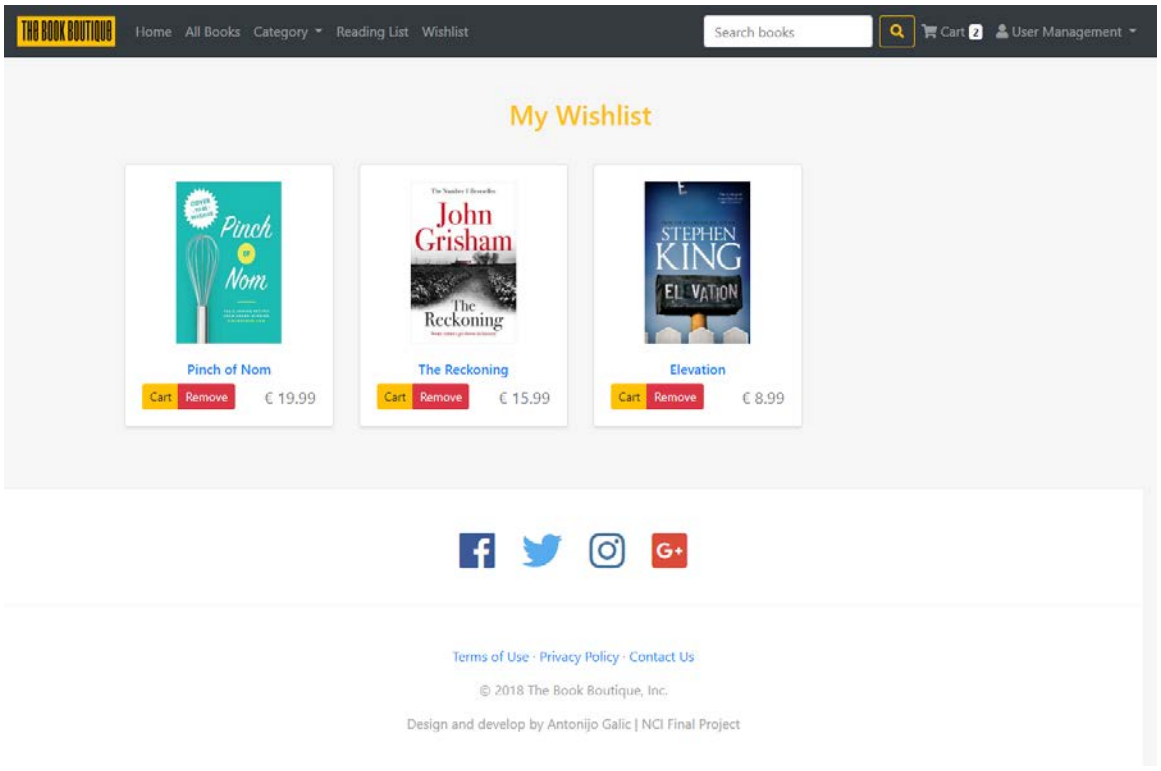
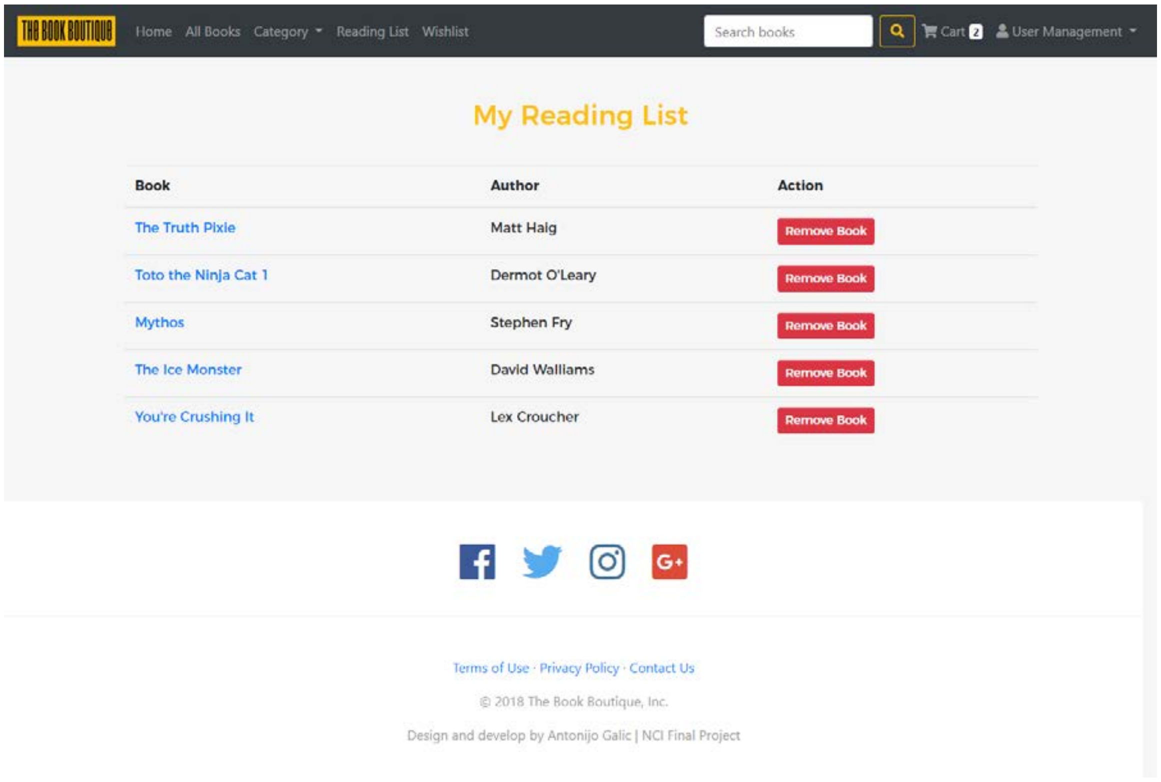


Figure 33. Wishlist page

2.5.7 Reading list

A reading list displaying books added from the user in the table view. Reading list shows the books which user already read and can track own reading history. Boks in the reading list can be sorted per book name or per author.



My Reading List

Book	Author	Action
The Truth Pixie	Matt Haig	Remove Book
Toto the Ninja Cat 1	Dermot O'Leary	Remove Book
Mythos	Stephen Fry	Remove Book
The Ice Monster	David Walliams	Remove Book
You're Crushing It	Lex Croucher	Remove Book

Terms of Use · Privacy Policy · Contact Us

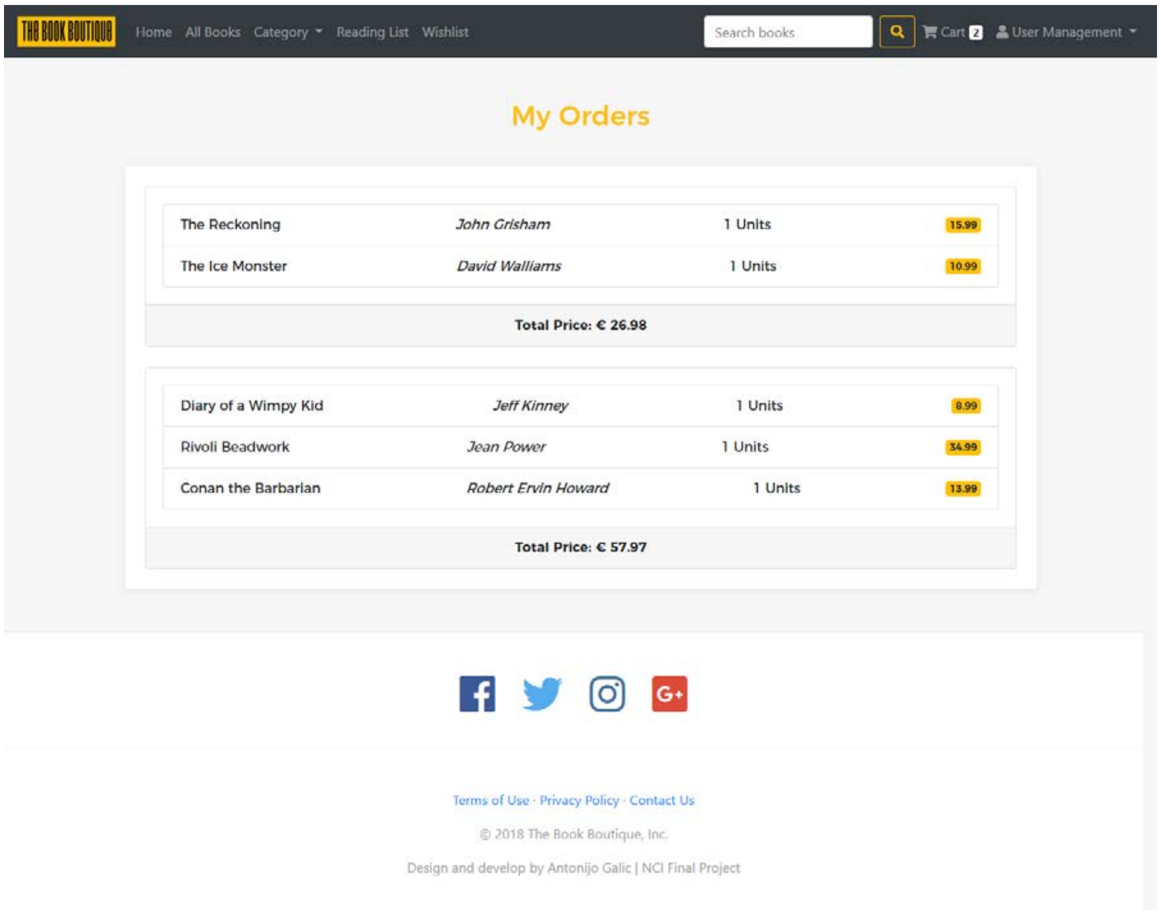
© 2018 The Book Boutique, Inc.

Design and develop by Antonijo Galic | NCI Final Project

Figure 34. Reading list page

2.5.8 My orders

This page displaying all orders from a specific user with name and author of the book, quantity, price and total price.



My Orders

The Reckoning	*John Grisham*	1 Units	15.99
The Ice Monster	*David Walliams*	1 Units	10.99

Total Price: € 26.98

Diary of a Wimpy Kid	*Jeff Kinney*	1 Units	8.99
Rivoli Beadwork	*Jean Power*	1 Units	34.99
Conan the Barbarian	*Robert Ervin Howard*	1 Units	13.99

Total Price: € 57.97

Terms of Use · Privacy Policy · Contact Us

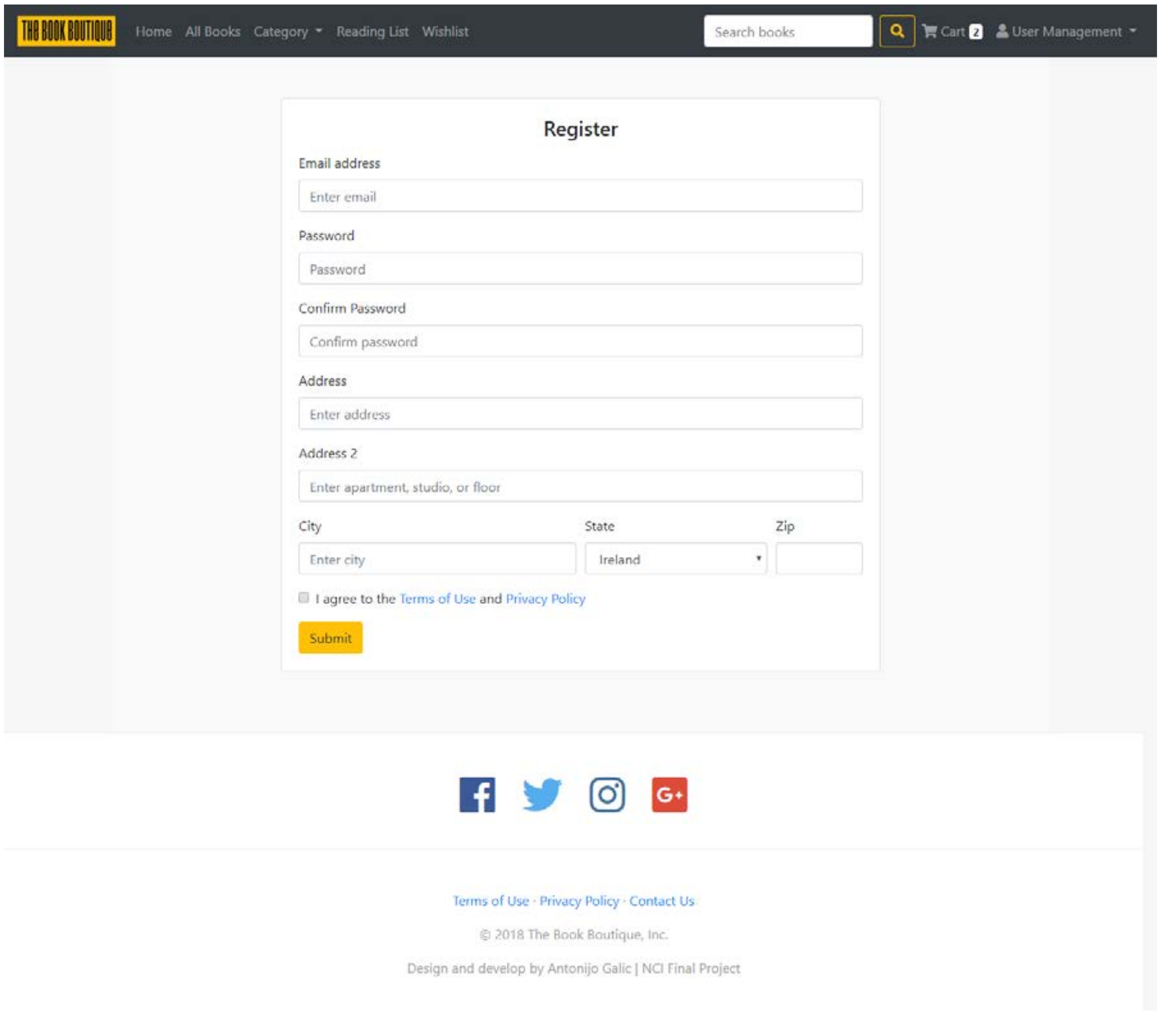
© 2018 The Book Boutique, Inc.

Design and develop by Antonijo Galic | NCI Final Project

Figure 35. Order history page

2.5.9 Register / Login

Register and login page have input form and will required text input from the user. Register and Login page are also validated and in case of error or success shows message depends on the situation.


THE BOOK BOUTIQUE Home All Books Category Reading List Wishlist Search books Cart 2 User Management

Register

Email address

Enter email

Password

Password

Confirm Password

Confirm password

Address

Enter address

Address 2

Enter apartment, studio, or floor

City	State	Zip
Enter city	Ireland	

I agree to the Terms of Use and Privacy Policy

Submit

Terms of Use · Privacy Policy · Contact Us

© 2018 The Book Boutique, Inc.

Design and develop by Antonijo Galic | NCI Final Project

Figure 36. Register page

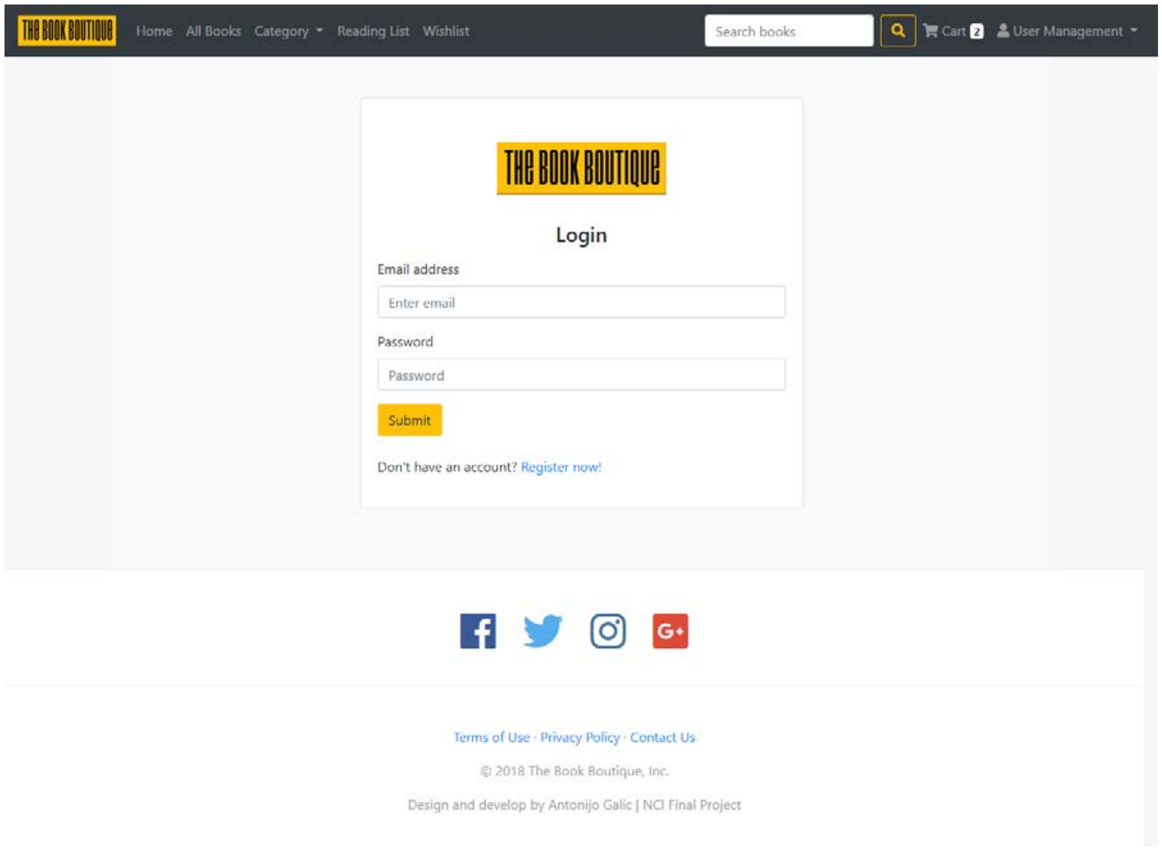
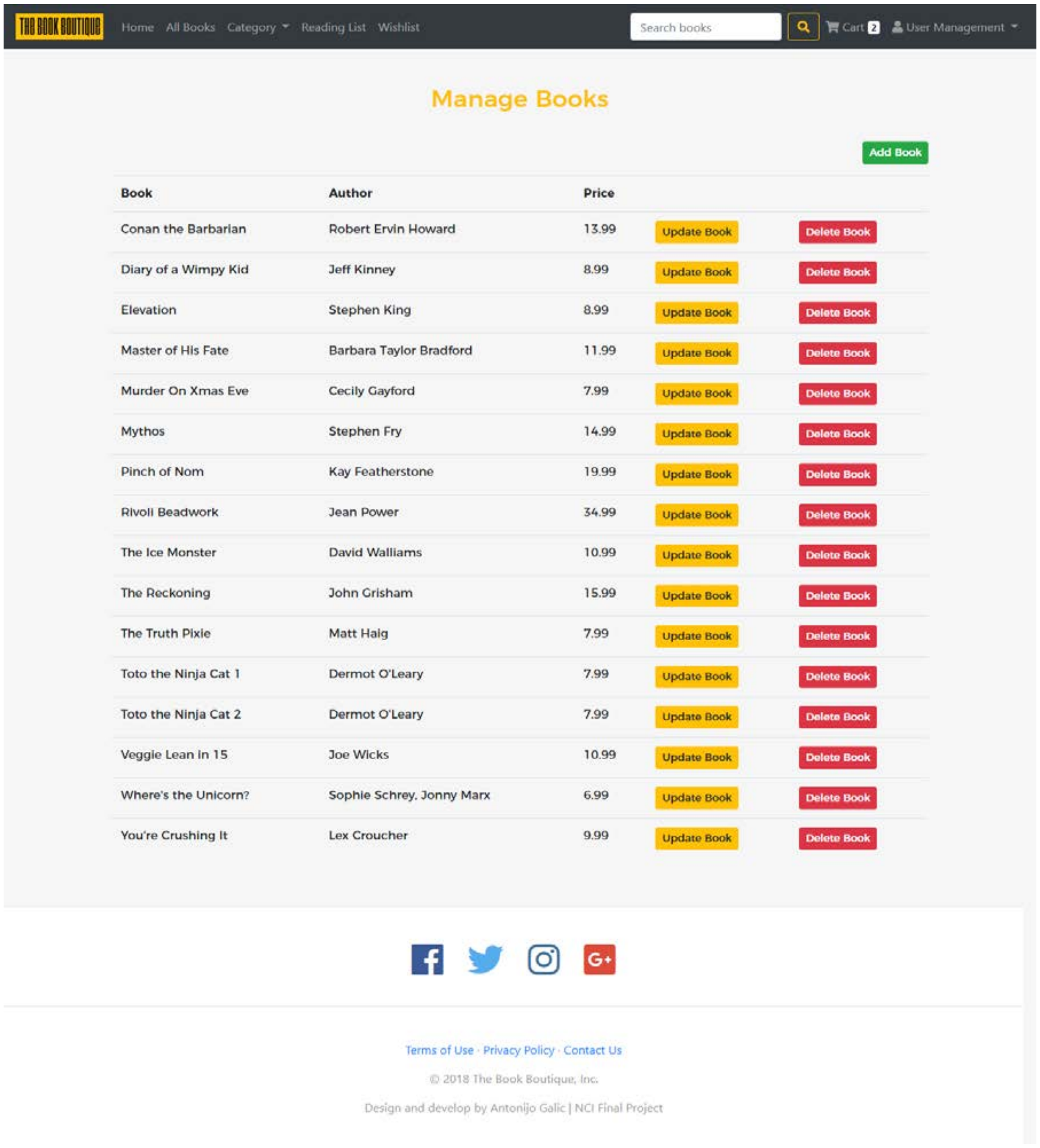

THE BOOK BOUTIQUE
Home
All Books
Category
Reading List
Wishlist
Search books
Cart 2
User Management
THE BOOK BOUTIQUE
Login
Email address
Enter email
Password
Password
Submit
Don't have an account? Register now!
Terms of Use · Privacy Policy · Contact Us
© 2018 The Book Boutique, Inc.
Design and develop by Antonijo Galic | NCI Final Project

Figure 37. Login page

2.5.10 Manage Books

Manage books page represented all books added to the web application. The user can sort books per author or per book name. For each book are displayed buttons to update book or delete the book from a web application. Green button for add book is displayed on the right side above table view of books already added to the web application.



Manage Books

Book	Author	Price		
Conan the Barbarian	Robert Ervin Howard	13.99	Update Book	Delete Book
Diary of a Wimpy Kid	Jeff Kinney	8.99	Update Book	Delete Book
Elevation	Stephen King	8.99	Update Book	Delete Book
Master of His Fate	Barbara Taylor Bradford	11.99	Update Book	Delete Book
Murder On Xmas Eve	Cecily Gayford	7.99	Update Book	Delete Book
Mythos	Stephen Fry	14.99	Update Book	Delete Book
Pinch of Nom	Kay Featherstone	19.99	Update Book	Delete Book
Rivoli Beadwork	Jean Power	34.99	Update Book	Delete Book
The Ice Monster	David Walliams	10.99	Update Book	Delete Book
The Reckoning	John Grisham	15.99	Update Book	Delete Book
The Truth Pixie	Matt Haig	7.99	Update Book	Delete Book
Toto the Ninja Cat 1	Dermot O'Leary	7.99	Update Book	Delete Book
Toto the Ninja Cat 2	Dermot O'Leary	7.99	Update Book	Delete Book
Veggie Lean in 15	Joe Wicks	10.99	Update Book	Delete Book
Where's the Unicorn?	Sophie Schrey, Jonny Marx	6.99	Update Book	Delete Book
You're Crushing It	Lex Croucher	9.99	Update Book	Delete Book

Terms of Use · Privacy Policy · Contact Us

© 2018 The Book Boutique, Inc.

Design and develop by Antonijo Galic | NCI Final Project

Figure 38. Manage books page

2.6 User testing

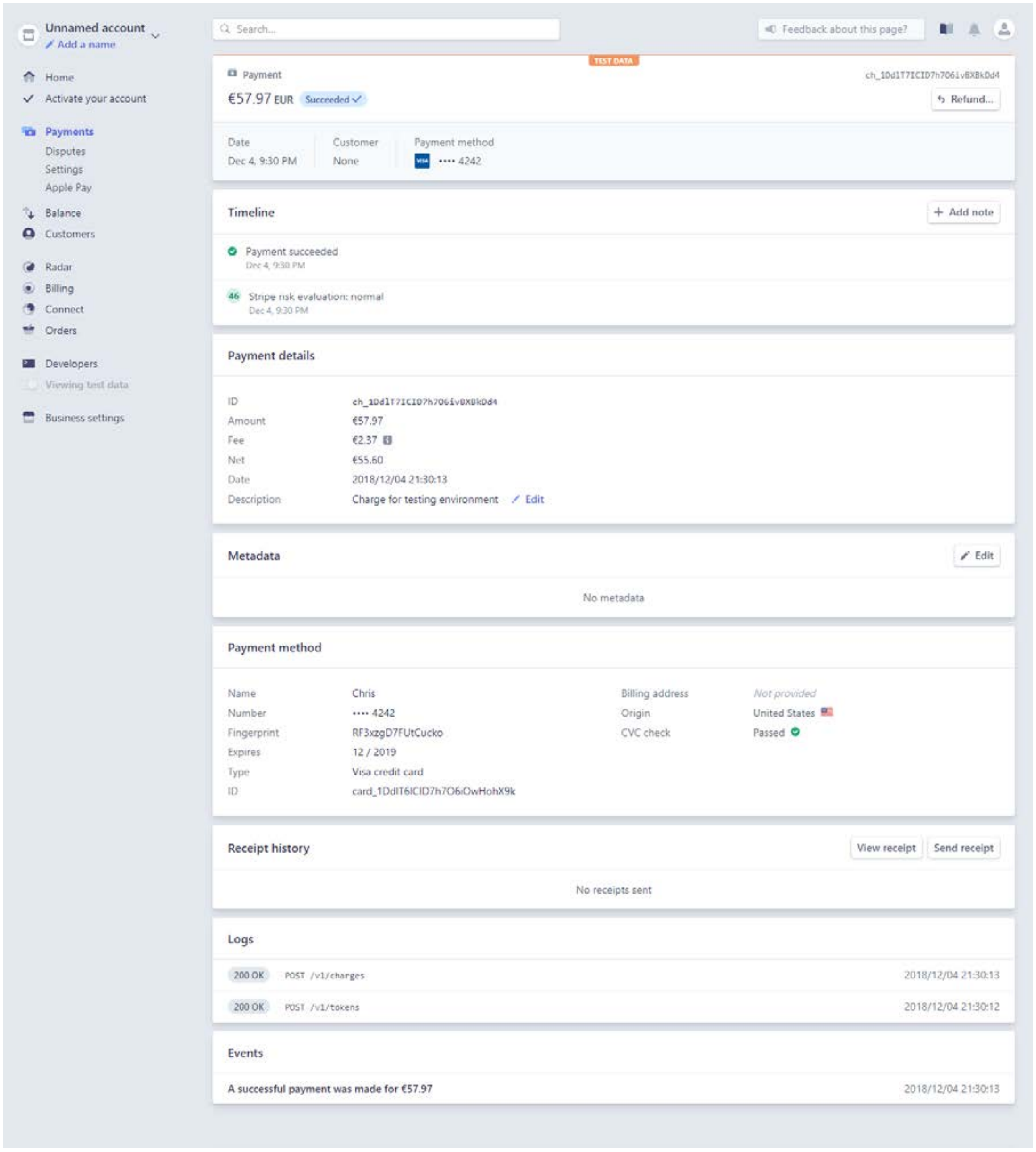
During User testing the following errors are discovered:

- Navbar is slightly breaking from 1094 px till 991px,
- Sorting per price in manage books section is not giving the satisfied result.

Except for the discovered errors user testing is considered positive because of the next:

- Users easy navigate through the application,
- Users adding successfully books to the cart, wishlist and reading list,
- Users successfully search for a book(s),
- Users successfully buying books after they proceed to checkout,
- In case of mistake from user side, user is able to finish the task after is shows an error message.

Users can easy check their order history also using Stripe dashboard. Using Stripe dashboard users need to go on their payments. Clicking on specific payment they can see detailed information about that payment (sample on the image below) and be sure that they are charged fair play.



Payment ch_1Dd1T7ICID7h706ivBXBkDd4

€57.97 EUR Succeeded ✓

Date	Customer	Payment method
Dec 4, 9:30 PM	None	•••• 4242

Timeline

Payment succeeded
Dec 4, 9:30 PM

46 Stripe risk evaluation: normal
Dec 4, 9:30 PM

Payment details

ID	ch_1Dd1T7ICID7h706ivBXBkDd4
Amount	€57.97
Fee	€2.37
Net	€55.60
Date	2018/12/04 21:30:13
Description	Charge for testing environment Edit

Metadata

No metadata

Payment method

Name	Chris	Billing address	*Not provided*
Number	•••• 4242	Origin	United States
Fingerprint	RF3xzgD7FUtCucko	CVC check	Passed ✓
Expires	12 / 2019		
Type	Visa credit card		
ID	card_1DdIT6ICID7h7O6iOwHohX9k		

Receipt history

No receipts sent

Logs

```
200 OK   POST /v1/charges   2018/12/04 21:30:13
200 OK   POST /v1/tokens    2018/12/04 21:30:12
```

Events

A successful payment was made for €57.97 2018/12/04 21:30:13

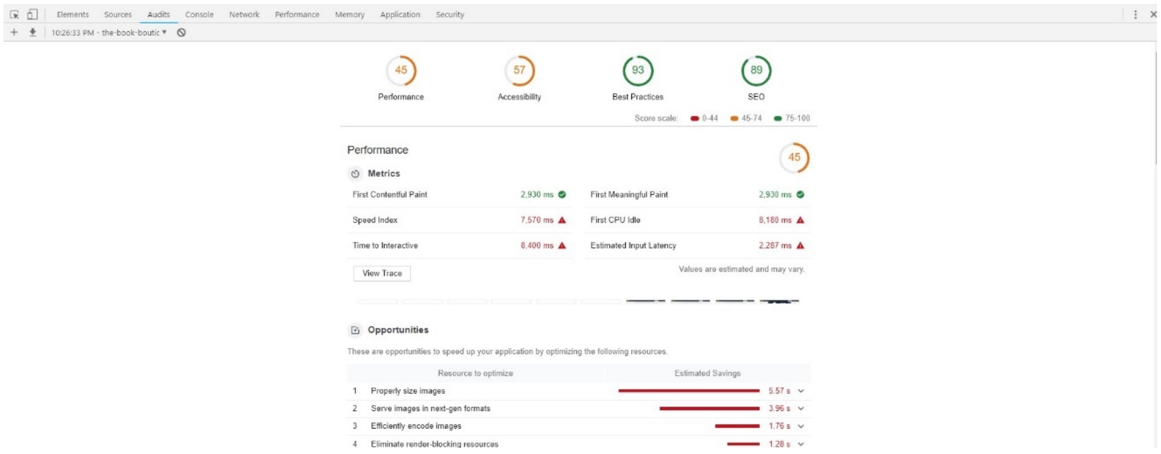
Figure 39. Payment details - Stripe dashboard

2.7 Evaluation

The Book Boutique e-commerce web application was evaluated from initial deployment to second evaluation and after that to third evaluation. For evaluation tool, it is used Google Lighthouse.

Lighthouse is an open-source, automated tool for improving the quality of web pages. You can run it against any web page, public or requiring authentication. It has audits for performance, accessibility, progressive web apps, and more. You can run Lighthouse in Chrome DevTools, from the command line, or as a Node module. You give Lighthouse a URL to audit, it runs a series of audits against the page, and then it generates a report on how well the page did. From there, use the failing audits as indicators on how to improve the page. Each audit has a reference doc explaining why the audit is important, as well as how to fix it. (Google, 2018)

The first deployment shows bad results in Performance and Accessibility where Best Practices and SEO has a great score. On the image below are results of the initial report.



45	57	93	89
Performance	Accessibility	Best Practices	SEO

Score scale: 🟥 0-44 🟧 45-74 🟩 75-100

Performance 45

Metrics

First Contentful Paint	2,930 ms ✓	First Meaningful Paint	2,930 ms ✓
Speed Index	7,570 ms ▲	First CPU Idle	8,180 ms ▲
Time to Interactive	8,400 ms ▲	Estimated Input Latency	2,287 ms ▲

Values are estimated and may vary.

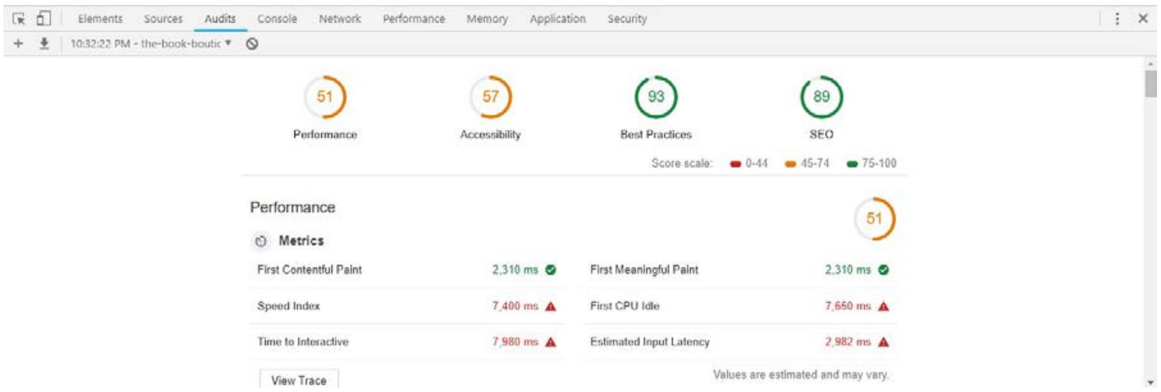
Opportunities

These are opportunities to speed up your application by optimizing the following resources.

	Resource to optimize	Estimated Savings
1	Properly size images	5.57 s
2	Serve images in next-gen formats	3.96 s
3	Efficiently encode images	1.76 s
4	Eliminate render-blocking resources	1.28 s

Figure 40. Initial Lighthouse report

After the developer do the image optimization next report (Figure 41) showing better results in Performance for 6%.



51 Performance | 57 Accessibility | 93 Best Practices | 89 SEO

Score scale: 0-44 45-74 75-100

Performance 51

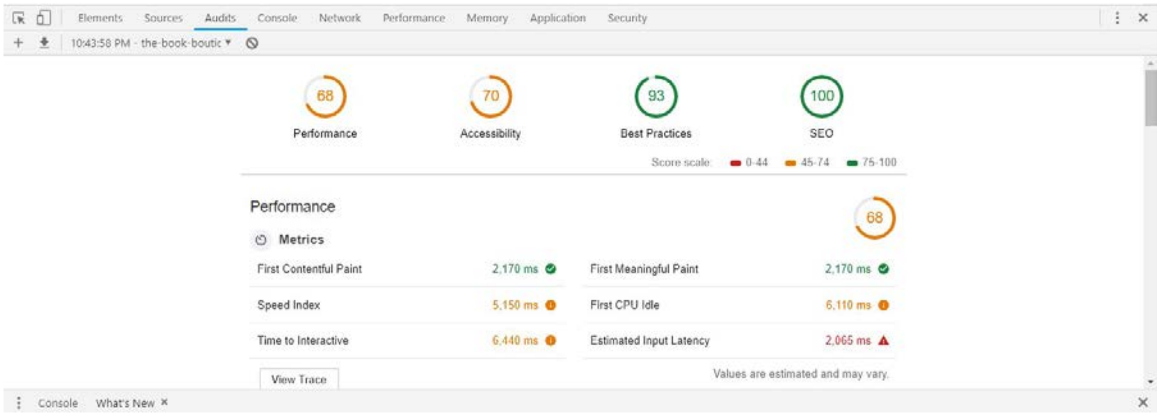
Metrics

First Contentful Paint	2,310 ms	First Meaningful Paint	2,310 ms
Speed Index	7,400 ms	First CPU Idle	7,650 ms
Time to Interactive	7,980 ms	Estimated Input Latency	2,982 ms

Values are estimated and may vary.

Figure 41. Second Lighthouse report

Next goal was to enhance Accessibility so developer added some code for meta tags and some alt attributes for images. After added extra code into application Performance rate raised for 17%, Accessibility for 13% and SEO for 11%. The final report is on the image below.



68 Performance | 70 Accessibility | 93 Best Practices | 100 SEO

Score scale: 0-44 45-74 75-100

Performance 68

Metrics

First Contentful Paint	2,170 ms	First Meaningful Paint	2,170 ms
Speed Index	5,150 ms	First CPU Idle	6,110 ms
Time to Interactive	6,440 ms	Estimated Input Latency	2,065 ms

Values are estimated and may vary.

Figure 42. Final Lighthouse report

In the future, the goal is to achieve better results but with limited time and limited experience of developer the results are satisfied.

3 Conclusions

There are many benefits of using Node.js. While most of its users mention the advantages related to the development process, such as increased developer productivity, improved developer satisfaction, and reduced development cost. Half of the respondents of Node.js 2017 User Survey noticed improved application performance in comparison to other solutions. (Ciszewski, 2018) Some of the reasons why Node.js was technology to develop this e-commerce web application are:

- Node.js is good at multitasking. It can process multiple tasks concurrently in one thread, instead of queueing them,
- Node.js works on the fastest V8 Javascript engine that is used in Google Chrome. Node.js is developed by Google which increases the chances that it will be well supported in the future,
- the technology stack enables the application to handle much more requests at the same time than other another solution.

This project has some limitations. One of the limitations is that user needs to create an account and be logged in into the web application to have full access. Full access to the web application, in this case, implies adding books(s) to cart, buying the book(s), leaving a review, adding a book to wishlist and on the reading list. The other limitation of this project is the time needed to develop a fully functional application, limited experience of developer and money needed for faster hosting of web application and database hosting.

Big disadvantages of web application and project itself are testing of web application. Unfortunately, with limited time and experience of the developer, the testing with Mocha and Chai which is firstly planned is cancelled and replaced with Google Lighthouse and TDD (Test Driven Development) during the development of the application.

On the end, developer is satisfied with the application because all functionalities necessary for work of web application are successfully implemented. The Book

Boutique web application after developing phase is deployed to Heroku service and it is live on https://the-book-boutique.herokuapp.com .

4 Further development or research

This project defines creating a web application to function as an online e-commerce web application for books. There are possibilities that web application could be evolved on selling other products connected with books like DVDs of movies based on books and similar. This e-commerce web application could be also evolved onto other products than books but that will require the name change of e-commerce web application.

To understand where is the limit in development of The Book Boutique e-commerce web application we can check the Amazon story. In July 1995, Amazon, the first online bookstore, launched a new business model by selling books online. This event-initiated competition between online bookstores and physical (brick-and-mortar) bookstores. Now, Amazon has grown to become the largest online bookstore in the world. Today, Amazon selling a lot more products than only books so only limit in further development of this e-commerce web application is the sky.

5 References

Academind, 2016. *Youtube - NodeJS / Express / MongoDB - Build a Shopping Cart.* [Online]
Available at: https://www.youtube.com/watch?v=-3vvxn78MH4&list=PL55RiY5tL51rajp7Xr_zk-fCFtzdlGKUp&index=2
[Accessed 5 12 2018].

Anderson, P., 2018. *Flexing Export Muscle Ahead of Brexit: The UK's Publishers Association Releases Its 2017 'Yearbook'.* [Online]
Available at: https://publishingperspectives.com/2018/07/publishers-association-2017-yearbook-emphasis-book-export/
[Accessed 5 October 2018].

Arneson, E., 2014. *CenturyLink.* [Online]
Available at: https://www.ctl.io/developers/blog/post/build-user-authentication-with-node-js-express-passport-and-mongodb
[Accessed 23 October 2018].

Beal, V., 2018. *Webopedia.* [Online]
Available at: https://www.webopedia.com/TERM/S/security_architecture.html
[Accessed 23 October 2018].

Ciszewski, B., 2018. *netguru.* [Online]
Available at: https://www.netguru.co/blog/nodejs-performance-web-application-benefit
[Accessed 23 October 2018].

Eeles, P., 2006. *IBM.* [Online]
Available at: https://www.ibm.com/developerworks/rational/library/feb06/eeles/index.html#notes
[Accessed 22 October 2018].

Google, 2018. *Tools for Web Developers - Lighthouse.* [Online]
Available at: https://developers.google.com/web/tools/lighthouse/#devtools
[Accessed 6 12 2018].

Inflectra, 2018. *Inflectra.* [Online]
Available at: https://www.inflectra.com/ideas/topic/requirements-definition.aspx
[Accessed 12 October 2018].

Liu, J., 2015. Design and Implementation of Online Bookstore Based on JSP and JavaBean Technology. *Modern Informatica.*

Mithun, S. & D'mello, B. J., 2017. *Web Development with MongoDB and Node.* Third Edition ed. s.l.:Packt Publishing.

Osetskyi, V., 2018. *DZone.* [Online]
Available at: https://dzone.com/articles/how-to-write-the-system-requirements-specification
[Accessed 12 October 2018].

Stringfellow, A., 2017. *Stackify.* [Online] Available at: https://stackify.com/web-application-architecture/ [Accessed 21 October 2018].

Svitla, 2018. *Svitla.* [Online] Available at: https://svitla.com/blog/web-application-architecture [Accessed 21 October 2018].

Techopedia, 2018. *Techopedia.* [Online] Available at: https://www.techopedia.com/definition/5435/graphical-user-interface-gui
[Accessed 15 October 2018].

Terkaly, B., 2014. *Microsoft | Developer.* [Online] Available at: https://blogs.msdn.microsoft.com/brunoterkaly/2014/02/02/installing-express-and-how-to-use-node-js-packages-with-visual-studio/
[Accessed 22 October 2018].

Traversy, B., 2018. *Node.js, Express and MongoDB Dev to Deployment.* s.l.:Packt Publishing.

Unknown, 2018. *handlebars vs pug.* [Online] Available at: https://www.npmtrends.com/handlebars-vs-pug [Accessed 5 October 2018].

Unknown, 2018. *U.S. Book Industry/Market - Statistics & Facts.* [Online] Available at: https://www.statista.com/topics/1177/book-market/ [Accessed 5 October 2018].

Vmoksha, 2017. *Vmoksha.* [Online] Available at: https://vmokshagroup.com/blog/building-restful-apis-using-node-js-express-js-and-ms-sql-server/
[Accessed 22 October 2018].

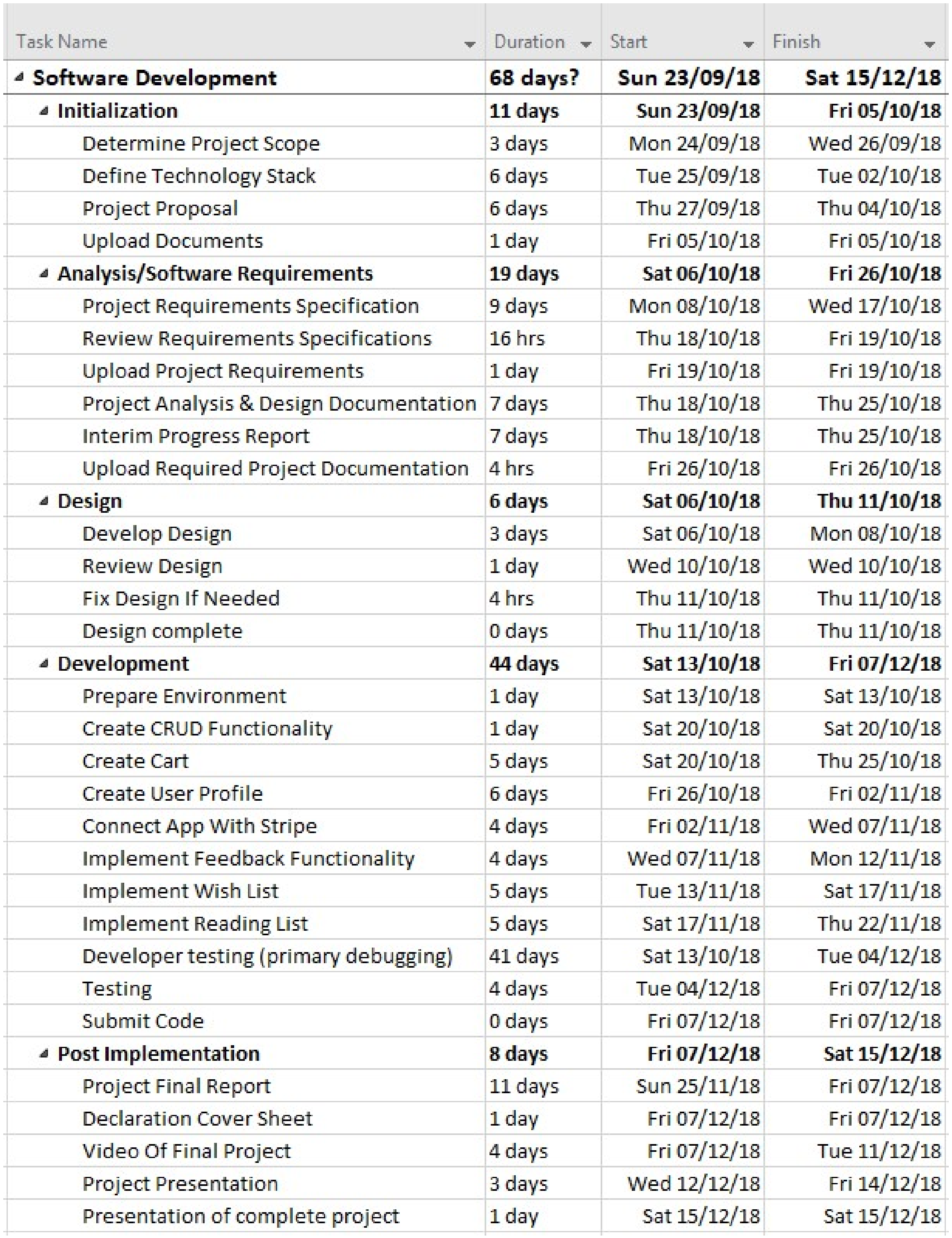
Zhai, Y. & Lu, W., 2017. *The online bookstore.* [Online] Available at: https://www.matec-conferences.org/articles/matecconf/pdf/2017/14/matecconf_gcmm2017_02045.pdf
[Accessed 5 October 2018].

6 Appendix

6.1 Project Proposal

Project proposal for the e-commerce application The Book Boutique is described in the Project Proposal document.

6.2 Project Plan



Task Name	Duration	Start	Finish
Software Development	**68 days?**	**Sun 23/09/18**	**Sat 15/12/18**
Initialization	**11 days**	**Sun 23/09/18**	**Fri 05/10/18**
Determine Project Scope	3 days	Mon 24/09/18	Wed 26/09/18
Define Technology Stack	6 days	Tue 25/09/18	Tue 02/10/18
Project Proposal	6 days	Thu 27/09/18	Thu 04/10/18
Upload Documents	1 day	Fri 05/10/18	Fri 05/10/18
Analysis/Software Requirements	**19 days**	**Sat 06/10/18**	**Fri 26/10/18**
Project Requirements Specification	9 days	Mon 08/10/18	Wed 17/10/18
Review Requirements Specifications	16 hrs	Thu 18/10/18	Fri 19/10/18
Upload Project Requirements	1 day	Fri 19/10/18	Fri 19/10/18
Project Analysis & Design Documentation	7 days	Thu 18/10/18	Thu 25/10/18
Interim Progress Report	7 days	Thu 18/10/18	Thu 25/10/18
Upload Required Project Documentation	4 hrs	Fri 26/10/18	Fri 26/10/18
Design	**6 days**	**Sat 06/10/18**	**Thu 11/10/18**
Develop Design	3 days	Sat 06/10/18	Mon 08/10/18
Review Design	1 day	Wed 10/10/18	Wed 10/10/18
Fix Design If Needed	4 hrs	Thu 11/10/18	Thu 11/10/18
Design complete	0 days	Thu 11/10/18	Thu 11/10/18
Development	**44 days**	**Sat 13/10/18**	**Fri 07/12/18**
Prepare Environment	1 day	Sat 13/10/18	Sat 13/10/18
Create CRUD Functionality	1 day	Sat 20/10/18	Sat 20/10/18
Create Cart	5 days	Sat 20/10/18	Thu 25/10/18
Create User Profile	6 days	Fri 26/10/18	Fri 02/11/18
Connect App With Stripe	4 days	Fri 02/11/18	Wed 07/11/18
Implement Feedback Functionality	4 days	Wed 07/11/18	Mon 12/11/18
Implement Wish List	5 days	Tue 13/11/18	Sat 17/11/18
Implement Reading List	5 days	Sat 17/11/18	Thu 22/11/18
Developer testing (primary debugging)	41 days	Sat 13/10/18	Tue 04/12/18
Testing	4 days	Tue 04/12/18	Fri 07/12/18
Submit Code	0 days	Fri 07/12/18	Fri 07/12/18
Post Implementation	**8 days**	**Fri 07/12/18**	**Sat 15/12/18**
Project Final Report	11 days	Sun 25/11/18	Fri 07/12/18
Declaration Cover Sheet	1 day	Fri 07/12/18	Fri 07/12/18
Video Of Final Project	4 days	Fri 07/12/18	Tue 11/12/18
Project Presentation	3 days	Wed 12/12/18	Fri 14/12/18
Presentation of complete project	1 day	Sat 15/12/18	Sat 15/12/18

6.3 Requirement Specification

Requirement specification for the e-commerce application The Book Boutique is described and documented in the Requirement Specification document.

6.4 Product Design Specification

Product design specification for the e-commerce application The Book Boutique is described and documented in the Product Design Specification document.

6.5 Figures

6.6 Application Links and Admin Access

GitHub: https://github.com/Galiant/NCI-Final-Project

Heroku: https://the-book-boutique.herokuapp.com/

Admin access:

- Email address: admin@gmail.com
- Password: admin