Online Bookstore for a Customer-to-Customer E-commerce Web Application

CAPSTONE PROJECT

School of Information

CHAIR

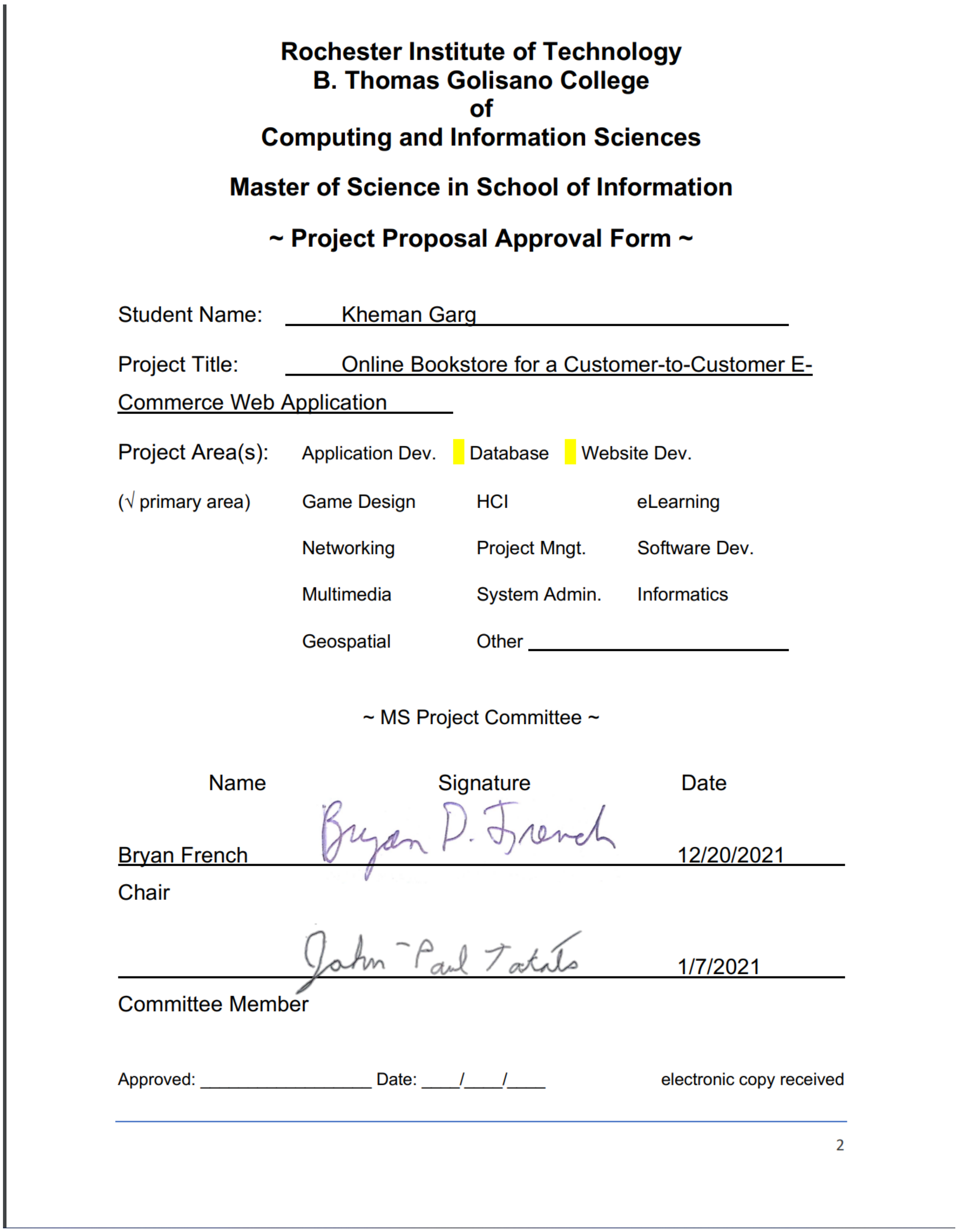
Prof. Bryan French

COMMITTEE MEMBER

Prof. JOHN-PAUL TAKATS

PROPOSAL BY

Kheman Garg



Rochester Institute of Technology
B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in School of Information

~ Project Proposal Approval Form ~

Student Name: Kheman Garg

Project Title: Online Bookstore for a Customer-to-Customer E-Commerce Web Application

Project Area(s):	Application Dev.	[✓ Database]	[✓ Website Dev.]
(√ primary area)	Game Design	HCI	eLearning
	Networking	Project Mngt.	Software Dev.
	Multimedia	System Admin.	Informatics
	Geospatial	Other ________	

~ MS Project Committee ~

Name	Signature	Date
Bryan French Chair	Bryan P. French	12/20/2021
Committee Member	[illegible]	1/7/2021

Approved: ________________ Date: ____/____/____ electronic copy received

Table of Contents

TABLE OF FIGURES

1. ABSTRACT

The facility to carry out quick, safe, and economical processes are the invention of e-commerce applications. Customer-to-customer trading of goods and services is a new, modern, and rapidly growing in-demand type of e-commerce application. RentBooks is a monolithic architecture web application that will provide peer-to-peer book rental services and with the ease of purchasing as well as selling books at a cheap price. With all the academic and cost of living fees, students already feel burdened, and rather than buying new books every single term they can easily buy them from other students for rental at a cheaper price across the campus. The books are not bound to just academics, but they also contain diverse genres of books along with the reviews written by customers. [1] This project can help solving the problem for students by helping them to learn about the book before purchasing it and read a plethora of books at a cheap price, which can help in boosting their knowledge and they can also gain money out of this web application. The system is implemented using Nuxt.js and Node.js whereas the database used is MongoDB which is a NoSQL database on MongoDB Atlas which is a multi-cloud database service accessible from anywhere in the world and it is tested on both Google Chrome and Mozilla Firefox.

KEYWORDS: E-commerce, Used Book system, Full Stack application, Rental Services

2. INTRODUCTION

Advancement in the internet industry has given birth to some amazing new businesses, one of which is known as e-commerce where a company sells a product to its consumers via online method. E-commerce started working because of the electronic data exchange which was invented when companies were trying to get paperless offices. Nowadays new e-commerce applications are taking place, and one of which includes direct customer-to-customer trading for different goods and services. Different social media companies have opened features like a marketplace to sell and buy abundant material. In my opinion, there is a great need for such an application where the items are easily exchanged or traded within a small community like a university. There are famous websites like eBay or Amazon but they act on a large scale rather than focusing on smaller groups.

This project focuses on a customer-to-customer book rental application built using a modern tech stack of Nuxt.js, Node.js, and MongoDB. This tech-stack helped in making a web application that fulfill needs like modern, fast, and reusable. Hence this project can be further explored to convert it into a microservices architecture in future. The application has two types of users: admin and consumer where consumer can buy or sell items and admin can control the list of users and can add products, type of product and owners. To maintain the security of the application the accounts are verified by sending an email to user's account that will need to be confirmed by the user in order to sign up. Then the project further uses the JSON Web Tokens (JWT) in order to secure the password hence improving the security. Another feature that bolsters the security of application is addition of sanitization and validation of inputs after the end user submits any type of form. This project is creating a payment gateway using the Stripe Application Programming Interface (API). Since the payment is a transactional feature, therefore, currently the Stripe is used to work on only development mode but with the ease of changing to production mode by just replacing the API key.

The application is a user-friendly and attractive in the design along with the use of proper validations on the form to prevent errors in data. The design of the application is inspired from the amazon website. These validations along with the sanitization of data is provided in the back end as well for security and data integrity. A search bar is also used to ease users in finding their desired book easily along with the location. This search bar is built using the Algolia API which is not only helping in filtering the results but also collecting data on search results and generating it in the Algolia app which can be used to improve the application and books. Features like recently viewed items has been established to help users to select the product more easily. Another feature will be to give ratings and reviews to the books which help users to identify if the product is right for them. In order to learn more about the book and to help in selling price recommendation there is a section which requires just the ISBN number which can be found inside the book that will help in finding the book in open-source openlibrary.org that will fetch the results. This would help them in easily finding the desired books and reading them at a cheaper price or earn money on selling books on rent. This application can help in reducing the waste by reusing the same books by different number of people.

Even though there are pool of opportunities to extend this application, one important thing to keep in mind is the privacy and security concerns for an application. [4] These concerns will be discussed in detail to make sure the growth of the bookstore application.

RentBooks is using the monolithic architecture for this project, meaning all of the application will be stored in a single codebase. In my opinion, this architecture is specifically preferred over microservices for initial development because it will be simple to develop and deploy a single unit rather than making a huge application in multiple parts and maintaining different codebases. There are some advantages that microservices provide like high scalability and loosely coupled code. It is easier to add functionalities and use multiple tech stacks while choosing microservice architecture. However, in my opinion, a microservices architecture application is difficult to build and maintain for a single developer and it is easier to start with the monolithic architecture. Therefore, in the future if the application grows in popularity and more functionalities or new services can be added to this project, then this application can be converted into microservices.

3. PROBLEM STATEMENT

In contemporary times there are a lot of new web applications that deal with real-life situations and make things easy for people in different ways. There are times when a person has a book and after some time that book is useless to them and just throw away the book. Instead of that, it can be reused by someone else through this web application who otherwise would have purchased the same book from a retailer at a high price. While purchasing from the retailer they might not know the review about the book but while using this application this issue can be resolved as there is a review as well as rating system on books. These books can be further sold to other customers using this application, hence making it cheap. Even though there are libraries in the college to borrow books, they provide limited supply of books as there is a restriction on number of books a student can borrow. Other sources of purchase like amazon and eBay act at a large scale and not university-based small clusters. So, in my opinion there are fewer options or fewer chances to have the same book that is required at the specific college.

4. FUNCTIONALTIES:

4.1 List of functionalities:

1. Email verification for the new user on sign-up to help bolster the security.
2. Created separate admin and client applications to separate them from each other.
2. Users can buy books at rent or full price. Also, sellers can sell books at prices generated from book lookup system.
3. Implemented a book lookup system to help sellers in pricing the book and learn about the book.
4. Created a payment gateway for customers to buy books using the Stripe API.
5. Implemented Algolia search to collect data on searches being made in order to improve application for future.
6. Built a recent watched products system to help customers with a better customer experience.

4.2 What makes it unique from other Competitors?

A Book lookup system will be one of the main features/functionalities that will help differentiate this application from the other apps. It is common to see selling prices go out of proportion on eBay while selling used products. It is really important to curb this problem which most of the websites lack. This application will have a section of price comparison where users can go and search for the book, they are about to sell to check the selling price from other websites and figure out the right selling price for the book.

There are 2 ways to apply price comparison which will be explored while building the project:

1. Web Scraping: This is the cheapest way and it gives most of the control to how or what data someone needs. This can be done by either building a web scraper bot or using third-party libraries.

2. Product feeds from APIs: Nowadays**,** most of the big e-commerce applications have open-source APIs which can be used for personal applications. They are easy to use as they solve the complexity which will be seen in scrapping. The caveat for this is it can be expensive as it costs after exceeding free API calls which vary for different APIs.

5 LITERATURE REVIEW

5.1 Intelligence design of a web application for customer to customer training:

It discusses the student trade application built for students who perform consumer-to-consumer trading using the internet. These items include books, household items, and electronics, and sports services. Different algorithms are tried and used to improve the price rate for the book when it goes up for trading. Besides setting the price this paper also talks about the different recommendation systems in the application that can be used to improve sales by forming the previous purchase pattern made by the user. Three methods can be used to build a recommendation system, i.e. content-based, collaborative based and hybrid-based. Content-based will be the recommendation of products based on the previous searches that the user has made while collaborative-search is the recommendation of the products that the other users have searched with similar characteristics. Hybrid as the name suggests is the mix of both content-based and collaborative-based. So, to improve the application this paper suggests a recommendation system for buyers that provide real-time searches and related items in the marketplace to come up with a price for a sale item. On the other hand, a buyer can express interest in the products if they are related/desirable to their needs. Hence in the end making the consumer-to-consumer application more intelligent and more user-friendly. [2]

5.2 Bookland – Android application for rental books:

An Android application built for mobile discusses the implementation of the bookstore and the system design that was used to build the application. This paper discusses what architecture they followed to build the application and the features that they injected into the application. The most important thing in this paper that can be seen is the flow of the user from user login to payment gateway and further deleting the number of books available. This flow chart is necessary to build to learn the flow of how the user will interact with the application and hence help in making it user-friendly. There are the database model and tables that are discussed in detail which can be used to form the relational database. [3]

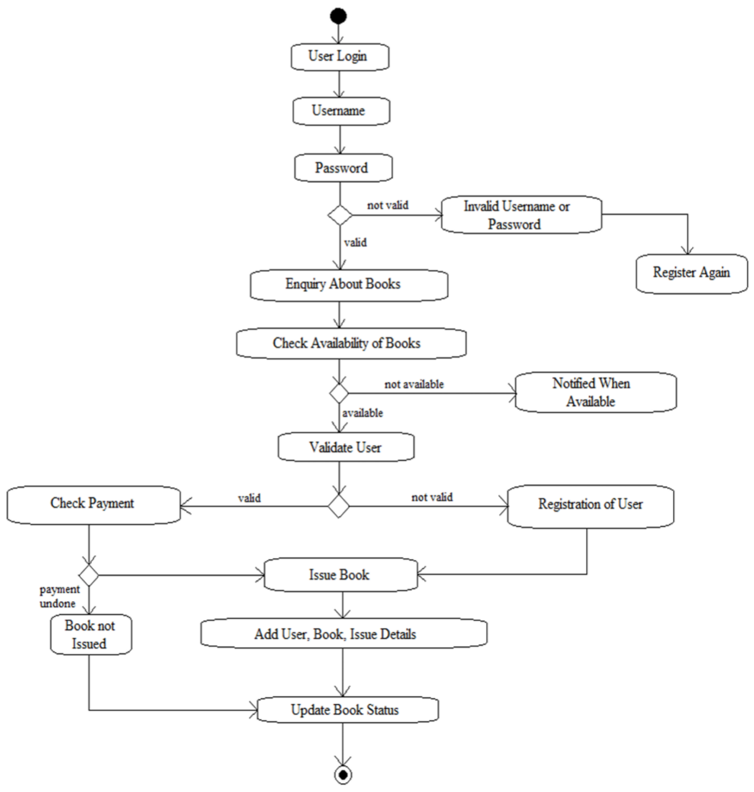


Figure 1. System Architecture [3]

5.3 <u>Online E-book store web design:</u>

Another research paper discusses each page and its functionality in the bookstore application in detail. The proposed design in this paper talks about how the authorization can be built among different parts of the application and how the database gets updated on each entry that is being made. It also discusses in detail what functionalities admin inherit like adding or removing a book from any user. [1] It shows excellent user-friendly interfaces and database management ideas that can be used. Security of the database can be improved if the admin will be able to add/remove books. Beside that it also discusses how admin has control over users in order to access the application and how they can give users certain permissions and authorization to access certain pages in the application.

6. METHODOLOGY

The goal of this project is to develop a general consumer to consumer e-commerce application with appealing user interface and strong backend. One of the key ideas behind this project is its innovation and mission to help student in learning through textbooks that are cheap in price. The system in this application has fixed the ease of selling the books as the user can easily go lookup the book using the ISBN number and determine the selling and cost price, including the details about the book. It bolsters the security of the application using number of features like JWT, Email Verification, Validation and Sanitization. The system inside application is highly configurable as the database of this application is designed in such a way that it can be configurable easily as adding a field in table or adding new table is very easy for MongoDB and hence easily customized with the needs of the project. These design decisions were taken to provide flexibility for a developer as relational database can increase the complexity in the application.

This project is divided into four phases, i.e., starting with the bare minimum template then followed by database design and creating endpoints in the backend. At last, a NUXT project will start where the user interface will be developed and integrated with the backend.

6.1 Architecture

This application follows the client-server network architecture. Client-server network architecture is a computer network architecture that consists of two components: client and server. A server helps listens for requests from client, processes those requests and sends response back to the client as shown in Figure 2. The client will act as a view interface that will be interacting with end users.[5]

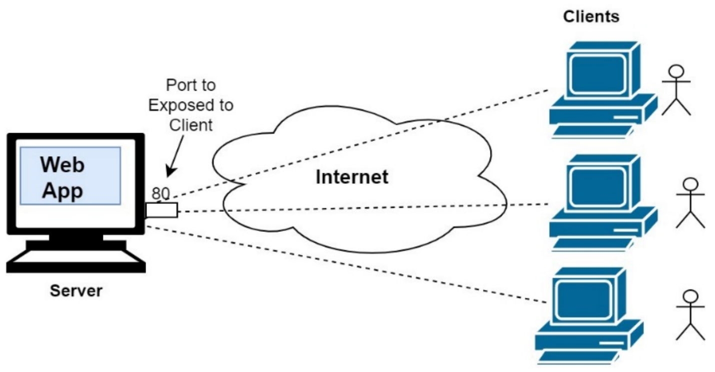


Figure 2 Client Server Architecture

The application is built on The Model-View-Controller (MVC) paradigm. It is the software architecture that is used for developing web applications. In this, a software application is divided into three interconnected logical components that are model, view and controller. It makes it easy to develop web applications. The model manages the data and fundamental behaviors of the application.[6] It can respond to the request for information, change states and notify observers/events that are fired on change of events. The controller acts as the collection of endpoints and business logic which takes in the user requests and save it to the model or respond with the appropriate response that is designed in the application. The view is basically the user interface part of the application. When end user starts the application it sends to request to the server which is handed by the controller which fetches the result from the database models and provide it to the end user.

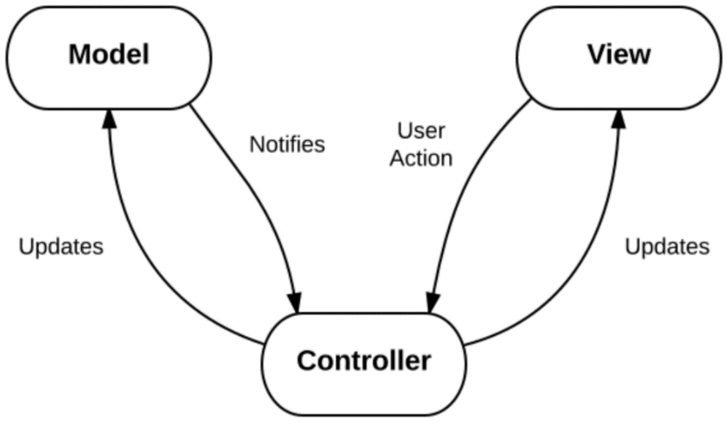


Figure 3 MVC Architecture

Reasons behind choosing MVC over other architectures is its support for parallel development. It allows to work on UI as well as Rest Services in parallel which helps in faster development and help in easily identifying the error in the code. Nuxt.js which handles the view of the application communicates with the database model via HTTP REST services and fetches or updates data in the database. The application uses HTTP are:

- GET: This method is used to retrieve the data from the server
- POST: This method is used to push data on the server
- PUT: This method is used to update the existing data on the server.
- DELETE: This method is used to delete the data on the server.

6.2 Tools and Technologies

The application was built using the MongoDB, Express, Vue.js and Node.js (MEVN) Full-stack technology stack which is discussed as following:

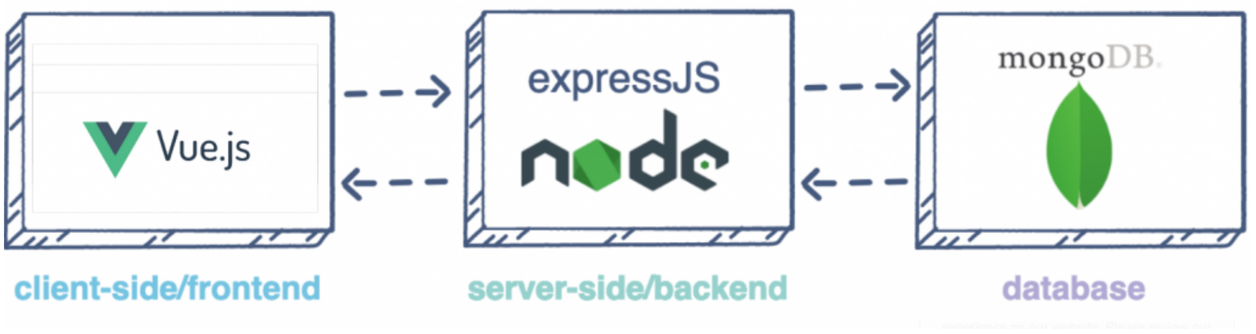


Figure 4 MEVN Technology

MongoDB:

MongoDB is a NoSQL database management system that uses a JavaScript Object Notation (JSON) styled storage format Binary JSON (BSON) which is an extension of JSON that helps in encoding type and length information which makes the parsing faster. BSON also helps in storing the binary data and types to efficiently index, map and nest data in support of complex query operations and expressions. One of the main advantages of using the NoSQL database is that it does not have a relational structure of data which means it isn't enforced hence it reduces the application development process by reducing the complexity and build up process during deployments.

In order to use MongoDB with server there is a node package called Mongoose. Mongoose helps as it acts a data modeling library that manages relationships between data and the schema in the backend. It also helps in modeling the JSON objects to the representation of the objects in MongoDB Atlas which is a multi-tier cloud platform accessible from anywhere in the world. MongoDB cloud database helps in development and production of websites without creating any database servers manually and it allows remote access to these databases from anywhere.[7] It also makes it possible to read the data directly from the database instead of writing code for it separately in some Extensible Markup Language (XML) file or any other data object relational mapping (ORM) tool.

Reason behind using NoSQL database instead of SQL relational database is first of all it is popular to use MongoDB with Node.js compare to SQL. SQL contains tables with fixed rows and columns which in order to convert to JSON objects requires ORM which makes the application complex as it forms another layer between server and database. Schemas in SQL are rigid whereas NoSQL are flexible and therefore faster while developing.

Express:

Express is a lightweight Node.js framework that helps in writing the back-end code for a web application. It helps developer in creating endpoints and a Web server in a simple, flexible and scalable way with minimum lines of code. It provides a thin layer of fundamental web applications without hiding any node.js feature. It is based on a Node.js middleware module called "connect" that helps in extending the built-in-server functionality. Middleware helps in making code shorter with just adding them in the code while comparing it to the Node.js. Express is asynchronous and hence doesn't affect the functionality of other features that are present in the website. The reason for selecting Express is the reusability that the library provides in the code to perform essential server functions like parsing of the payload, cookies, storage sessions. It also helps in reusing the selection of this route pattern. It is also Fast and efficient with a great reach of developers all over the world.

Nuxt.js:

It is free and opensource JavaScript library built on top of the famous JavaScript framework called Vue.js. This framework is built inspiring from Next.js which is built on top of the React framework. The most important feature of this library is that it handles the routing by itself and creates a good Search Engine Optimization (SEO). Developer does not need to install complex routing libraries while using Nuxt.js. It makes the configuration and setup of the application very easy and developer can quickly start with the project. Vue.js which is the main body of this library is a progressive JavaScript framework which is easy to approach and highly performant with extremely versatile as it is easily scalable. It uses the vue extension files instead of JS files like React, and add HTML, CSS, JS in a single file for each component. In my opinion this framework is rising in the market besides React and Angular. The reason for using Nuxt.js is that it handles the routing itself and it has good SEO.

Node.js:

Node.js is very popular JavaScript run-time environment that includes everything that a program needs to execute. The reason behind the invention of this application was to run JavaScript code outside the browser and can act as a standalone application. Node.js has many application in a diverse range as it can be used in building Internet of Things (IOT), Video String, Live chatting, e-commerce or any kind of consumer based application. Node.js which is also known as back-end JavaScript runs on V8 engine and executes JavaScript outside browser. It also helps developer to write the code in command line tools which are used in server-side scripting hence resulting in dynamic web page content. The architecture that Node.js use is event-driven architecture which is capable of asynchronous Input/Output (I/O).[8] This design was specifically aimed because it can help in optimizing throughput and scalability in web applications and hence making it good for real-time web applications. The reason to select Node.js is that it is lightweight and efficient and it does not need to run every time you change code instead it can auto-compile every time you change something using a node package called nodemon thus making it a right choice to build this application along with this technology stack.

Vuex:

Handling data in Vue/NUXT.js framework becomes complicated as the application starts growing therefore Vue brings an interesting concept of State management pattern and library which handles the data globally that you can manipulate and fetch anytime from any component. This state management pattern + library is known as Vuex. It has a centralized store for all the components in the application and thus reducing the complexity of the application. It has rules ensuring that state can only be mutated in a certain style. The only disadvantage of this library is that it costs to learn more concepts but once a developer can learn them then it makes the development lifecycle easy and efficient. Since my app was getting complicated so I introduced Vuex in the application to make some data centralized and manipulate it through different components. In the code it is always situated inside the store folder.

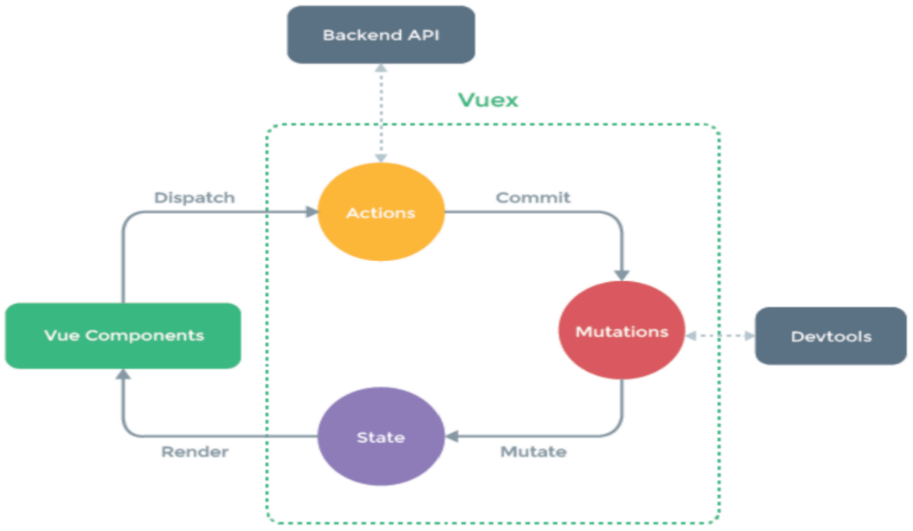


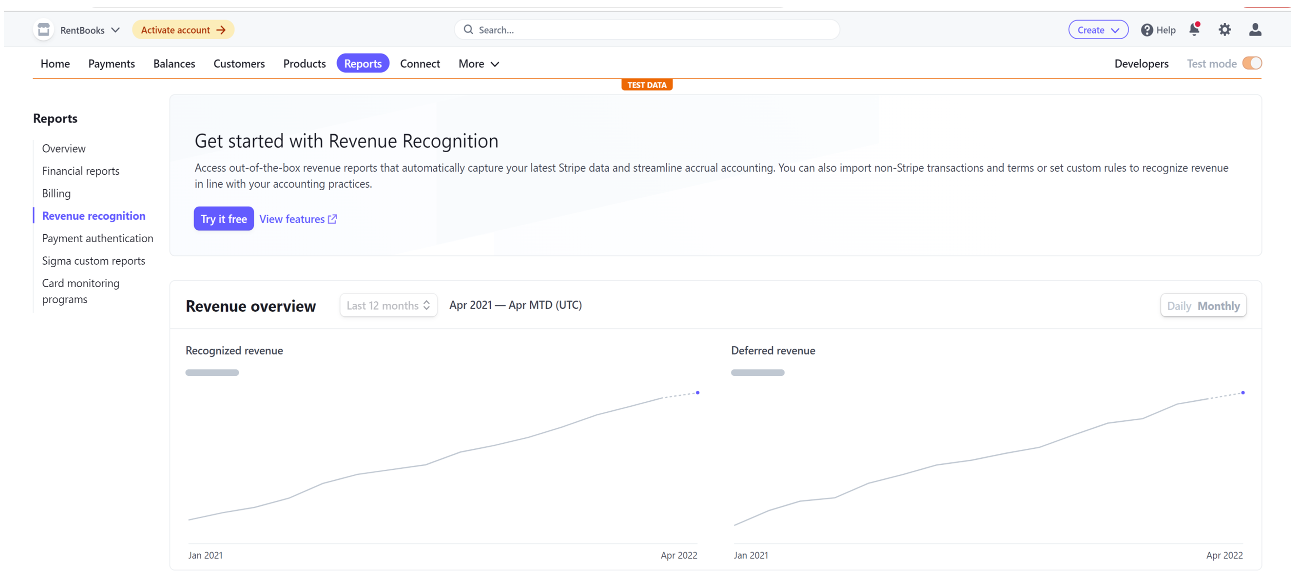
Figure 5 Vuex Store Management System

Amazon Web Services S3:

Amazon Simple Storage Service (Amazon S3) is used as object storage service as it offers a cloud storage that is one of the best platforms when it comes to scalability, data availability, security and performance. There is plethora of industries that trust this platform when it comes to these features. This is also cost-effective storage classes which contains many user-friendly features that are easy to use and optimizes the costs. It also helps in organizing data and configure fine tuned access controls and providing authorization services to these stored objects. In my application I am storing all kind of images in the Amazon S3 which gives a link back to me for that image and I can further store that link in my MongoDB database thus saving space in the NoSQL database and faster read and writes of the images in the application.

Stripe API:

It allows developers to access the Stripe which is a payment gateway application. This API can help in sending invoices, accepting payments, managing subscription billing and editing managing account information. It also allows users to accept payment online which is the main usage for my application. Along with that stripe also provides access to a dashboard where you can keep track of the payments and it also gives data insights on the products that are selling.[9] Good thing about this API is that it has two different keys for development and production mode which makes it easy to use in the application depending on the environment.



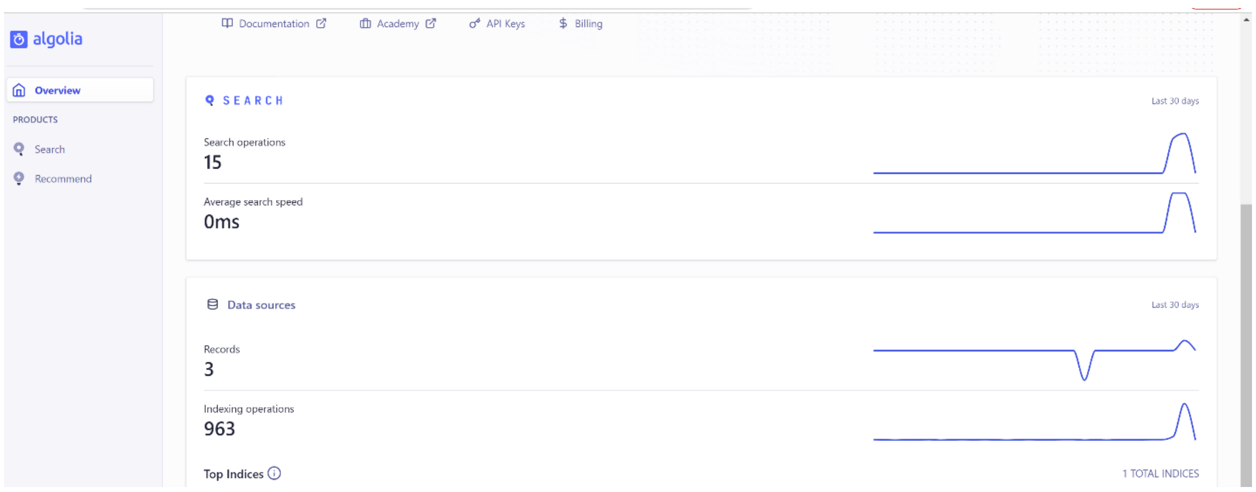
Get started with Revenue Recognition

Access out-of-the-box revenue reports that automatically capture your latest Stripe data and streamline accrual accounting. You can also import non-Stripe transactions and terms or set custom rules to recognize revenue in line with your accounting practices.

Figure 6 Stripe Dashboard

Algolia API:

Algolia is a powerful hosted search engine just like Amazon Elasticsearch or Apache SOLR. It offers a full-text, numerical and even facet-based searches that are capable of delivering real time results from the very first character. This API is implemented with database model that is built inside the node.js server for MongoDB and then it unlocks the feature of quickly and seamlessly implementation of search with the website search bar. It delivers results under 100ms anywhere in the world. It also supports a good data analytics system which provides what the users are mostly searching about and it can help in improving the application further by these insights.



Search operations
15

Average search speed
0ms

Records
3

Indexing operations
963

Figure 7 Algolia Dashboard

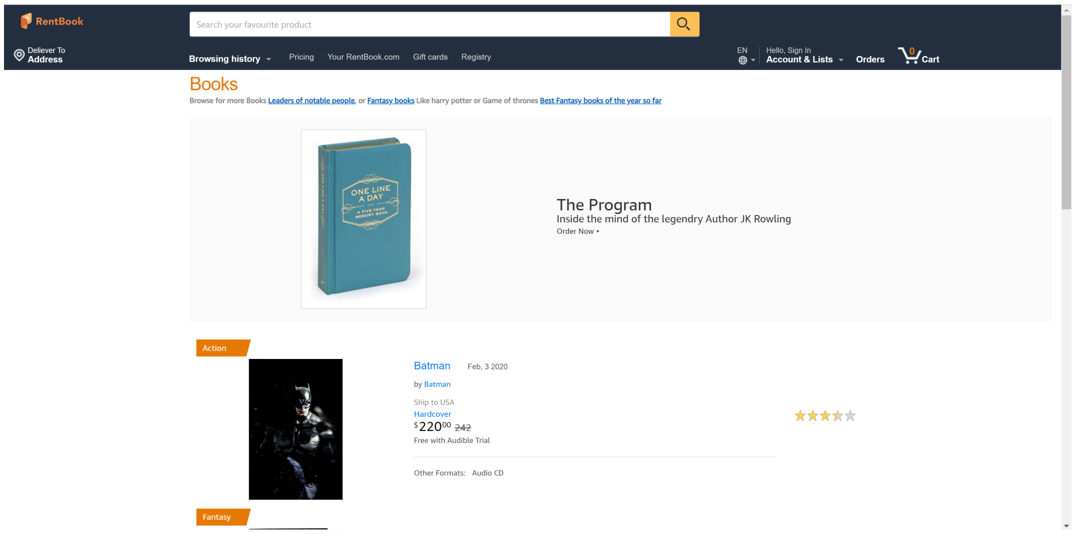
REST API:

Representational State Transfer (REST) API is an architectural style that is commonly used for web services. It helps in interacting data between web server and user. REST API is built on six principles that are providing a wall between client and server, consistent interface, reusability of response data, multi-layered systems and allowing running API code in the form of applets as well as scripts. The HTTP calls that are made using REST API are stateless hence making it compatible with cloud-based applications as they can be easily redeployed in case if something fails. It can also help in scaling or if there are any load changes.[10] Some of the HTTP methodologies used by the REST architecture in this project are:

- GET: This method is used to retrieve the data from the server
- POST: This method is used to push data on the server
- PUT: This method is used to update the existing data on the server.
- DELETE: This method is used to delete the data on the server.

7. RESULTS

RentBooks is mainly focused on students who are not able to afford or don’t want to spend much on new books can go through this website and get a book on rent or purchase a used book. When they launch the website, they will be interacted with the books that are available. They will have to sign up in the application to see the payment gateway or selling options. Here are some of the images how of how landing page and sign up/ sign in page looks like. The design is inspired from the Amazon website.

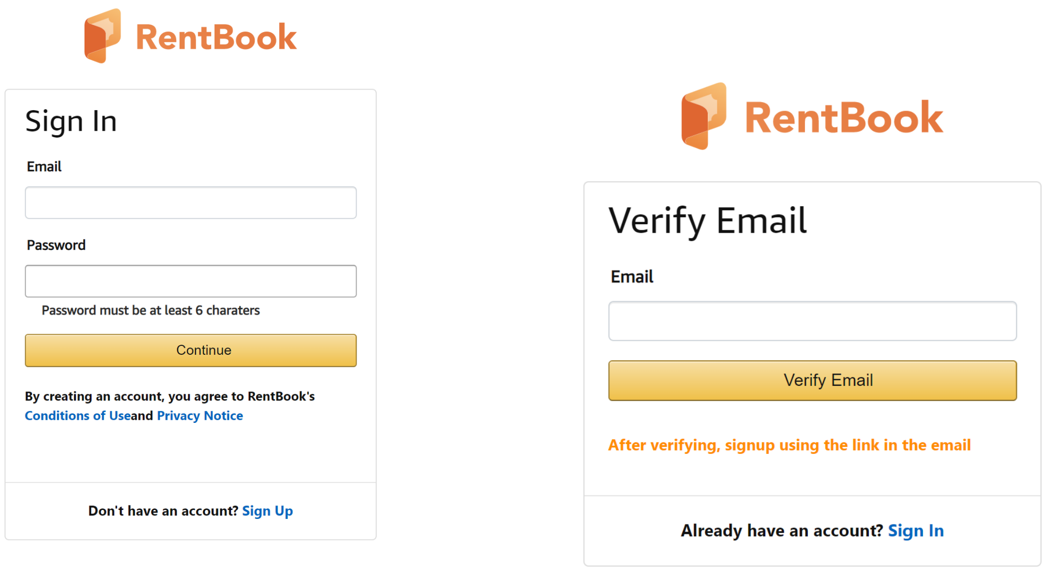


Browse for more Books Leaders of notable people, or Fantasy books Like harry potter or Game of thrones Best Fantasy books of the year so far

The Program
Inside the mind of the legendry Author JK Rowling

Batman Feb, 3 2020
by Batman
Ship to USA
Hardcover
$220.00 ~~242~~
Free with Audible Trial
Other Formats: Audio CD

Figure 8 Dashboard



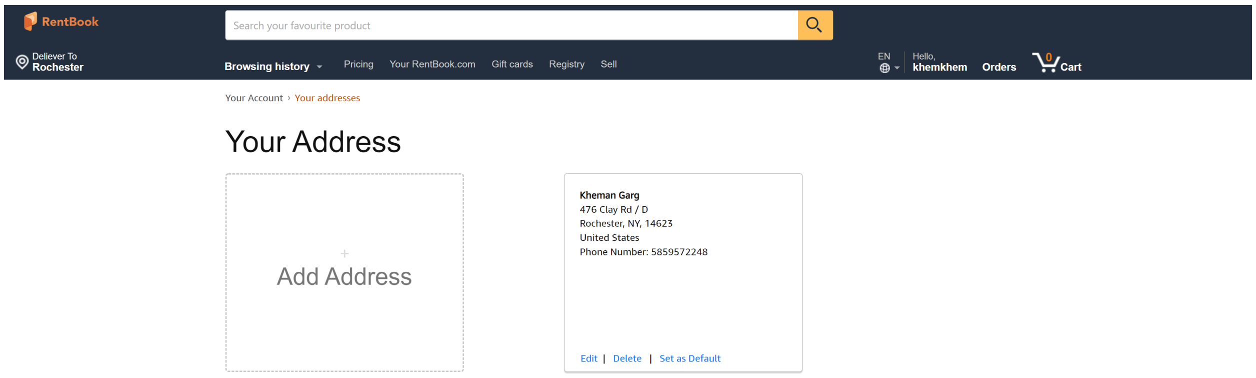
Password must be at least 6 charaters

By creating an account, you agree to RentBook's Conditions of Useand Privacy Notice

After verifying, signup using the link in the email

Figure 9 Sign-In and Sign-up Page

Once the user signs up they are able to explore the books and can even post the books to sell. To get the idea of the prices and learn more about the book they can note down the ISBN number from the book which will link them to open-source website called OpenLibrary.org that will show the selling prices and details about the book. The customer can explore the list of books and then add books to the cart whichever is liked by the customer. After which they can go to the cart and can check out. In order to deliver the book, it necessary to have an address which can also be done by adding the address.

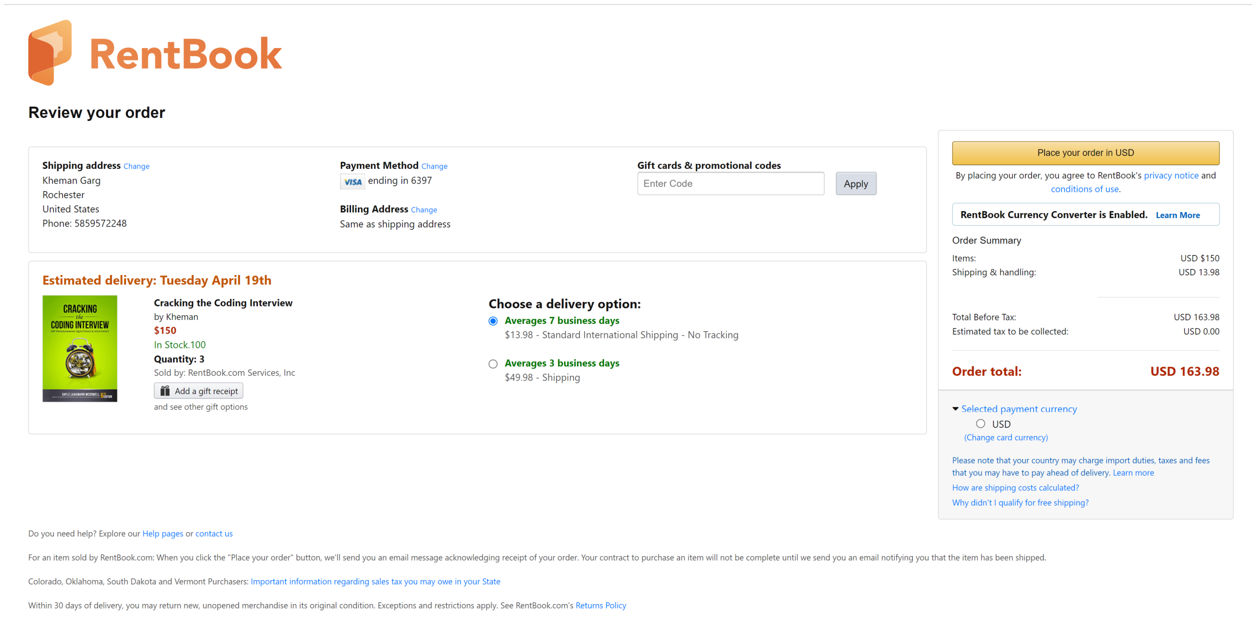


Your Address

Kheman Garg
476 Clay Rd / D
Rochester, NY, 14623
United States
Phone Number: 5859572248

Figure 10 Add/Edit Address Page

After that you can review your order once you go to the checkout section and choose the type of shipping between normal or express delivery and the price will change according to that option.



Review your order

Shipping address
Kheman Garg
Rochester
United States
Phone: 5859572248

Payment Method
VISA ending in 6397

Billing Address
Same as shipping address

Gift cards & promotional codes

Estimated delivery: Tuesday April 19th

Cracking the Coding Interview
by Kheman
$150
In Stock.100
Quantity: 3
Sold by: RentBook.com Services, Inc
and see other gift options

Choose a delivery option:
[✓ Averages 7 business days]
$13.98 - Standard International Shipping - No Tracking
Averages 3 business days
$49.98 - Shipping

By placing your order, you agree to RentBook's privacy notice and conditions of use.

RentBook Currency Converter is Enabled.

Order Summary

Items:	USD $150
Shipping & handling:	USD 13.98
Total Before Tax:	USD 163.98
Estimated tax to be collected:	USD 0.00
Order total:	**USD 163.98**

Selected payment currency
USD
(Change card currency)

Please note that your country may charge import duties, taxes and fees that you may have to pay ahead of delivery. Learn more
How are shipping costs calculated?
Why didn't I qualify for free shipping?

Do you need help? Explore our Help pages or contact us

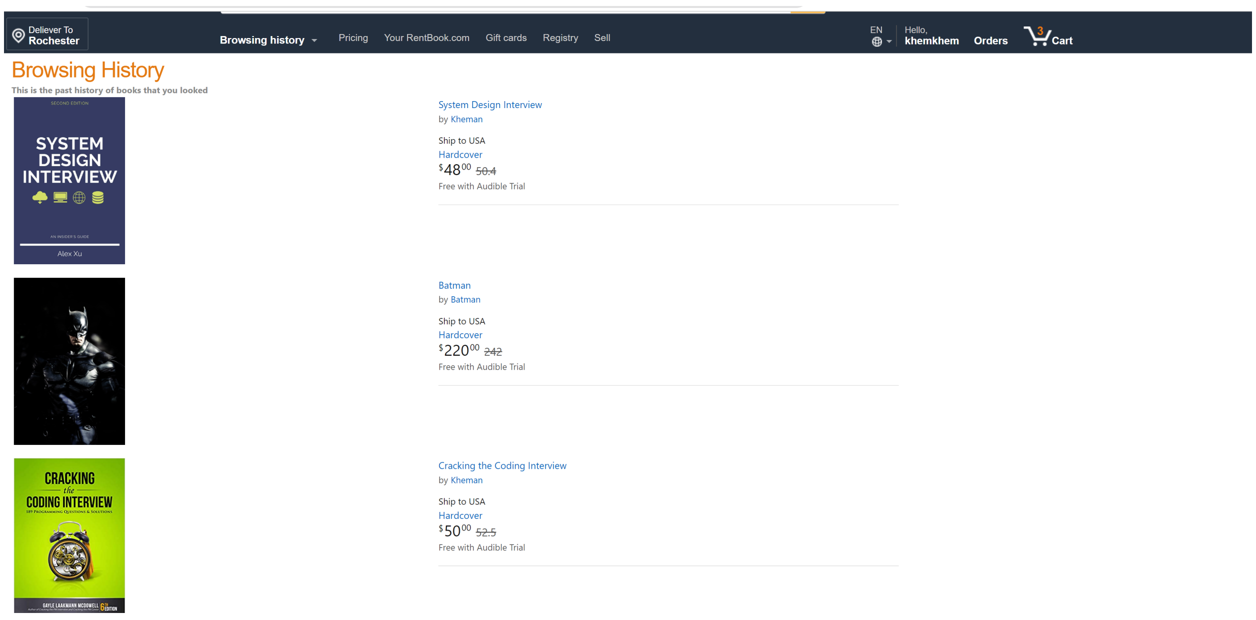
For an item sold by RentBook.com: When you click the "Place your order" button, we'll send you an email message acknowledging receipt of your order. Your contract to purchase an item will not be complete until we send you an email notifying you that the item has been shipped.

Colorado, Oklahoma, South Dakota and Vermont Purchasers: Important information regarding sales tax you may owe in your State

Within 30 days of delivery, you may return new, unopened merchandise in its original condition. Exceptions and restrictions apply. See RentBook.com's Returns Policy

Figure 11 Checkout Page

You can also check the browsing history page to see the history of your products in case if someone saw a product and want to revisit. This will help user to smoothly go through the pages. There is also top 2 search lookups present during the check out just in case user wants to revisit the products that were just looked.


Delever To Rochester | Browsing history | Pricing | Your RentBook.com | Gift cards | Registry | Sell | EN | Hello, khemkhem | Orders | 3 Cart

Browsing History

This is the past history of books that you looked

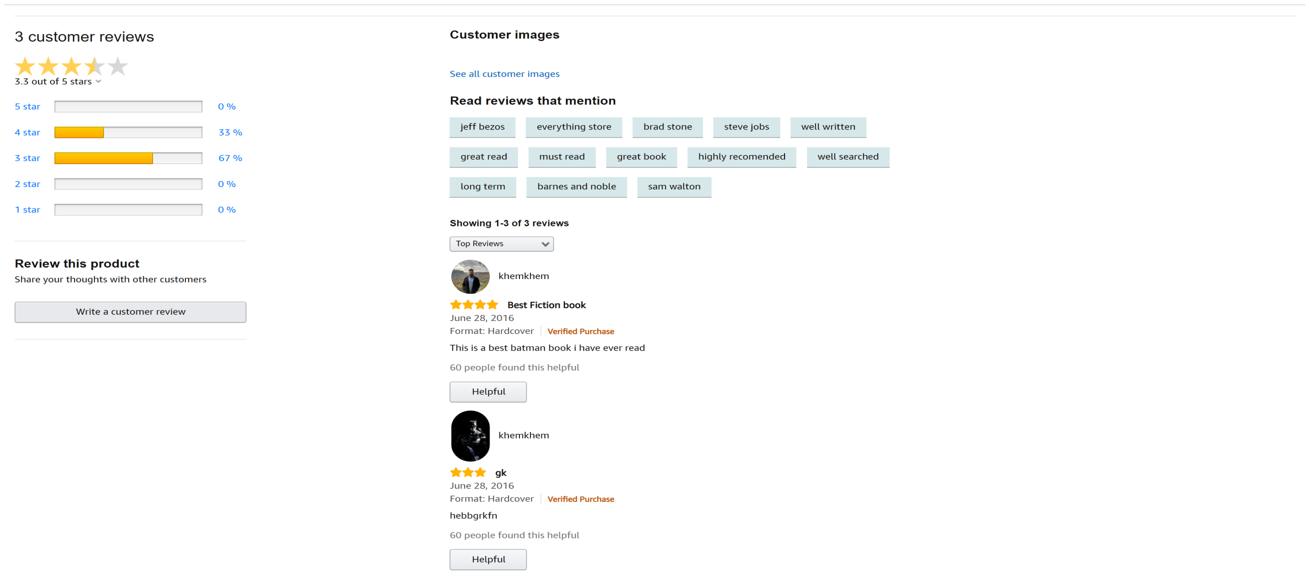
System Design Interview
by Kheman
Ship to USA
Hardcover
$48^00 ~~50.4~~
Free with Audible Trial

Batman
by Batman
Ship to USA
Hardcover
$220^00 ~~242~~
Free with Audible Trial

Cracking the Coding Interview
by Kheman
Ship to USA
Hardcover
$50^00 ~~52.5~~
Free with Audible Trial

Figure 12 Browsing History Page

You can also view each product by just clicking on the product. The page will show the details for that product and then there is also a review system in place that will help in reviewing the product and hence making it a verified product. Based on these, customers can decide if they want to buy the product.


3 customer reviews

3.3 out of 5 stars

5 star	0 %
4 star	33 %
3 star	67 %
2 star	0 %
1 star	0 %

Review this product

Share your thoughts with other customers

Write a customer review

Customer images

See all customer images

Read reviews that mention

jeff bezos | everything store | brad stone | steve jobs | well written | great read | must read | great book | highly recomended | well searched | long term | barnes and noble | sam walton

Showing 1-3 of 3 reviews

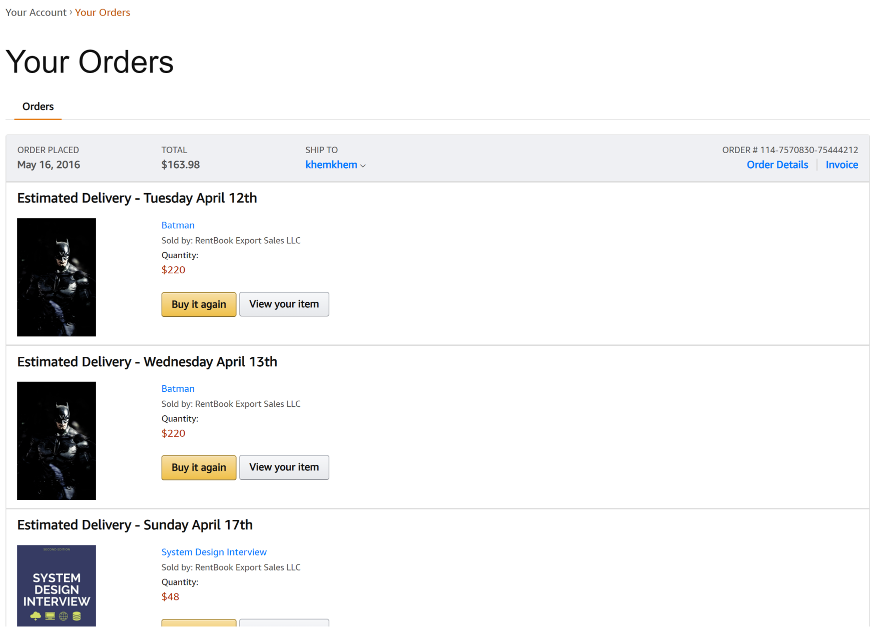
Top Reviews

khemkhem
Best Fiction book
June 28, 2016
Format: Hardcover | Verified Purchase
This is a best batman book i have ever read
60 people found this helpful
Helpful

khemkhem
gk
June 28, 2016
Format: Hardcover | Verified Purchase
hebbgrkfn
60 people found this helpful
Helpful

Figure 13 Review System Page

Finally, when the order is done and checked out. Then the user can go to payment page and pay for the product which is using Stripe API. After that user can go to You orders Page and check the orders list in that page.


Your Account › Your Orders

Your Orders

Orders

ORDER PLACED	TOTAL	SHIP TO	ORDER # 114-7570830-75444212
May 16, 2016	$163.98	khemkhem	Order Details · Invoice

Estimated Delivery - Tuesday April 12th

Batman
Sold by: RentBook Export Sales LLC
Quantity:
$220

Estimated Delivery - Wednesday April 13th

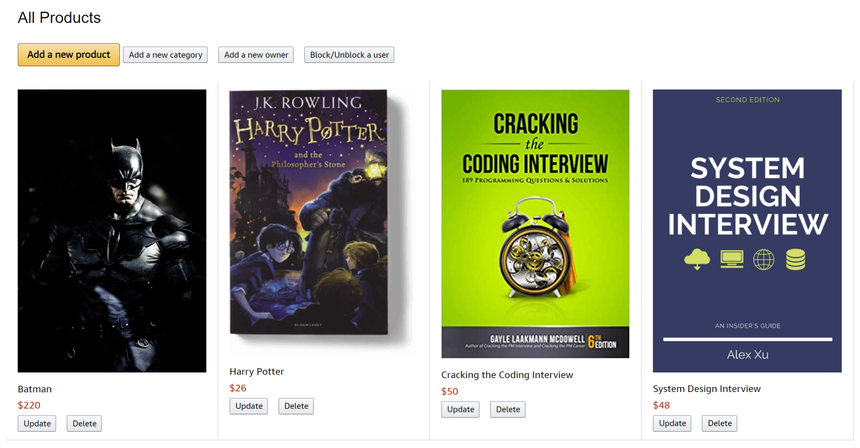
Batman
Sold by: RentBook Export Sales LLC
Quantity:
$220

Estimated Delivery - Sunday April 17th

System Design Interview
Sold by: RentBook Export Sales LLC
Quantity:
$48

Figure 14 Orders Page

This is the lifecycle of an end user as a client that this application can provide. Other than that, there is also an admin application which has functions like performing CRUD operations in the application for a product/owner/category. They also have the ability to block/unblock any user from accessing the website. Following is how the admin page looks like.


All Products

Batman
$220

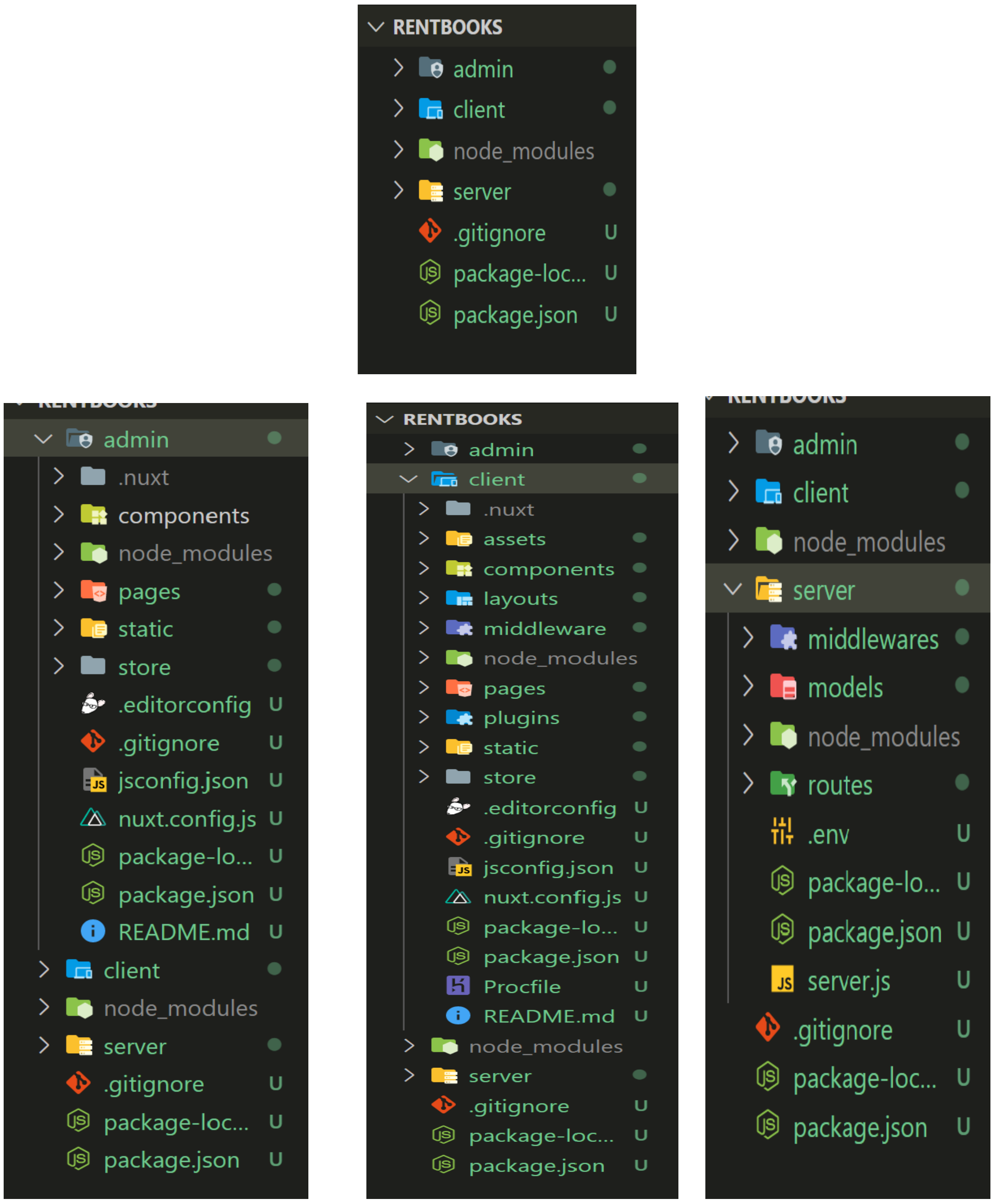
Harry Potter
$26

Cracking the Coding Interview
$50

System Design Interview
$48

Figure 15 Admin Page Dashboard

The code is divided in three different folders that are server, client and admin where server represents the backend, client and admin represents to different projects for the user interface of the end user. The project is running on three different portals. In locally it is running on port 3000, 9000, 6000 respectively but these ports can be change depending on the developer's preference. There are secret keys used in the server that can be dangerous if accessed by anyone outside the project so those are stores in a special file that can only be accessed by the developer in order to maintain security from harmful attacks.



```
RENTBOOKS
> admin
> client
> node_modules
> server
  .gitignore        U
  package-loc...    U
  package.json      U
```

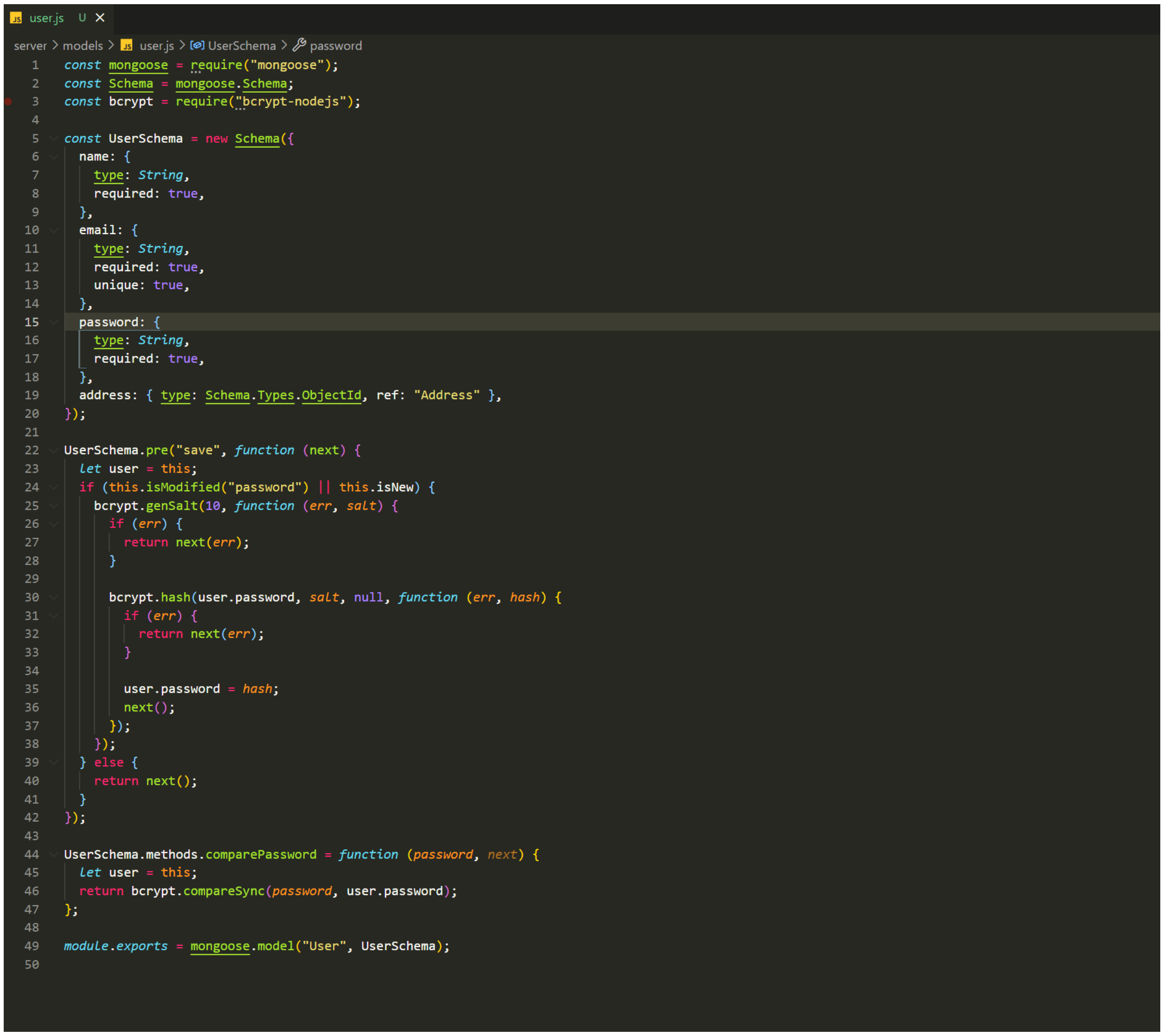
```
v admin
  > .nuxt
  > components
  > node_modules
  > pages
  > static
  > store
    .editorconfig   U
    .gitignore      U
    jsconfig.json   U
    nuxt.config.js  U
    package-lo...   U
    package.json    U
    README.md       U
> client
> node_modules
> server
  .gitignore        U
  package-loc...    U
  package.json      U
```

```
RENTBOOKS
> admin
v client
  > .nuxt
  > assets
  > components
  > layouts
  > middleware
  > node_modules
  > pages
  > plugins
  > static
  > store
    .editorconfig   U
    .gitignore      U
    jsconfig.json   U
    nuxt.config.js  U
    package-lo...   U
    package.json    U
    Procfile        U
    README.md       U
> node_modules
> server
  .gitignore        U
  package-loc...    U
  package.json      U
```

```
> admin
> client
> node_modules
v server
  > middlewares
  > models
  > node_modules
  > routes
    .env            U
    package-lo...   U
    package.json    U
    server.js       U
  .gitignore        U
  package-loc...    U
  package.json      U
```

Figure 16 Folder Structure of the application

The models are generated using a NPM library called Mongoose and a Schema is developed in the node.js that is also being imported to MongoDB when the data is posted on it. There are optional values that can enforce what kind of data is needed and help in the validation of the database.

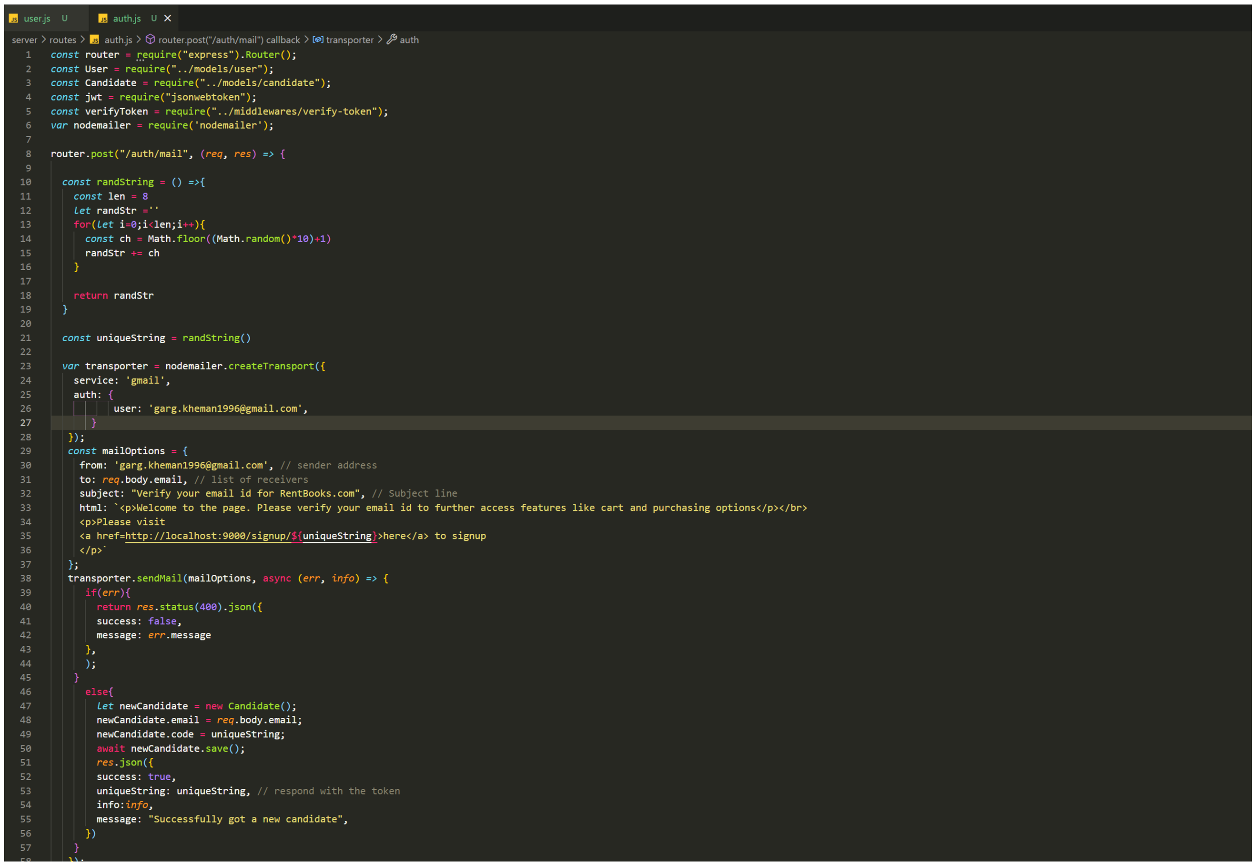


```
const mongoose = require("mongoose");
const Schema = mongoose.Schema;
const bcrypt = require("bcrypt-nodejs");

const UserSchema = new Schema({
  name: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  address: { type: Schema.Types.ObjectId, ref: "Address" },
});

UserSchema.pre("save", function (next) {
  let user = this;
  if (this.isModified("password") || this.isNew) {
    bcrypt.genSalt(10, function (err, salt) {
      if (err) {
        return next(err);
      }

      bcrypt.hash(user.password, salt, null, function (err, hash) {
        if (err) {
          return next(err);
        }

        user.password = hash;
        next();
      });
    });
  } else {
    return next();
  }
});

UserSchema.methods.comparePassword = function (password, next) {
  let user = this;
  return bcrypt.compareSync(password, user.password);
};

module.exports = mongoose.model("User", UserSchema);
```

Figure 17 User Model

In the above model user schema is defined which consists of name, email and password with everything being required. Also, before the data goes to the table the data gets modified like before saving the password will be encrypted using a library called bcrypt.js

Further these models are accessed using the endpoints which is the backend layer of the project which contains routes/controllers that contains endpoints and access the REST APIs. Below is the code for one of the endpoints that is being used in the application. This endpoint is consuming a library called node-mailer that can help in verifying the email and the we will send a unique code to the user's email that will help in identifying user.


user.js U auth.js U
server > routes > auth.js > router.post("/auth/mail") callback > transporter > auth

```
const router = require("express").Router();
const User = require("../models/user");
const Candidate = require("../models/candidate");
const jwt = require("jsonwebtoken");
const verifyToken = require("../middlewares/verify-token");
var nodemailer = require('nodemailer');

router.post("/auth/mail", (req, res) => {

  const randString = () =>{
    const len = 8
    let randStr =''
    for(let i=0;i<len;i++){
      const ch = Math.floor((Math.random()*10)+1)
      randStr += ch
    }

    return randStr
  }

  const uniqueString = randString()

  var transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
          user: 'garg.kheman1996@gmail.com',
      }
   });
   const mailOptions = {
     from: 'garg.kheman1996@gmail.com', // sender address
     to: req.body.email, // list of receivers
     subject: "Verify your email id for RentBooks.com", // Subject line
     html: `<p>Welcome to the page. Please verify your email id to further access features like cart and purchasing options</p></br>
     <p>Please visit
     <a href=http://localhost:9000/signup/${uniqueString}>here</a> to signup
     </p>`
   };
   transporter.sendMail(mailOptions, async (err, info) => {
      if(err){
        return res.status(400).json({
        success: false,
        message: err.message
      },
      );
    }
      else{
        let newCandidate = new Candidate();
        newCandidate.email = req.body.email;
        newCandidate.code = uniqueString;
        await newCandidate.save();
        res.json({
        success: true,
        uniqueString: uniqueString, // respond with the token
        info:info,
        message: "Successfully got a new candidate",
      })
    }
```

Figure 18 Authentication Endpoint

Further there is a client section which is using NUXT.js framework along with the bootstrap-vue library which is a framework for CSS. Along with that code also contains pages and components where pages represent the routes and components are reusable components that can be used on any page. Example for this let's take the profile page the html for that will look like this wrapped inside the template tags.

```
<template>
  <main>
    <div class="container-fluid">
      <div class="row">
        <div class="col-sm-6">
          <div class="a-spacing-top-medium"></div>
          <h2 class="text-center">Profile Page {{ loggedInUser.name }}</h2>
          <a href="#" @click="onLogout">Logout</a>
          <form action>
            <!-- Name -->
            <div class="a-spacing-top-medium">
              <label style="margin-bottom:0px;">Name</label>
              <input
                v-model="name"
                type="text"
                class="a-input-text"
                style="width:100%"
                :placeholder="loggedInUser.name"
              />
            </div>
            <!-- Email -->
            <div class="a-spacing-top-medium">
              <label style="margin-bottom:0px;">E-mail</label>
              <input
                v-model="email"
                type="email"
                class="a-input-text"
                style="width:100%"
                :placeholder="loggedInUser.email"
              />
            </div>

            <!-- Password -->
            <div class="a-spacing-top-medium">
              <label style="margin-bottom:0px;">Password</label>
              <input v-model="password" type="password" class="a-input-text"
style="width:100%" />
            </div>

            <hr />
            <!-- Button Submit -->
            <div class="a-spacing-top-large">
                  <span class="a-button-text"
@click="onUpdateProfile">Update</span>
            </div>
          </form>
          <br />
        </div>
      </div>
    </div>
  </main>
</template>
```

This file will also contain the javascript which contains the information that we are trying to call above in the brackets “{{loggedInUser.name}}”.

```
<script>
import { mapGetters,mapActions } from "vuex";
export default {
  data() {
    return {
      name: "",
      email: "",
      password: ""
    };
  },
  computed: {
    ...mapGetters(["loggedInUser"])
  },
  methods: {
    ...mapActions(["clearCache"]),
    async onUpdateProfile() {
      try {
        let data = {
          name: this.name,
          email: this.email,
          password: this.password
        };
        let response = await this.$axios.$put("/api/auth/user", data);
        if (response.success) {
          this.name = "";
          this.email = "";
          this.password = "";
          await this.$auth.fetchUser();
        }
      } catch (err) {
        console.log(err);
      }
    },
    async onLogout() {
      await this.clearCache();
      await this.$auth.logout();
    }
  }
};
</script>
```

This file contains data which contains the state of values that we are using in this page. Then there are methods that perform when the function is called by the end user. It contains mapGetters and mapActions, where the former one is to get some data from the centralized store whereas the later one is when you need to perform mutation to the data that is present in that vuex store.

8. FUTURE WORKS

The aim of this application is the help people getting books easily at cheaper price or selling the books if they do not require it. Even though the UI is built inspiring from amazon but it could become better. For the sake of user interface there are some hard coded texts which would be great to convert into more dynamic nature. Also, this application requires usability testing where representative users can try it and give advice on how the application can be improved in the perspective of user interface. Besides that, it also requires unit testing for both front-end and backend in order to see if there is any breaking point in the application. There is be a limited amount of data that will be used in the website due to time constraints and since the website will be just in a beta phase and not in production mode. But eventually this data could be increased and loading tests can be applied to application to see the amount of traffic that the website can handle. The delivery service could be improved in this as some delivery methods like getting books deliver through UPS or any other postal service could be an interesting add on. As this will make easier for end users who are selling books easier and more affordable.

The application currently uses NoSQL as the database but in future some part of the application like payment gateway can be migrated to SQL relational database because the relational database is helpful and safer for when it comes to financial tasks as they support property called Atomicity, consistency, isolation and durability (ACID). Therefore, some part of the application can use SQL which directly leads to the next concept which is microservices application. Currently the application has a single codebase and is therefore monolithic application. But in future as the services and features increases along with the requirement of high scalability then the application can be broken down to different parts where each part is scaled and is easily pluggable with other services and hence convert it to a microservices architecture. This change can help bring more tech stack and reusable services bring in the application.

Furthermore, the application is currently built upon REST API which is currently protected through authentication and authorization. But still there could be data leaks in the application that are not known at this point. Data leaks can be extremely bad for the website as it makes the application vulnerable. In order to stop these leaks, it will require intensive testing. There is some data that can be stored in the database instead of the local storage which could be vulnerable. Hence making those changes would be a good way to start removing data from the reach of any end user.

9. CONCLUSION

Idea is to build a customer-to-customer Bookstore e-commerce application that can help students in lending/renting books easily online in a small area. This will help students or any user to save up money and resources. This application is also helping the reuse of books rather than throwing them. There are be three type of roles that are using this application buyer, seller and admin. The development of the application was a great part of learning as teaches a lot about the development lifecycle. It also taught how MVC architecture can be implemented to create a real-life project. There are some standards that were being followed while building the application one of them was to make the code write in a correct code format. To follow this a package called prettier and pretty-quick were installed. These packages helped in making the code look well formatted and easier to understand for anyone who reads it. A good amount of suggestion was provided by Professors that helped in shaping the application.

The application is built on a lot of different APIs and technology stack hence Full Stack application. All the third-party APIs contains sensitive keys that are hidden and are only available to the developer. There were some challenging tasks like shaping the database and then structuring the data model in order to consume the REST APIs. Lot of efforts were put in researching the third-party APIs that will be appropriate for this application and then those were filtered. There was a tough decision on selecting the frontend framework but since I wanted the application to be simple and light weighted therefore, I chose NUXT.js which comes with Vue.js framework and Vuex. It decreased the amount of work for routing. This website is inspired from amazon because that application also first started with the books so that was inspiration point behind the design. The application pages were divided in different components in order to reuse them which saves a lot of time and effort.

This project also helped me in learning about external technologies that are used to build this application. For example, postman was heavily used to build endpoints. Even though I have worked on postman before but for this application I tried some different options that made me good with postman and its functionality. I learned how the documentations and stack overflow can be helpful for any developer who is building project. Overall, this project has been a perfect opportunity for me as it summed up my learning from the Masters in Information in Science and Technology program and also made me one step closer to deal with the real-world web applications.

9. REFERENCES

1. Abdala, Mohammed & Khider, Noor. (2011). Online E-Book Store Website Design. i-manager's journal on software engineering. 5. 41-46. 10.26634/jse.5.4.1449.

2. Deepanvita Paliwal, B. Sreevidya, "Design and development of an intelligent web application for a direct consumer to consumer trading over the Internet", *Inventive Communication and Computational Technologies (ICICCT) 2017 International Conference on*, pp. 56-58, 2017.

3. Department of Information Technology Vellalar College of Engineering and Technology, Thindal, Erode, Tamilnadu, India, Pin Code: 638012, and M Sindhia. "BOOKLAND – AN ANDROID APPLICATION FOR RENTAL BOOKS." *GEDRAG & ORGANISATIE REVIEW* 33, no. 02 (May 14, 2020). https://doi.org/10.37896/GOR33.02/139.

4. Gil, Joon-Min, JongBum Lim, and Dong-Mahn Seo. "Design and Implementation of MapReduce-Based Book Recommendation System by Analysis of Large-Scale Book-Rental Data." In *Advanced Multimedia and Ubiquitous Engineering*, edited by James J. (Jong Hyuk) Park, Hai Jin, Young-Sik Jeong, and Muhammad Khurram Khan, 713–19. Singapore: Springer Singapore, 2016.

5. Juszczuk, D., Tarnawski, J., Karla, T., & Duzinkiewicz, K. (2017, June). Real-time basic principles nuclear reactor simulator based on client-server network architecture with WebBrowser as user interface. In *Polish Control Conference* (pp. 344-353). Springer, Cham.

6. Majeed, A., & Rauf, I. (2018). MVC architecture: a detailed insight to the modern web applications development. *Peer Review Journal of Solar & Photoenergy Systems*, *1*(1), 1-7.

7. Banker, K., Garrett, D., Bakkum, P., & Verch, S. (2016). *MongoDB in Action: Covers MongoDB version 3.0*. Simon and Schuster.

8. Liang, L., Zhu, L., Shang, W., Feng, D., & Xiao, Z. (2017, May). Express supervision system based on NodeJS and MongoDB. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)* (pp. 607-612). IEEE.

9. Jewell, J., & Marden, M. (2018). The business value of the stripe payments platform.

10. Masse, M. (2011). *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc.".